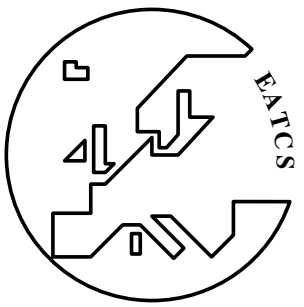
43rd International Colloquium on Automata, Languages, and Programming

ICALP 2016, Rome, Italy, July 12–15, 2016

Edited by

Ioannis Chatzigiannakis
Michael Mitzenmacher
Yuval Rabani
Davide Sangiorgi



Editors

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

Michael Mitzenmacher
School of Engineering and Applied Sciences
Harvard University
michaelm@eecs.harvard.edu

Yuval Rabani
Computer Science and Engineering
The Hebrew University of Jerusalem
yrabani@cs.huji.ac.il

Davide Sangiorgi
Department of Computer Science
University of Bologna
davide.sangiorgi@gmail.com

ACM Classification 1998
F. Theory of Computation

ISBN 978-3-95977-013-2

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at http://www.dagstuhl.de/dagpub/978-3-95977-013-2.

Publication date
August, 2016

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

Digital Object Identifier: 10.4230/LIPIcs.ICALP.2016.0

ISBN 978-3-95977-013-2 **ISSN 1868-8969** **http://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

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

ISSN 1868-8969

http://www.dagstuhl.de/lipics

Contents

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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

Track C: Foundations of Networked Computation: Models, Algorithms and Information Management

Preface

ICALP 2016, the 43rd edition of the International Colloquium on Automata, Languages and Programming, was held in Rome, Italy during July 12–15, 2016. ICALP is a series of annual conferences of the European Association for Theoretical Computer Science (EATCS), which first took place in 1972. This year, the ICALP program consisted of the established track A (focusing on algorithms, automata, complexity, and games) and track B (focusing on logic, semantics, and theory of programming), and of the recently introduced track C (focusing on foundations of networking). In response to the call for papers, the Program Committee received 515 submissions, the highest ever: 319 for track A, 121 for track B, and 75 for track C. Out of these, 146 papers were selected for inclusion in the scientific program: 89 papers for Track A, 36 for Track B, and 21 for Track C. The selection was made by the Program Committees based on originality, quality, and relevance to theoretical computer science. The quality of the manuscripts was very high indeed, and many deserving papers could not be selected.

The EATCS sponsored awards for both a best paper and a best student paper for each of the three tracks, selected by the Program Committees. The best paper awards were given to the following papers:

- Track A: Andreas Galanis, Andreas Göbel, Leslie Ann Goldberg, John Lapinskas and David Richerby. "Amplifiers for the Moran Process".
- Track A: Neeraj Kayal, Chandan Saha and Sébastien Tavenas. "An almost Cubic Lower Bound for Depth Three Arithmetic Circuits".
- Track B: Olivier Bournez, Daniel Graça and Amaury Pouly. "Polynomial Time corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length".

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

- Track A: Samuel Hetterich. "Analysing Survey Propagation Guided Decimation on Random Formulas".
- Track C: Keerti Choudhary. "An Optimal Dual Fault Tolerant Reachability Oracle".

Apart from the contributed talks, ICALP 2016 included invited presentations by Devavrat Shah, Xavier Leroy, Subhash Khot and Marta Z. Kwiatkowska. Abstracts of their talks are included in these proceedings as well. The program of ICALP 2016 also included presentation of the EATCS Award 2016 to Dexter Kozen, the Gödel Prize 2016 to Steve Brookes and Peter O'Hearn, and the Presburger Award 2016 to Mark Braverman.

This volume of the proceedings contains all contributed papers presented at the conference together with the papers and abstracts of the invited speakers.

We wish to thank all authors who submitted extended abstracts for consideration, the Program Committees for their scholarly effort, and all referees who assisted the Program Committees in the evaluation process. We thank the sponsors (Microsoft; Microsoft Research; AICA, Facebook; Department of Informatics, Sapienza University of Rome; and Austrian) for their support. We are also grateful to Tiziana Calamoneri, Irene Finocchi, Nicola Galesi and Daniele Gorla for organizing ICALP 2016 and all the support staff of the Organizing Committee.

Thanks to Andrei Voronkov for writing the conference management system EasyChair, which was used in handling the submissions and the electronic Program Committee meeting,

as well as in assisting in the assembly of the proceedings. Last but not least, we would like to thank Luca Aceto, the president of EATCS, for his generous advice on the organization of the conference and Efi Chita and the secretary office of EATCS for their support in the preparation of the proceedings.

July 2016

Ioannis Chatzigiannakis
Michael Mitzenmacher
Yuval Rabani
Davide Sangiorgi

Organization

Program Committee

Track A

Yuval Rabani	The Hebrew University of Jerusalem, Israel, Chair
Susanne Albers	TU Munchen, Germany
Andris Ambainis	University of Latvia, Latvia
Per Austrin	KTH Royal Institute of Technology, Sweden
Harry Buhrman	Centrum Wiskunde & Informatica, Netherlands
Elisa Celis	EPFL, Switzerland
Nicolò Cesa-Bianchi	University of Milano, Italy
Marek Cygan	University of Warsaw, Poland
Ilias Diakonikolas	University of Southern California, USA
Josep Diaz	University Polytechnica de Catalunya, Spain
Benjamin Doerr	Ecole Polytechnique, France
Dimitris Fotakis	NTUA, Greece
Anna Gal	University of Texas at Austin, USA
Cyril Gavoille	University of Bordeaux, France
Fabrizio Grandoni	IDSIA, Switzerland
Iftach Haitner	Tel Aviv University, Israel
Monika Henzinger	University of Vienna, Austria
Rahul Jain	National University of Singapore, Singapore
Ken-Ichi Kawarabayashi	National Institute of Informatics, Japan
Piotr Krysta	University of Liverpool, UK
François Le Gall	University of Tokyo, Japan
Stefano Leonardi	University of Roma 1, Italy
Jian Li	Tsinghua University, China
Nutan Limaye	Indian Institute of Technology, India
Satya Lokam	Microsoft Research, India
Raghu Meka	UCLA, USA
Lorenzo Orecchia	Boston University, USA
Rotem Oshman	Tel Aviv University, Israel
Giuseppe Persiano	University of Salerno, Italy
Nikhil Srivastava	University of California, Berkeley, USA
Mikkel Thorup	University of Copenhagen, Denmark
Dominique Unruh	University of Tartu, Estonia
Justin Ward	EPFL, Switzerland

Track B

Davide Sangiorgi	Bologna, Italy, Chair
Parosh Aziz Abdulla	University Uppsala
Tomás Brázdil	Brno, Czech Republic
Arnaud Carayol	CNRS, Marne-La-Vallee, France

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Taolue Chen	Oxford, UK
Silvia Crafa	Padova, Italy
Loris D'Antoni	University of Pennsylvania, USA
Pedro D'Argenio	Córdoba, Argentina
Yuxin Deng	Shanghai, China
Maribel Fernández	King's College London, UK
Matthew Hague	University of London, UK
Anna Ingólfsdóttir	Reykjavik, Iceland
Jarkko Kari	University of Turku, Finland
Joost-Pieter Katoen	Aachen University, Germany
Barbara Konig	University Duisburg-Essen, Germany
Bartek Klin	University of Warsaw, Poland
Parthasarathy Madhusudan	University of Illinois, USA
Massimo Merro	Verona, Italy
Stephan Merz	Inria Nancy, France
Madhavan Mukund	Chennai Mathematical Institute, India
Filip Murlak	University of Warsaw, Poland
Aleksandar Nanevski	Madrid Institute of Advanced Studies, Spain
C.-H. Luke Ong	University of Oxford, UK
J. Perez	Groningen, Netherlands
Damien Pous	Lyon, France
Jakob Rehof	Dortmund, Germany
Tachio Terauchi	Jaist, Japan

Track C

Michael Mitzenmacher	Harvard University, USA, Chair
Luca Becchetti	Sapienza University of Rome, Italy
Shuchi Chawla	USA
Krishnendu Chatterjee	IST Austria
Lap Chi Lau	Chinese University Hong Kong/Waterloo
Flavio Chierichetti	Italy
Graham Cormode	University of Warwick, UK
Edith Elkind	UK
Keren-Censor Hillel	Israel
Martin Hoefer	Germany
Valerie King	Canada
Marc LeLarge	INRIA
Katrina Ligett	USA
Cris Moore	USA
Thomas Moscibroda	Microsoft Research, China
Rasmus Pagh	Denmark
Rajmohan Rajaraman	USA
Justin Thaler	Yahoo Labs, USA
Udi Wieder	VMware Research, USA

Organizing Committee

Tiziana Calamoneri	Sapienza University of Rome, Italy
Daniele Gorla	Sapienza University of Rome, Italy
Irene Finocchi	Sapienza University of Rome, Italy
Nicola Galesi	Sapienza University of Rome, Italy

Financial Sponsors

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

Additional Reviewers

A.V. Sreejith
Aaron Bernstein
Abbas Mehrabian
Abhishek Jain
Achim Blumensath
Adam Kasperski
Adam Meyerson
Adam O'Neill
Adi Rosén
Adrian Vetta
Aggelos Kiayias
Ágnes Cseh
Ahmed Rezine
Aiswarya Cyriac
Akitoshi Kawamura
Alan Roytman
Alberto Marchetti-Spaccamela
Alejandro Lopez-Ortiz
Alejandro Sanchez
Aleksandrs Belovs
Aleksi Saarela
Alessandra Cherubini
Alessandro Chiesa
Alex Simpson
Alexander Holroyd
Alexander Schiendorfer
Alexander Sherstov
Alexander Wolff
Alexandr Andoni
Ali Sezgin
Alina Ene
Allan Sly
Allyx Fontaine
Amaldev Manuel
Amalia Duch
Amir Abboud
Amir Nayyeri
Amirali Abdullah
Amit Deshpande
Amit Kumar
Amitabh Trehan
Amos Fiat
Amy Glen
Anastasios Sidiropoulos
André van Renssen
Andrea Asperti
Andrea Clementi
Andrea Turrini
Andreas Emil Feldmann
Andreas Galanis
Andreas Pavlogiannis
Andrei Bulatov
Andrei Krokhin
Andrej Bogdanov
Andrej Dudenhefner
Andrew McGregor
Andrew Winslow
Angelo De Caro
Angelo Fanelli
Anke van Zuylen
Ankur Moitra
Anne Broadbent
Anthony Widjaja Lin
Antonin Kucera
Antonio Blanca
António Ravara
Antonis Antonopoulos
Anup Rao
Anupam Gupta
Aranyak Mehta
Archontia Giannopoulou
Ariel Gabizon
Aristides Gionis
Arkadev Chattopadhyay
Armin Weiss
Arnab Bhattacharyya
Artem Khyzha
Artur Czumaj
Arturs Backurs
Ashish Chiplunkar
Ashutosh Rai
Ashutosh Trivedi
Aurélien Lemay
Avishay Tal
B Srivathsan
Bai Xue
Balasubramanian Sivan
Barnaby Martin
Bart M. P. Jansen

Benjamin Miller
Benjamin Monmege
Benjamin Moseley
Benny Pinkas
Benoît Valiron
Bernard Boigelot
Bernard Chazelle
Bernardo Toninho
Bernhard Steffen
Bingkai Lin
Bingtian Xue
Bireswar Das
Bojana Kodric
Brendan Lucier
Bruno Loff
C. Aiswarya
Cameron Musco
Carla Ferreira
Carlo Blundo
Carlo Mereghetti
Carme Alvarez
Carmine Ventre
Carroll Morgan
Catherine Greenhill
Cewei Cui
Chaitanya Swamy
Chandra Chekuri
Chandra Thapa
Charles Paperman
Chien-Chung Huang
Chris Heunen
Christian Dehnert
Christian Schaffner
Christian Scheffer
Christian Sommer
Christof Löding
Christoph Dürr
Christoph Matheja
Christopher Broadbent
Christopher Portmann
Christos Tzamos
Clément Canonne
Colin White
Cong Quy Trinh
Conrado Martínez
Cristian Riveros
Cristina Fernandes
Cristobal Rojas
Damian Straszak
Dana Fisman
Daniel Gottesman
Daniel Kane
Daniel Keren
Daniel Lokshtanov
Dániel Marx
Daniel Nagaj
Daniel Wichs
Daniela Petrisan
Daniele Venturi
Danny Hermelin
Danupon Nanongkai
Dario Della Monica
Dariusz Leniowski
Darren Strash
David Adjiashvili
David Eppstein
David Peleg
David R. Wood
David Richerby
David Woodruff
Davide Bilò
Davide Bresolin
Deeparnab Chakrabarty
Dejan Nickovic
Denis Kuperberg
Didier Caucal
Dieter Mitsche
Dimitrios Letsios
Dimitrios Thilikos
Dimitris Achlioptas
Dimitris Chatzidimitriou
Dimitris Paparas
Dimitris Tsipras
Diodato Ferraioli
Dirk Sudholt
Divesh Aggarwal
Dmitriy Traytel
Dmitry Chistikov
Domagoj Vrgoc
Dominik D. Freydenberger
Dominik Scheder
Dominique Schroeder
Ehsan Emamjomeh-Zadeh
Elaine Pimentel
Eli Ben-Sasson
Elias Koutsoupias
Emanuele Natale
Emmanouil Zampetakis
Emmanuel Beffara
Emmanuel Hainry
Eran Omri
Erez Kantor
Eric Blais
Éric Colin de Verdière
Eric Ruppert
Ernst Moritz Hahn
Eryk Kopczynski
Estela Rodrigues
Esther Arkin
Eugene Asarin
Eun Jung Kim
Eylon Yogev
Fahad Panolan
Federico Olmedo
Fedor Fomin
Florian Horn
Florian Speelman
Florian Steinberg
Francesco Pasquale
Francesco Ranzato
Franck Cassez
Frank Pfenning
Frank Stephan
Frank Valencia
Gabor Ivanyos
Gabriele Fici
Gautham Shenoy R
Geevarghese Philip
Georg Zetzsche
George Christodoulou
George Mertzios
Georgios Piliouras
Georgios Stamoulis
Gerard Renardel De Lavalette
Gil Cohen
Gilberto Filé
Gillat Kol
Giorgio Bacci
Giovanna Rosone
Gopal Pandurangan
Gourab Ghoshal
Gramoz Goranci
Grant Schoenebeck
Greg Zaverucha
Gruia Calinescu
Guido Proietti
Guillem Perarnau

Guy Kortsarz
Haitao Wang
Hamza Fawzi
Hans-Joachim Boeckenhauer
Harumichi Nishimura
Henrik Björklund
Hossein Jowhari
Hugo A. López
Iddo Tzameret
Igor Sergeev
Ilan Komargodski
Ioannis Chatzigiannakis
Irene Finocchi
Ivan Visconti
Jaikumar Radhakrishnan
James Brotherston
James Worrell
Jan Bessai
Janardhan Kulkarni
Jean-Eric Pin
Jennifer Iglesias
Jerome Feret
Jian Ding
Jiong Guo
Joachim Spoerhase
Johannes Blömer
John Iacono
Jop Briet
Joshua Grochow
Julia Chuzhoy
Julien Lange
Kamal Lodaya
Ken Clarkson
Khaled Elbassioni
Klaus Jansen
Krzysztof Onak
Laure Daviaud
Leen Stougie
Lehilton L. C. Pedrosa
Leroy Chew
Li-Yang Tan
Lila Fontes
Lorenzo Clemente
Luc Dartois
Lucian Ilie
Luigi Santocanale
M. Praveen
Magnus Wahlström
Mamadou Moustapha Kanté
György Dósa
Hamed Amini
Hannah Cairns
Haris Angelidakis
Heiko Röglin
Holger Dell
Hsien-Chih Chang
Hugo Gimbert
Ignacio Fábregas
Igor Walukiewicz
Ilia Gorelik
Ioannis Giotis
Irit Dinur
Ivkin Nikita
Jakub Łącki
James Laird
Jamie Vicary
Jan Kretinsky
Janos Varga
Jean-Marc Talbot
Jeremy Karp
Jérôme Leroux
Jin-Yi Cai
Jiri Srba
Joanna Ochremiak
Johannes Carmesin
Jon Kleinberg
José Verschae
Joshua Sack
Julian Mestre
K. Narayan Kumar
Karl Bringmann
Kentaro Honda
Kim Thang Nguyen
Koen Groenland
Kunihiko Sadakane
Laurent Doyen
Leen Torenvliet
Lei Song
Leslie Ann Goldberg
Liam Roditty
Linda Farczadi
Loukas Georgiadis
Luca Moscardelli
Luciano Gualà
Łukasz Jeż
Maciej Skórski
Mahdi Cheraghchi
Manfred Schmidt-Schauss
Haim Kaplan
Hamidreza Jahanjou
Hans L. Bodlaender
Hartmut Klauck
Henning Urbat
Hossein Esfandiari
Huan Long
Ichiro Hasuo
Ignaz Rutter
Ilan Cohen
Ines Marusic
Ioannis Panageas
Ishay Haviv
Jacobo Torán
James Aspnes
James Lee
Jan Arne Telle
Jan Obdrzalek
Jaroslaw Byrka
Jelena Marasevic
Jeroen Zuiddam
Jesper Nederlof
Jingcheng Liu
Jittat Fakcharooenphol
Joe Sawada
John Fearnley
Joost Winter
Joshua Brody
Juhani Karhumaki
Julien Cervelle
Kai Salomaa
Katsuhisa Yamanaka
Kevin Buchin
Kirstin Peters
Konstantinos Panagiotou
Lars Jaffke
Leah Epstein
Lefteris Kirousis
Lelia Blin
Leszek Kolodziejczyk
Libor Barto
Lionel Rieg
Lubos Korenciak
Luca Tesei
Ludwig Schmidt
Lukasz Kowalik
Madhur Tulsiani
Makrand Sinha
Marc Zeitoun

Marcin Pilipczuk
Marcin Wrochna
Marco Bernardo
Marco Chiesa
Marek Adamczyk
Margus Veanes
Maria Serna
Marie Van Den Bogaard
Marie-Pierre Béal
Maris Ozols
Mark Jerrum
Markus Blaeser
Markus Chimani
Markus Holzer
Markus Lohrey
Martin Beaudry
Martin Dyer
Martin Farach-Colton
Martin Gairing
Martin Grohe
Martin Lang
Martin Ziegler
Marvin Künnemann
Massimo Lauria
Mathieu Hoyrup
Mathieu Lauriere
Matias Korman
Matteo Mio
Matthew Johnson
Matthias Englert
Matthias Függer
Matthias Mnich
Matthias Westermann
Matúš Mihalák
Mauricio Ayala-Rincon
Maxim Sviridenko
Maya Stein
Mayank Goswami
Melanie Schmidt
Michael A. Forbes
Michael B. Cohen
Michael Ben Or
Michael Brautbar
Michael Dinitz
Michael Elberfeld
Michael Kapralov
Michael Lampis
Michael Luttenberger
Michael Mislove
Michael Vanden Boom
Michael Walter
Michal Koucky
Michał Pilipczuk
Michele Pagani
Michiel Smid
Mika Göös
Mikko Koivisto
Milan Bradonjic
Mimmo Parente
Moez Draief
Mohammad Ali Abam
Mohammad Salavatipour
Mohammadtaghi Hajiaghayi
Mohsen Ghaffari
Mohsen Rezapour
Monaldo Mastrolilli
Moni Naor
Mordecai J. Golin
Mukund Raghothaman
Naonori Kakimura
Nathan Ross
Naveen Garg
Neal Young
Neeldhara Misra
Neeraj Kayal
Neil Olver
Neil Thapen
Nengkun Yu
Nicolas Basset
Nicolas Gillis
Nicolas Ollinger
Nikhil Balaji
Nikolaos Fountoulakis
Nikolay Vereshchagin
Nikos Parotsidis
Nima Anari
Ning Xie
Nir Bitansky
Nir Piterman
Nishanth Chandran
Nisheeth Vishnoi
Noah Stephens-Davidowitz
Ocan Sankur
Oded Lachish
Oded Maler
Ofer Neiman
Oliver Friedmann
Oliver Schaudt
Olivier Carton
Olivier Serre
Omar Fawzi
Orna Kupferman
Orr Fischer
Ozan Kahramanogullari
Pablo Barceló
Parasara Sridhar Duggirala
Paresh Nakhe
Partha Mukhopadhyay
Pascal Fontaine
Pascal Schweitzer
Pascal Vanier
Patricia Bouyer
Patrick Totzke
Paul Bell
Paul Brunet
Paul Goldberg
Paulo Oliva
Pavel Hubacek
Pavol Hell
Pawel Gawrychowski
Pawel Komosa
Paweł Parys
Pedro Sánchez Terraf
Peter Jeavons
Peter Kostolányi
Petr Novotný
Philip Bille
Philip Klein
Philipp Kindermann
Pierre Fraigniaud
Pierre Ganty
Pietro Ferrara
Piotr Mardziel
Piyush Srivastava
Prahladh Harsha
Prakash Panangaden
Prakash Saivasan
Praneeth Netrapalli
Prasad Raghavendra
Pravesh Kothari
Przemysław Uznański
Qi Cheng
Qiang Zhang
Qin Zhang
R. Ramanujam
Rachel Cummings
Radha Jagadeesan

Radu Curticapean
Radu Mardare
Raghav Kulkarni
Raghunath Tewari
Rahul Santhanam
Rahul Shah
Rajesh Chitnis
Ran Cohen
Ran Duan
Ranko Lazic
Raphael Clifford
Raphaël Jungers
Rasmus Ibsen-Jensen
Ravishankar Krishnaswamy
Rene Sitters
Richard Mayr
Richard Peng
Rina Panigrahy
Rishi Saket
Rob van Stee
Robert Elsässer
Robert Ganian
Robert Kleinberg
Robert Krauthgamer
Roberto Solis-Oba
Rocco Servedio
Rodrigo de Souza
Rogério Reis
Rohit Gurjar
Ronald de Wolf
Rotem Oshman
Roy Schwartz
Ruiwen Chen
Runwei Zhang
Ryan O'Donnell
Ryan Williams
Ryuhei Uehara
S P Suresh
Sabina Rossi
Sabine Storandt
Saket Saurabh
Salvatore Ingala
Samir Datta
Samuel J. van Gool
Sangxia Huang
Sanjeev Khanna
Sayan Bhattacharya
Sebastian Krinninger
Sebastian Maneth
Sebastian Siebertz
Sébastien Tavenas
Seeun William Umboh
Seffi Naor
Seiichiro Tani
Sepehr Assadi
Serge Fehr
Seth Fogarty
Seth Pettie
Sevag Gharibian
Shaull Almagor
Shay Gershtein
Shay Kutten
Shi Li
Shiri Chechik
Shivam Garg
Shuichi Miyazaki
Shweta Agrawal
Siddhartha Banerjee
Sigal Oren
Silke Czarnetzki
Simon Korman
Simone Rinaldi
Siu On Chan
Søren Dahlgaard
Sourav Chakraborty
Souymodip Chakraborty
Spyros Angelopoulos
Srikanth Srinivasan
Srinivasan Arunachalam
Stanislav Živný
Stasys Jukna
Stefan Dziembowski
Stefan Göller
Stefan Kiefer
Stefan Kratsch
Stefan Neumann
Stefano Galatolo
Stephan Kreutzer
Stephane Gaubert
Stephen Alstrup
Stephen Chestnut
Steve Butler
Steven Ramsay
Stratis Skoulakis
Subrahmanyam Kalyanasundaram
Suguru Tamaki
Sumedh Tirodkar
Sune K. Jakobsen
Surender Baswana
Svante Janson
Sven Schewe
Swastik Kopparty
Swen Jacobs
Sylvain Salvati
Sylvain Schmitz
Szabolcs Ivan
Szymon Toruńczyk
Tal Wagner
Telikepalli Kavitha
Tero Harju
Thanasis Lianeas
Thao Dang
Thatchaphol Saranurak
Themistoklis Gouleakis
Thomas Colcombet
Thomas Erlebach
Thomas Lidbetter
Thomas Noll
Thomas Place
Thomas Sauerwald
Thomas Schwentick
Thomas Vidick
Thomas Watson
Thore Husfeldt
Tianyi Zhang
Till Tantau
Timothy M. Chan
Tobias Mömke
Tobias Mueller
Tom Bannink
Tom Hayes
Tom Hirschowitz
Tomasz Jurdzinski
Tony Tan
Toshimasa Ishii
Tsz Chiu Kwok
Udi Peled
Uri Zwick
V.S.P. Vijay Bhattiprolu
Vasilis Syrgkanis
Venkata Subrahmanyam
Veronika Loitzenbauer
Véronique Terrier
Vesa Halava

Vida Dujmovic
Vikraman Arvind
Ville Salo
Vincent Cohen-Addad
Vincent Penelle
Vincenzo Bonifaci
Vinod Vaikuntanathan
Vinodchandran Variyam
Virginia Va ssilevska Williams
Viswanath Nagarajan
Vittorio Bilò
Vladimir Kolmogorov
Vojtech Forejt
Vojtech Rehak
Walid Krichene
Walter Morris
William K. Moses Jr.
Wing-Kai Hon
Wojciech Rytter
Wolfgang Mulzer
Wu Hengyang
Xi Chen
Xiaohui Bei
Xiaoming Sun
Yael Kalai
Yang Cai
Yangjia Li
Yaoyun Shi
Yin Tat Lee
Yitong Yin
Yixin Cao
Yoichi Iwata
Yoshio Okamoto
Yota Otachi
Yu-Fang Chen
Yuichi Yoshida
Yun Kuen Cheung
Yury Makarychev
Yusuke Kobayashi
Yuval Filmus
Zahed Rahmati
Zdenek Dvorak
Zeev Dvir
Zhenyu Liao

List of Authors

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

Abhradeep Thakurta
Yahoo! Labs
United States
guhathakurta.abhradeep@gmail.com

Adam Kurpisz
Wroclaw University of Technology
Poland
adam.kurpisz@pwr.wroc.pl

Afshin Nikzad
Stanford University
United States
nikzad@stanford.edu

Akanksha Agrawal
University of Bergen
Norway
akanksha.agrawal@uib.no

Alan Frieze
Carnegie Mellon University
United States
alan@random.math.cmu.edu

Aleksander Madry
Massachusetts Institute of Technology
United States
madry@mit.edu

Alessio Conte
University of Pisa
Italy
ale.conte89@gmail.com

Alexandr Andoni
Columbia University
United States
andoni@mit.edu

Alina Ene
University of Warwick
United Kingdom
aene@cs.princeton.edu

Allan Grønlund
MADALGO, Department of Computer Science, Aarhus University
Denmark
jallan@cs.au.dk

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

Amer Mouawad
University of Bergen
Norway
a.mouawad@uib.no

Amey Bhangale
Rutgers University
United States
ameyrbh@gmail.com

Amir Yehudayoff
Technion-IIT
Israel
amir.yehudayoff@gmail.com

Amit Sahai
UCLA
United States
sahai@cs.ucla.edu

Amos Fiat
Tel-Aviv university
Israel
fiat@tau.ac.il

Anaël Grandjean
Université Montpellier
France
anael.grandjean@lirmm.fr

Anastasios Sidiropoulos
The Ohio State University
United States
sidiropo@gmail.com

Anca Muscholl
LaBRI, Universite Bordeaux
France
anca@labri.fr

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

András Pongrácz
University of Debrecen
Hungary
andras.pong@gmail.com

André Nichterlein
TU Berlin
Germany
andre.nichterlein@tu-berlin.de

Andrea Lincoln
Stanford
United States
andrealincoln42@gmail.com

Andrea Marino
Università di Pisa
Italy
marino@di.unipi.it

Andreas Emil Feldmann
Charles University in Prague
Czech Republic
andreas.feldmann@uwaterloo.ca

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

Andreas Göbel
University of Oxford
United Kingdom
Andreas.Goebel@cs.ox.ac.uk

Andrei Gurtov
Aalto University
Finland
gurtov@hiit.fi

Andrew Childs
University of Maryland
United States
amchilds@umd.edu

Anke van Zuylen
College of William and Mary
United States
anke@wm.edu

Anna Karlin
University of Washington
United States
karlin@cs.washington.edu

Anne-Marie Kermarrec
INRIA Rennes
France
anne-marie.kermarrec@inria.fr

Antonio Restivo
Università di Palermo
Italy
antonio.restivo@unipa.it

Anupam Gupta-Speaker
Carnegie Mellon University
United States
anupamg@cs.cmu.edu

Arnaud Sangnier
LIAFA, Univ Paris Diderot, Sorbonne Paris Cité, CNRS, France
France
sangnier@liafa.univ-paris-diderot.fr

Artur Czumaj
University of Warwick
United Kingdom
A.Czumaj@warwick.ac.uk

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

Arturs Backurs
Massachusetts Institute of Technology
Latvia
backurs@mit.edu

Assaf Naor
Princeton University
United States
naor@math.princeton.edu

Avishay Tal
Institute for Advanced Study
United States
avishay.tal@gmail.com

Aviv Adler
MIT
United States
adlera@mit.edu

B Srivathsan
Chennai Mathematical Institute
India
sri@cmi.ac.in

Barnaby Martin
Engineering and Computing Sciences,
Durham University
United Kingdom
barnabymartin@gmail.com

Bart M. P. Jansen
Technical University Eindhoven
Netherlands
bmpjansen@gmail.com

Benjamin Doerr
LIX, École Polytechnique
France
doerr@lix.polytechnique.fr

Benjamin Gras
École Normale Superieure de Lyon
France
benjamin.gras@ens-lyon.fr

Benjamin Grégoire
Inria
France
benjamin.gregoire@inria.fr

Benoit Libert
ENS Lyon, LIP Laboratory
France
benoit.libert@ens-lyon.fr

Bernadette Charron-Bost
CNRS, Ecole polytechnique
France
charron@lix.polytechnique.fr

Bill Fefferman
University of Maryland
United States
wjf@umd.edu

Brendan Juba
Washington University in St. Louis
United States
bjuba@alum.mit.edu

Bundit Laekhanukit
The Weizmann Institute of Science
Israel
bundit.laekhanukit@weizmann.ac.il

Calvin Newport
Georgetown University
United States
cnewport@cs.georgetown.edu

Casper Petersen
Department of Computer Science, University
of Copenhagen
Denmark
cazz@di.ku.dk

Cedric Yen-Yu Lin
University of Maryland
United States
cedricl@umiacs.umd.edu

Chaitanya Swamy
University of Waterloo
Canada
cswamy@uwaterloo.ca

Chandan Saha
Indian Institute of Science
India
chandan@csa.iisc.ernet.in

Chandra Chekuri
University of Illinois at Urbana-Champaign
United States
chekuri@cs.illinois.edu

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

Christel Baier
Technische Universität Dresden
Germany
christel.baier@tu-dresden.de

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

Christian Sommer
Apple Inc
United States
csom@csail.mit.edu

Christian Wulff-Nilsen
Department of Computer Science, University of Copenhagen
Denmark
koolooz@di.ku.dk

Christof Löding
Rheinisch-Westfälische Technische Hochschule Aachen
Germany
loeding@informatik.rwth-aachen.de

Christoph Berkholz
Humboldt-Universität zu Berlin
Germany
berkholz@informatik.hu-berlin.de

Christoph Haase
Laboratoire Spécification et Vérification (LSV), CNRS & ENS de Cachan
France
haase@lsv.ens-cachan.fr

Christoph Koch
TU Graz
Austria
ckoch@math.tugraz.at

Christos Tzamos
Massachusetts Institute of Technology
Greece
tzamos@mit.edu

Claire Mathieu
CNRS, Ecole Normale Superieure
France
clairemmathieu@gmail.com

Clifford Stein
Columbia University
United States
cliff@ieor.columbia.edu

Colin Cooper
King's College London
United Kingdom
colin.cooper@kcl.ac.uk

Colin White
Carnegie Mellon University
United States
crwhite@cs.cmu.edu

Constantinos Daskalakis
MIT
United States
costis@csail.mit.edu

Dakshita Khurana
UCLA
United States
dakshkhurana@gmail.com

Damien Stehle
ENS de Lyon
France
damien.stehle@ens-lyon.fr

Daniel Gburek
Technische Universität Dresden
Germany
daniel.gburek@tu-dresden.de

Daniel Graça
DM/FCT, Universidade do Algarve & SQIG - Instituto de Telecomunicações, Portugal
Portugal
dgraca@ualg.pt

Daniel Lokshtanov
University of Bergen, Norway
Norway
daniello@uib.no

Dániel Marx
Institute for Computer Science and Control, Hungarian Academy of Sciences
Hungary
dmarx@cs.bme.hu

Daniel Reichman
University of California - Berkeley
United States
daniel.reichman@gmail.com

Daniel Stan
LSV - ENS Cachan - CNRS
France
dstan@lsv.ens-cachan.fr

Daniel Štefankovič
University of Rochester
United States
stefanko@cs.rochester.edu

Daniela Petrisan
Université Paris Diderot - Paris 7
France
daniela.petrisan@gmail.com

Danny Hermelin
Ben-Gurion University of the Negev
Israel
hermelin@bgu.ac.il

David Adjiashvili
ETZ Zurich
Switzerland
addavid@ethz.ch

David P. Woodruff
IBM Almaden
United States
dpwoodru@us.ibm.com

David Richerby
University of Oxford
United Kingdom
david.richerby@cs.ox.ac.uk

Debajyoti Mondal
Department of Computer Science, University of Manitoba
Canada
jyoti@cs.umanitoba.ca

Deeparnab Chakrabarty
Microsoft Research
India
deeparnab@gmail.com

Dennis Michaels
TU Dortmund University
Germany
dennis.michaels@math.tu-dortmund.de

Di Wang
UC Berkeley
United States
wangd@eecs.berkeley.edu

Diego Figueira
CNRS
France
dfigueir@labri.fr

Dimitris Fotakis
National Technical University of Athens
Greece
fotakis@cs.ntua.gr

Diptapriyo Majumdar
Institute of Mathematical Sciences, Chennai
India
diptapriyom@imsc.res.in

Dmitry Chistikov
Max Planck Institute for Software Systems (MPI-SWS)
Germany
dch@mpi-sws.org

Elena Grigorescu
Purdue University
United States
elena-g@purdue.edu

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

Ely Porat
Bar-Ilan University
Israel
porately@cs.biu.ac.il

Emmanuel Filiot
Université Libre de Bruxelles
Belgium
efiliot@gmail.com

Erik Demaine
MIT
United States
edemaine@mit.edu

Esben Bistrup Halvorsen
University of Copenhagen
Denmark
esben@bistruphalvorsen.dk

Facundo Memoli
The Ohio State University
United States
memoli@math.osu.edu

Félix Baschenis
LaBRI
France
felix.baschenis@labri.fr

Frans Schalekamp
Cornell University
United States
fms9@cornell.edu

Frederik Mallmann-Trenn
École normale supérieure
Canada
fmallman@sfu.ca

Gabriele Fici
Università di Palermo
Italy
gabriele.fici@unipa.it

Gabriele Puppis
LaBRI, Bordeaux
France
gabriele.puppis@gmail.com

Gagan Goel
Google Research
United States
gagan.goel@gmail.com

Georg Zetzsche
LSV, CNRS & ENS Cachan, Université Paris-Saclay
France
zetzsche@cs.uni-kl.de

George Giakkoupis
INRIA Rennes
France
george.giakkoupis@inria.fr

Gerhard J. Woeginger
Technical University Eindhoven
Netherlands
gwoegi@win.tue.nl

Gil Zussman
Columbia University
United States
gil@ee.columbia.edu

Gilles Barthe
IMDEA Software Institute
Spain
gjbarthe@gmail.com

Giuseppe F. Italiano
University of Rome "Tor Vergata"
Italy
italiano@disp.uniroma2.it

Gökalp Demirci
University of Chicago
United States
demirci@cs.uchicago.edu

Gramoz Goranci
University of Vienna
Austria
gramoz.goranci@univie.ac.at

Guru Guruganesh
Carnegie Mellon University
United States
ggurugan@cs.cmu.edu

Guy Kortsarz
Rutgers University
United States
guyk@camden.rutgers.edu

Hamid Mahini
University of Maryland
United States
hamid.mahini@gmail.com

Hans L. Bodlaender
Utrecht University
Netherlands
H.L.Bodlaender@uu.nl

Harald Racke
Technische Universitat Munchen
Germany
raecke@in.tum.de

Harumichi Nishimura
Nagoya University
Japan
hnishimura@is.nagoya-u.ac.jp

Hellis Tamm
Institute of Cybernetics, Tallinn University of Technology
Estonia
hellis@cs.ioc.ee

Heng Guo
Queen Mary, University of London
United Kingdom
hguo@cs.wisc.edu

Henning Fernau
Univ. Trier
Germany
fernau@uni-trier.de

Henry Yuen
MIT
United States
hyuen@csail.mit.edu

Hirotada Kobayashi
National Institute of Informatics
Japan
hirotada@nii.ac.jp

Hubie Chen
Universidad del País Vasco and Ikerbasque
Spain
hubiechen@gmail.com

Huck Bennett
New York University
United States
hbennett@cs.nyu.edu

Igor Shinkar
New York University
United States
ishinkar@cims.nyu.edu

Igor Shparlinski
University of New South Wales
Australia
igor.shparlinski@unsw.edu.au

Igor Walukiewicz
CNRS, LaBRI
France
igw@labri.fr

Ilario Bonacina
KTH Royal Institute of Technology
Sweden
ilario@kth.se

Ilya Mironov
Google
United States
mironov@google.com

Ilya Nikolaevskiy
Aalto University
Finland
ilya.nikolaevskiy@aalto.fi

Ines Marusic
University of Oxford
United Kingdom
ines.marusic@cs.ox.ac.uk

Inge Li Gørtz
Technical University of Copenhagen
Denmark
inge@dtu.dk

Ioannis Panageas
Georgia Institute of Technology
United States
panageasj@gmail.com

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

Itai Arad
Hebrew University
Israel
arad.itai@fastmail.com

Ivona Bezáková
Rochester Institute of Technology
United States
ib@cs.rit.edu

Jaikumar Radhakrishnan
Tata Institute of Fundamental Research
India
jaikumar@tifr.res.in

Jakob Grue-Simonsen
Department of Computer Science, University of Copenhagen
Denmark
simonsen@di.ku.dk

Jakob Nordstrom
KTH Royal Institute of Technology
Sweden
jakobn@kth.se

Jalaj Upadhyay
Pennsylvania State University
United States
jalaj@psu.edu

James Worrell
University of Oxford
United Kingdom
James.Worrell@cs.ox.ac.uk

James Worrell
Oxford University
United Kingdom
jbw@cs.ox.ac.uk

Janardhan Kulkarni
Microsoft Research
United States
janardhan.kulkarni@gmail.com

Jaroslaw Blasiok
Harvard
United States
jblasiok@g.harvard.edu

Jean Goubault-Larrecq
LSV, ENS Cachan & CNRS & INRIA, Université Paris-Saclay
France
goubault@lsv.ens-cachan.fr

Jean-Francois Raskin
Universite Libre de Bruxelles
Belgium
jraskin@ulb.ac.be

Jeff Edmonds
York University
Canada
jeff@cs.york.ca

Jelani Nelson
Harvard, School of Engineering and Applied Sciences
United States
minilek@seas.harvard.edu

Jelena Marasevic
Columbia University
United States
jelena@ee.columbia.edu

Jesper Nederlof
Eindhoven University of Technology
Netherlands
j.nederlof@tue.nl

Jesper Sindahl Nielsen
Aarhus University
Denmark
jasn@cs.au.dk

Jesper W. Mikkelsen
University of Southern Denmark
Denmark
jesperwm@imada.sdu.dk

Jieming Mao
Princeton University
United States
jiemingm@cs.princeton.edu

Jinshan Zhang
The University of Liverpool
United Kingdom
jinshan.zhang@liverpool.ac.uk

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

Johannes Lengler
ETH Zürich
Switzerland
johannes.lengler@inf.ethz.ch

John Lapinskas
University of Oxford
United Kingdom
john.lapinskas@cs.ox.ac.uk

Jon Schneider
Princeton University
United States
js44@cs.princeton.edu

Jonah Brown-Cohen
UC Berkeley
United States
jonahbc@eecs.berkeley.edu

Jose Verschae
Pontificia Universidad Católica de Chile
Chile
jverschae@uc.cl

Joseph Naor
Technion
Israel
naor@cs.technion.ac.il

Joshua Grochow
Santa Fe Institute
United States
jgrochow@santafe.edu

Joshua Wang
Stanford
United States
jrwang@stanford.edu

Juho Hirvonen
Aalto University
Finland
juho.hirvonen@aalto.fi

Jun Wan
Tsinghua University
China
wanj12@mails.tsinghua.edu.cn

Justin Hsu
University of Pennsylvania
United States
justhsu@cis.upenn.edu

Justin Thaler
Yahoo Labs
United States
jthaler@fas.harvard.edu

Kamesh Munagala
Duke University
United States
kamesh@cs.duke.edu

Karl Wimmer
Duquesne University
United States
wimmerk@duq.edu

Kashyap Dixit
The Pennsylvania State University
United States
dixit.kashyap@gmail.com

Kasper Green Larsen
MADALGO, Department of Computer Science, Aarhus University
Denmark
larsen@cs.au.dk

Katrin Casel
Universitaet Trier
Germany
Casel@uni-trier.de

Kazuyuki Asada
University of Tokyo
Japan
asada@kb.is.s.u-tokyo.ac.jp

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

Keisuke Fujii
Kyoto University
Japan
fujii.keisuke.2s@kyoto-u.ac.jp

Ketan Mulmuley
The University of Chicago
United States
mulmuley@uchicago.edu

Kevin Buchin
Technical University Eindhoven
Netherlands
k.a.buchin@tue.nl

Kim-Manuel Klein
University of Kiel
Germany
kmk@informatik.uni-kiel.de

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

Kohei Kishida
University of Oxford
United Kingdom
kishidakohei@gmail.com

Krishnendu Chatterjee
Institute of Science and Technology (IST)
Austria
krish.chat@gmail.com

Kyriakos Axiotis
National Technical University of Athens
Greece
kaxiotis@corelab.ntua.gr

Laurent Doyen
LSV, ENS Cachan & CNRS
France
doyen@lsv.ens-cachan.fr

Laurent Feuilloley
University Paris Diderot
France
laurent.feuilloley@liafa.univ-paris-diderot.fr

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

Liad Blumrosen
The Hebrew University
Israel
blumrosen@gmail.com

Liang Li
Microsoft Research
China
liangl@microsoft.com

Loukas Georgiadis
University of Ioannina
Greece
loukas@gmail.com

Luca Quadro Zamboni
Université Claude Bernard Lyon 1
France
zamboni@math.univ-lyon1.fr

Luca Reggio
Université Paris Diderot - Paris 7
France
luca.reggio@liafa.univ-paris-diderot.fr

Luca Versari
Scuola Normale Superiore
Italy
veluca93@gmail.com

M. Praveen
Chennai Mathematical Institute
India
praveenm@cmi.ac.in

Mahdi Cheraghchi
Imperial College London
United Kingdom
m.cheraghchi@imperial.ac.uk

Mahsa Shirmohammadi
University of Oxford
United Kingdom
mahsa.shirmohammadi@gmail.com

Mai Gehrke
Université Paris Diderot - Paris 7
France
mgehrke@liafa.univ-paris-diderot.fr

Manfred Kufleitner
University of Stuttgart, Institut für Formale Methoden der Informatik
Germany
kufleitner@fmi.uni-stuttgart.de

Manoj Prabhakaran
University of Illinois Urbana-Champaign
United States
manojmp@gmail.com

Manuel Bodirsky
TU Dresden
Germany
Manuel.Bodirsky@tu-dresden.de

Manuel Silva
Universidade Nova de Lisboa
Portugal
mnasilva@gmail.com

Marcin Pilipczuk
Institute of Informatics, University of Warsaw
Poland
malcin@mimuw.edu.pl

Marco Chiesa
Universite Catholique de Louvain
Belgium
marco.chiesa@uclouvain.be

Marco Gaboardi
University at Buffalo, The State University of New York (SUNY)
United States
gaboardi@buffalo.edu

Maria Bruna
University of Oxford
United Kingdom
bruna@maths.ox.ac.uk

Maria-Florina Balcan
Carnegie Mellon University
United States
ninamf@cs.cmu.edu

Mark Braverman
Princeton University
United States
mbraverm@cs.princeton.edu

Mark Bun
Harvard University
United States
mbun@seas.harvard.edu

Mark de Berg
Technical University Eindhoven
Netherlands
m.t.d.berg@tue.nl

Mark Jerrum
Queen Mary, University of London
United Kingdom
m.jerrum@qmul.ac.uk

Markus L. Schmid
Universitaet Trier
Germany
MSchmid@uni-trier.de

Martin Dyer
University of Leeds
United Kingdom
M.E.Dyer@leeds.ac.uk

Martin Grohe
RWTH Aachen
Germany
grohe@informatik.rwth-aachen.de

Marvin Künnemann
Max Planck Institute for Informatics
Germany
marvin@mpi-inf.mpg.de

Mathieu Hoyrup
LORIA
France
mathieu.hoyrup@loria.fr

Mathieu Sablik
I2M
France
mathieu.sablik@univ-amu.fr

Matthias Függer
Max-Planck-Institut für Informatik
Germany
mfuegger@mpi-inf.mpg.de

Meena Boppana
Harvard University
United States
boppana@college.harvard.edu

Meiram Murzabulatov
Pennsylvania State University
United States
mzm269@psu.edu

Melanie Schmidt
University of Bonn
Germany
m.s-mail@web.de

Michael B. Cohen
MIT
United States
micohen@mit.edu

Michael Mahoney
UC Berkeley
United States
mmahoney@stat.berkeley.edu

Michael Mitzenmacher
Harvard University
United States
michaelm@eecs.harvard.edu

Michael Pinsker
Department of Algebra, Charles University
Czech Republic
marula@gmx.at

Michael Schapira
Hebrew University of Jerusalem
Israel
schapiram@cs.huji.ac.il

Michael Schraudner
CMM
Chile

Michał Skrzypczak
University of Warsaw
Poland
mskrzypczak@mimuw.edu.pl

Mickael Randour
Université Libre de Bruxelles (U.L.B.)
Belgium
mickael.randour@gmail.com

Mika Göös
University of Toronto
Canada
mika.goos@mail.utoronto.ca

Miklos Santha
IRIF, Univ. Paris 7, CNRS
France
miklos.santha@gmail.com

Mikolaj Bojanczyk
Warsaw University
Poland
bojan@mimuw.edu.pl

Mohammadtaghi Hajiaghayi
University of Maryland
United States
hajiagha@cs.umd.edu

Mohsen Ghaffari
MIT
United States
ghaffari@csail.mit.edu

Monaldo Mastrolilli
IDSIA
Switzerland
monaldo@idsia.ch

Monika Henzinger
University of Vienna
Austria
monika.henzinger@univie.ac.at

Moshe Babaioff
Microsoft Research
Israel
moshe@microsoft.com

Moti Yung
Google Inc. and Columbia University
United States
moti@cs.columbia.edu

Myrto Arapinis
University of Birmingham
United Kingdom
marapini@inf.ed.ac.uk

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

Nathalie Bertrand
Inria
France
nathalie.bertrand@inria.fr

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

Neeraj Kayal
Microsoft Research India
India
neeraka@microsoft.com

Nicolas Markey
LSV, CNRS & ENS Cachan
France
markey@lsv.fr

Nicolás Rivera
King's College London
United Kingdom
nicolas.rivera@kcl.ac.uk

Nika Haghtalab
Carnegie Mellon University
United States
nhaghtal@cs.cmu.edu

Nikos Parotsidis
University of Rome "Tor Vergata"
Italy
nikos.parotsidis@uniroma2.it

Ning Xie
Florida International University
United States
nxie@cs.fiu.edu

Nishanth Chandran
Microsoft Research
India
nichandr@microsoft.com

Nishanth Dikkala
Massachusetts Institute of Technology
India
nishanthd@csail.mit.edu

Nisheeth Vishnoi
EPFL
Switzerland
nisheeth.vishnoi@gmail.com

Nithin Mahendra Varma
The Pennsylvania Sate University
India
nithvarma@gmail.com

Noa Elad
Technion
Israel
noako@cs.technion.ac.il

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

Noy Rotbart
Department of Computer Science, University of Copenhagen
Denmark
noyro@di.ku.dk

Ofer Neiman
Ben-Gurion University
Israel
neimano@cs.bgu.ac.il

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

Olivier Gauwin
LaBRI, University of Bordeaux
France
olivier.gauwin@labri.fr

Omkant Pandey
Drexel University
United States
omkant@drexel.edu

Omkant Pandey
University of California, Berkeley
United States
omkant@gmail.com

Pan Peng
Department of Computer Science, TU Dortmund, Germany
Germany
pan.peng@tu-dortmund.de

Patricia Bouyer
LSV, CNRS & ENS Cachan, Université Paris Saclay
France
bouyer@lsv.fr

Patrick Totzke
DIMAP, Department of Computer Science, University of Warwick
United Kingdom
p.totzke@warwick.ac.uk

Pavel Hrubes
Institute of Mathematics of CAS
Czech Republic
hrubes@math.cas.cz

Peng Zhang
Georgia Institute of Technology
United States
pzhang60@gatech.edu

Peter Davies
University of Warwick
United Kingdom
P.W.Davies@warwick.ac.uk

Peter Kling
Simon Fraser University
Canada
pkling@sfu.ca

Petra Berenbrink
SFU
Canada
petra@cs.sfu.ca

Peyman Afshani
Aarhus University
Denmark
peyman@cs.au.dk

Pierre Carlier
Umons/ENS Cachan/LSV
Belgium
pierre.carlier@umons.ac.be

Pierre Fraigniaud
University Paris Diderot
France
pierre.fraigniaud@liafa.univ-paris-diderot.fr

Pierre-Yves Strub
IMDEA Software Institute
Spain
pierre-yves@strub.nu

Piotr Berman
Pennsylvania State University
United States
berman@cse.psu.edu

Piotr Krysta
The University of Liverpool
United Kingdom
p.krysta@liverpool.ac.uk

Prachi Goyal
Microsoft Research
India
t-prgoya@microsoft.com

Prahladh Harsha
Tata Institute of Fundamental Research
India
prahladh@gmail.com

Prasad Raghavendra
UC Berkeley
United States
prasad@eecs.berkeley.edu

Pratyay Mukherjee
University of California, Berkeley
United States
pratyay85@gmail.com

Radu Curticapean
Simons Institute for the Theory of Computing
United States
radu.curticapean@gmail.com

Radu Grigore
University of Kent
United Kingdom
radugrigore@gmail.com

Rafail Ostrovsky
UCLA
United States
rafail@cs.ucla.edu

Raffaella Gentilini
University of Perugia
Italy
gentilini.raffaella@gmail.com

Rahul Jain
Centre for Quantum Technologies and Department of Computer Science, National University of Singapore
Singapore
rahul@comp.nus.edu.sg

Rajiv Gandhi
Rutgers University
United States
rajivg@camden.rutgers.edu

Ran Gelles
Princeton University
United States
rgelles@cs.princeton.edu

Rani Hod
Harvard University
United States
rani.hod@gmail.com

Ranko Lazic
DIMAP, Department of Computer Science,
University of Warwick
United Kingdom
r.s.lazic@warwick.ac.uk

Raphael Clifford
University of Bristol
United Kingdom
Raphael.Clifford@bristol.ac.uk

Ravishankar Krishnaswamy
Microsoft Research
India
rakri@microsoft.com

Renato Paes-Leme
Google Research
United States
renatoppl@google.com

Richard Lipton
GeorgiaTech
United States
rjl@cc.gatech.edu

Rob van Stee
University of Leicester
United Kingdom
rob.vanstee@leicester.ac.uk

Roberto Grossi
Universita' di Pisa
Italy
grossi.roberto@gmail.com

Robin Kothari
Center for Theoretical Physics,
Massachusetts Institute of Technology
United States
rkothari@mit.edu

Rodica Condurache
Universite Paris Est, Universite Libre de
Bruxelles
France
rodica.bozianu@gmail.com

Rohit Khandekar
KCG holdings Inc., USA
United States
rkhandekar@gmail.com

Rolf Niedermeier
TU Berlin
Germany
rolf.niedermeier@tu-berlin.de

Rotem Zach
Tel Aviv University
Israel
rotemz@gmail.com

Ryan Williams
Stanford University
United States
rrwilliams@gmail.com

Saeed Seddighin
University of Maryland
United States
saeedreza.seddighin@gmail.com

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

Samuel Hetterich
Goethe University Frankfurt
Germany
hetterich@math.uni-frankfurt.de

Samuli Leppänen
IDSIA
Switzerland
samuli@idsia.ch

Sandy Heydrich
Max Planck Institute for Informatics
Germany
heydrich@mpi-inf.mpg.de

Sara Ahmadian
University of Waterloo
Canada
sahmadian@uwaterloo.ca

Sarah Winter
Rheinisch-Westfälische Technische
Hochschule Aachen
Germany
winter@automata.rwth-aachen.de

Sascha Klüppelholz
Technische Universität Dresden
Germany
sascha.klueppelholz@tu-dresden.de

Satish Rao
University of California, Berkeley
United States
satishr@cs.berkeley.edu

Satyen Kale
Yahoo Research
United States
satyen@yahoo-inc.com

Scott Shenker
International Computer Science Institute
United States
shenker@icsi.berkeley.edu

Sébastien Tavenas
Microsoft Research India
India
sebastien.tavenas@ens-lyon.org

Seiichiro Tani
NTT Communication Science Labs.
Japan
tani.seiichiro@lab.ntt.co.jp

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

Shalev Ben-David
Massachusetts Institute of Technology
United States
shalev@mit.edu

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

Shi Bai
ENS de Lyon
France
shi.bai@ens-lyon.fr

Shi Li
University at Buffalo
United States
shil@buffalo.edu

Shih-Han Hung
University of Maryland
United States
shung@umd.edu

Shuhei Tamate
National Institute of Informatics
Japan
tamate.sh@gmail.com

Sina Dehghani
University of Maryland
United States
sina.dehghani@gmail.com

Slobodan Mitrovic
École Polytechnique Fédérale de Lausanne
Switzerland
slobodan.mitrovic@epfl.ch

Sofya Raskhodnikova
The Pennsylvania State University
United States
sofya@cse.psu.edu

Soheil Ehsani
University of Maryland
United States
soheilehsani@gmail.com

Somindu C. Ramanna
ENS Lyon, LIP Laboratory
France
somindu.ramanna@ens-lyon.fr

Søren Dahlgaard
University of Copenhagen
Denmark
soren.dahlgaard@gmail.com

Stacey Jeffery
Institute for Quantum Information and
Matter, Caltech
Canada
sjeffery@caltech.edu

Stefan Göller
LSV, CNRS & ENS Cachan, Université
Paris-Saclay
France
goeller@lsv.ens-cachan.fr

Stefan Kiefer
University of Oxford
United Kingdom
stefan.kiefer@cs.ox.ac.uk

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

Stephane Durocher
University of Manitoba
Canada
durocher@cs.umanitoba.ca

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

Stephen Cook
University of Toronto
Canada
sacook@cs.toronto.edu

Sungjin Im
University of California, Merced
United States
sjin.im@gmail.com

Suzanne van der Ster
Technische Universitaet Muenchen
Germany
ster@in.tum.de

Sylvain Schmitz
LSV, ENS Cachan & CNRS & INRIA,
Université Paris-Saclay
France
schmitz@lsv.ens-cachan.fr

Tatiana Starikovskaya
University of Bristol
United Kingdom
tat.starikovskaya@gmail.com

Telikepalli Kavitha
Tata Institute of Fundamental Research,
Mumbai
India
kavitha.telikepalli@gmail.com

Thomas Brihaye
Université de Mons
Belgium
thomas.brihaye@umons.ac.be

Thomas Colcombet
CNRS, IRIF, Université Paris Diderot
France
thomas.colcombet@liafa.univ-paris-diderot.fr

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

Thomas Moscibroda
Microsoft Research
Switzerland
moscitho@microsoft.com

Thomas Nowak
Université Paris-Sud
France
thomas.nowak@lri.fr

Thomas Watson
University of Toronto
Canada
thomasw@cs.toronto.edu

Thomas Wilke
Kiel University
Germany
thomas.wilke@email.uni-kiel.de

Till Fluschnik
TU Berlin
Germany
till.fluschnik@tu-berlin.de

Tobias Walter
Universität Stuttgart
Germany
tobias.walter@fmi.uni-stuttgart.de

Tom Friedetzky
Durham University
United Kingdom
tom.friedetzky@dur.ac.uk

Tom Morgan
Harvard University
United States
tdmrgn@gmail.com

Tom van der Zanden
Utrecht University
Netherlands
T.C.vanderZanden@uu.nl

Tomoyuki Morimae
Gunma University
Japan
morimae@gunma-u.ac.jp

Tomoyuki Morimae
Gunma University
Japan
morimae@gmail.com

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

Tsuyoshi Ito
NEC
United States
tsuyoshi.ito.2006@gmail.com

Uri Zwick
Tel Aviv University
Israel
zwick@tau.ac.il

Vahab Mirrokni
Google Research
United States
mirrokni@google.com

Vahid Liaghat
Stanford University
United States
vliaghat@gmail.com

Valia Mitsou
Institute for Computer Science and Control,
Hungarian Academy of Sciences
Hungary
vmitsou@sztaki.hu

Vassilis Zikas
Rensselaer Polytechnic Institute
United States
vassilis.zikas@gmail.com

Venkatesh Medabalimi
University of Toronto
Canada
venkatm@cs.toronto.edu

Ventsislav Chonev
Institute of Science and Technology Austria
Austria
ventsislav.chonev@ist.ac.at

Victor Poupet
Université Montpellier
France
victor.poupet@lirmm.fr

Vijay Sridhar
The Ohio State University
United States
sridhar.38@buckeyemail.osu.edu

Viktor Bindewald
TU Dortmund University
Germany
viktor.bindewald@math.tu-dortmund.de

Vincent Cohen-Addad
ENS
France
vincent.cohen@ens.fr

Vinod M Prabhakaran
TIFR, Mumbai
India
vinodmp@tifr.res.in

Vipul Goyal
Microsoft Research
India
vipul@microsoft.com

Virginia Vassilevska Williams
Stanford University
United States
virgito@gmail.com

Volker Diekert
Universität Stuttgart, Institut für Formale
Methoden der Informatik
Germany
diekert@fmi.uni-stuttgart.de

Weiqiang Wen
ENS de Lyon
France
weiqiang.wen@ens-lyon.fr

Wim Van Dam
UC Santa Barbara
United States
vandam@cs.ucsb.edu

Yang Yuan
Cornell University
United States
callowbird@gmail.com

Yifeng Zhang
University of Alberta
Canada
yifeng2@ualberta.ca

Yitong Yin
Nanjing University
China
yitong.yin@gmail.com

Youming Qiao
University of Technology Sydney
Australia
jimmyqiao86@gmail.com

Yu Xia
Tsinghua University
China
xiay12@mails.tsinghua.edu.cn

Yun Kuen Cheung
University of Vienna
Austria
yun.kuen.cheung@univie.ac.at

Zachary Friggstad
University of Alberta
Canada
zacharyf@ualberta.ca

Zengfeng Huang
University of New South Wales
Australia
zengfeng.huang@unsw.edu.au

Zeyuan Allen-Zhu
MIT CSAIL
United States
zeyuan@csail.mit.edu

Zhenyu Liao
Boston University
United States
zhenyul@bu.edu

Compute Choice

Devavrat Shah

MIT, Cambridge, USA
devavrat@mit.edu

Abstract

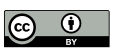
In this talk, we shall discuss the question of learning distribution over permutations of n choices based on partial observations. This is central to capturing the so called "choice" in a variety of contexts: understanding preferences of consumers over a collection of products based on purchasing and browsing data in the setting of retail and e-commerce, learning public opinion amongst a collection of socio-economic issues based on sparse polling data, and deciding a ranking of teams or players based on outcomes of games. The talk will primarily discuss the relationship between the ability to learn, nature of partial information and number of available observations. Connections to the classical theory of social choice and behavioral psychology, as well as modern literature in Statistics, learning theory and operations research will be discussed.

1998 ACM Subject Classification F. Theory of Computation

Keywords and phrases Decision Systems, Learning Distributions, Partial observations

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.1

Category Invited Talk



43rd International Colloquium on Automata, Languages,and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 1; pp. 1:1–1:1

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

Formally Verifying a Compiler: What Does It Mean, Exactly?

Xavier Leroy

INRIA, Paris, France
xavier.leroy@inria.fr

Abstract

Compilers, and especially optimizing compilers, are complicated programs. Bugs in compilers happen, and can lead to miscompilation: the production of wrong executable code from a correct source program. Miscompilation is documented in the literature and a concern for high-assurance software, as it endangers the guarantees obtained by source-level formal verification of programs.

Compiler verification is a radical solution to the miscompilation problem: by applying program proof to the compiler itself, we can obtain mathematically strong guarantees that the generated executable code is faithful to the semantics of the source program. The state of the art in this line of research is arguably the CompCert verified compiler. This talk will give an overview of this optimizing C compiler and of its formal verification, conducted with the Coq proof assistant.

A formal verification is as good as the specifications it uses. In other words, verification reduces the problem of trusting a large implementation to that of ensuring that its formal specification enforce the intended correctness properties. In the case of CompCert, the correctness statement that is proved is rather complex, as it involves large operational semantics (for the C language and for the assembly languages of the target architectures) and simulations between these semantics that support both choice refinement and behavior refinement. The talk will review and discuss these elements of the specification, along with some of the accompanying proof principles.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Compilers, Compiler Optimization, Compiler Verification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.2

Category Invited Talk

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 2; pp. 2:1–2:1

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

Hardness of Approximation

Subhash Khot

Computer Science Department, Courant Institute of Mathematical Sciences, New York University, New York, USA
khot@cims.nyu.edu

Abstract

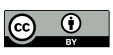
The talk will present connections between approximability of NP-complete problems, analysis, and geometry, and the role played by the Unique Games Conjecture in facilitating these connections.

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

Keywords and phrases NP-completeness, Approximation algorithms, Inapproximability, Probabilistically Checkable Proofs, Discrete Fourier analysis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.3

Category Invited Talk



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 3; pp. 3:1–3:1

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

Model Checking and Strategy Synthesis for Stochastic Games: From Theory to Practice*

Marta Z. Kwiatkowska

University of Oxford, Oxford, UK
marta.kwiatkowska@cs.ox.ac.uk

Abstract

Probabilistic model checking is an automatic procedure for establishing if a desired property holds in a probabilistic model, aimed at verifying quantitative probabilistic specifications such as the probability of a critical failure occurring or expected time to termination.

Much progress has been made in recent years in algorithms, tools and applications of probabilistic model checking, as exemplified by the probabilistic model checker PRISM (http://www.prismmodelchecker.org). However, the unstoppable rise of autonomous systems, from robotic assistants to self-driving cars, is placing greater and greater demands on quantitative modelling and verification technologies. To address the challenges of autonomy we need to consider collaborative, competitive and adversarial behaviour, which is naturally modelled using game-theoretic abstractions, enhanced with stochasticity arising from randomisation and uncertainty. This paper gives an overview of quantitative verification and strategy synthesis techniques developed for turn-based stochastic multi-player games, summarising recent advances concerning multi-objective properties and compositional strategy synthesis. The techniques have been implemented in the PRISM-games model checker built as an extension of PRISM.

1998 ACM Subject Classification F.3.1 Specifying and Verifying and Reasoning about Programs. D.2.4 Software/Program Verification

Keywords and phrases Quantitative verification, Stochastic games, Temporal logic, Model checking, Strategy synthesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.4

Category Invited Talk

1 Introduction

Probability is pervasive: it is used to quantify component failures, unreliable communication media and sensor imprecision, in randomised schemes to ensure efficiency of distributed coordination and resource management mechanisms, when dealing with environmental uncertainty, and to model performance and biological systems. Probabilistic model checking [48], also referred to as quantitative probabilistic verification, is an automated technique for verifying quantitative temporal logic properties of probabilistic models, which typically arise as extensions of Markov chains or Markov decision processes, additionally annotated with quantitative information such as time, energy or cost. The properties capture a variety of quantitative correctness, reliability or performance properties, and provide formal guarantees for properties such as "the maximum probability of the airbag failing to deploy within 0.02

* This work was partially supported by ERC Advanced Grant VERIWARE and EPSRC Mobile Autonomy Programme Grant EP/M019918/1.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 4; pp. 4:1–4:18

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

seconds is at most 10^{-10}". Probabilistic model checking involves conventional model checking techniques, for example symbolic and automata-theoretic methods, in combination with numerical or statistical analysis. For models that allow nondeterministic choices, it also enables strategy synthesis from temporal logic specifications to automate decision-making in applications such as robotics and security, where it can be employed to generate controllers or counter-attacks. This is similar to motion planning and optimal control, except that temporal logics are used to rigorously specify high-level goals, yielding controllers that are guaranteed to be correct.

Quantitative verification has been the focus of much interest in recent years, resulting in the adoption of tools such as the real-time model checker UPPAAL [3] and the probabilistic model checker PRISM [51] in several application domains. While UPPAAL is based on the theory of timed automata developed in the 1990s [2], the algorithms underlying PRISM have been known since the mid-1980s [69, 31], but have not entered the mainstream until around year 2000, enabled by symbolic techniques [7, 33, 50, 60] implemented in PRISM's first release [49]. Since then several new models and features have been added, including continuous-time Markov chains and probabilistic timed automata. PRISM has been successfully used to verify quantitative properties of a wide variety of real-life systems, automatically discovering design flaws in some of them. These include anonymity protocols [65], randomised distributed algorithms [53], wireless communication protocols [37], security [8], nanotechnology designs [59], probabilistic software [47], self-adaptive systems [15], molecular signalling pathways [45] and computational DNA circuits [57, 32].

Despite much success of probabilistic model checking, the accelerating technological advances in autonomous systems, to mention robotic assistants, self-driving cars and closed-loop medical devices, place greater and greater demands on quantitative verification technologies. Increasingly, we wish to delegate day-to-day tasks to autonomous, wirelessly connected, Internet-enabled and software-controlled devices, whose aim is to achieve certain objectives, such as safe and fuel efficient autonomous driving on a motorway. This incurs the need to consider cooperative, competitive and adversarial behaviour due to conflicting goals, e.g., safety versus fuel efficiency, while also taking into account their (possibly adversarial) interaction with environment, e.g., other cars. We also need to consider communities comprising digital and human agents, able to represent typical interactions and relationships present in scenarios such as semi-autonomous driving scenarios or collaborating with robotic assistants. Game-theoretic approaches, which explicitly represent the moves and countermoves of players and adversaries, are a natural model to represent and analyse such behaviours through the study of winning strategies. Strategy synthesis from quantitative specifications, in particular, can be viewed as constructing a strategy that is winning against all adversarial players. In these data-rich dynamic environments, stochasticity is needed not only to quantify aspects such as failure, safety and sensor uncertainty, but also to facilitate model derivation through inference from data, for example GPS sensor data.

In this paper we summarise recent progress made towards development of algorithmic techniques, tool implementation and real-life applications for turn-based stochastic multi-player games, implemented in an extension of the PRISM model checker called PRISM-games [25, 54]. Game theory is widely studied in areas such as economics and mechanism design, though is less well established as a modelling abstraction for autonomy. In the verification community, the majority of current research activity is focused on the study of algorithmic techniques and their computational complexity. In contrast, we report on ongoing effort towards the "theory to practice" transfer of quantitative verification and synthesis techniques to the challenging application domain of mobile autonomy. The overview

will cover verification and strategy synthesis from quantitative temporal logic specifications, with a focus on zero-sum, complete observation stochastic games, and will include both single- and multi-objective properties, together with a wide range of quantitative objectives (expected total reward, longrun average and ratio objectives) and compositional assume-guarantee strategy synthesis. We will also briefly describe a number of real-life examples from autonomous systems and lessons learnt through applying PRISM-games in these scenarios. We conclude the paper with a discussion of work in progress and future research in this area.

2 Stochastic Multi-Player Games

Stochastic multi-player games (SMGs) [64, 30] are a generalisation of Markov decision processes, where we distinguish between several types of nondeterministic choices, each corresponding to a different player. SMGs thus allow us to reason about strategic decisions of multiple players competing or collaborating to achieve the same objective. Several stochastic game models exist, which include concurrent games [64, 22] and partial-observation games [18, 20]. Here we focus on *turn-based* games as studied in [30], in which a single player controls the choices made in a given state. The presentation is based on [23, 24, 26, 11, 10, 9, 67], where we refer the reader for further details.

Let $\mathcal{D}(X)$ denote the family of all probability distributions over a set X.

▶ **Definition 1** (Stochastic multi-player game (SMG)). A *stochastic multi-player game* (SMG) is a tuple $\mathcal{G} = (\Pi, S, (S_\Pi, S_p), s_{\text{init}}, \Delta, \mathrm{A}, L)$, where Π is a finite set of players; S is a finite nonempty set of states partitioned into player states $S_\Pi = \bigcup S_{i\in\Pi}$ and probabilistic states S_p; $s_{\text{init}} \in S_\Pi$ is an initial state; $\Delta\colon S \times S \to [0,1]$ is a probabilistic transition function such that for all player states $s, t \in S_\Pi$ and probabilistic states $s' \in S_p$ we have $\Delta(s,t) = 0$, $\Delta(s,s') \in \{0,1\}$ and $\sum_{t\in S_\Pi} \Delta(s',t) = 1$; A is a finite nonempty set of actions; and $L : S_p \to \mathrm{A}$ is an action labelling function.

Note that each state of an SMG is controlled by a single player. The game proceeds as follows. It starts in the initial player state s_{init} and, when in a player $i \in \Pi$ state s, the player chooses the next state s' such that $\Delta(s,s') = 1$, and when in a probabilistic state $s \in S_p$ the next state is sampled according to the probability distribution $\Delta(s',\cdot)$.

Stochastic games can be equivalently defined, see e.g. [24], by partitioning the state space into player states S_Π only, and associating with each such state $s \in S_\Pi$ a set of action-distribution pairs (a,μ) called *moves*, where $a = L(s')$ for some $s' \in S_p$ such that $\Delta(s,s') = 1$ and the distribution μ, defined by $\mu(t) = \Delta(s',t)$ for all $t \in S$, gives the probability of moving to a successor state. If the sets of all players but one are empty, then a stochastic turn-based game is a probabilistic automaton in the sense of Segala [63], a mild generalisation of a Markov decision process (MDP).

We unfold an SMG $\mathcal{G}$ into *paths*, namely possibly infinite sequences $\lambda = s_0 s_1 s_2 \ldots$ such that $\Delta(s_i, s_{i+1}) > 0$ for all $i \geq 0$. Note that player states and probabilistic states alternate in a path. For a finite path $\lambda = s_0 s_1 \ldots s_k$ we use $|\lambda| = k+1$ to denote the length of the path and $\text{last}(\lambda) = x_k$ denotes its last element. A *trace* of λ is the sequence of actions that label the probabilistic states within λ. We use $\text{Path}_{\mathcal{G},s}$ to denote the set of all infinite paths originating in a state $s \in S$ and $\text{Path}_{\mathcal{G}} = \bigcup_{s\in S} \text{Path}_{\mathcal{G},s}$. The sets $\text{FPath}_{\mathcal{G},s}, \text{FPath}_{\mathcal{G}}$ of finite paths are defined analogously.

SMGs can be annotated with rewards, which allows us to formulate a variety of quantitative analyses that assign reward values to paths. We consider cumulative, longrun average and ratio rewards. The analysis typically involves ensuring that the game achieves a certain reward bound or optimising these values in the game.

▶ **Definition 2** (Reward structure)**.** Given a game $\mathcal{G} = (\Pi, S, (S_\Pi, S_p), s_{\text{init}}, \Delta, \mathrm{A}, L)$, a *reward structure* for $\mathcal{G}$ is a function $r : S \to \mathbb{R}$.

To resolve the nondeterminism in an SMG, similarly to MDPs we use strategies, except we now have a strategy for each player $i \in \Pi$. We work with an explicit memory representation of strategies due to [14].

▶ **Definition 3** (Strategy)**.** For an SMG $\mathcal{G} = (\Pi, S, (S_\Pi, S_p), s_{\text{init}}, \Delta, \mathrm{A}, L)$, a *strategy* σ_i for player i of $\mathcal{G}$ is a tuple $\sigma_i = (M, \sigma_i^u, \sigma_i^n, \sigma_i^{\mathsf{init}})$, where M is a countable set of memory elements, $\sigma_i^u \colon M \times S \to \mathcal{D}(M)$ is a memory update function, $\sigma_i^n \colon M \times S_i \to \mathcal{D}(S)$ is a next move function such that $\sigma_i^n(m, s)(s') > 0$ only if $\Delta(s, s') > 0$, and $\sigma_i^{\mathsf{init}} \colon S \to \mathcal{D}(M)$ is an initial memory element function. If the memory update function maps to point distributions, i.e. is of type $\sigma_i^u \colon M \times S \to M$, the strategy σ_i is *deterministic memory update*, and otherwise it is *stochastic memory update*. The set of all strategies for player $i \in \Pi$ is denoted by $\Sigma_{\mathcal{G}}^i$. A *strategy profile* $\sigma = \sigma_1, ...\sigma_{|\Pi|}$ comprises a strategy for every player in $\mathcal{G}$.

For a given strategy σ_i of player i, the game proceeds as follows. It starts in a player state with memory sampled according to the initial distribution function σ_i^{init}. Then, in every step of the game, player i updates the current memory element based on the current state of the game using the memory update function σ_i^u. Moreover, if the game is in a player i state, player i chooses the next state of the game using the next move action σ_i^n.

Strategies can be classified according to the use of randomisation or memory. A given strategy σ_i of player i is *deterministic* (pure) if the next move action σ_i^n is of type $\sigma_i^n \colon M \times S_i \to S$, and otherwise it is *randomised*. Similarly, σ_i is *memoryless* if M is a singleton, *finite memory* if M is finite, and *infinite memory* otherwise. An alternative, standard representation of a deterministic player i strategy is as a function from finite paths terminating in a player i state to distributions over the available moves (a, μ) in the given state. Deterministic memoryless strategies can be simply represented as functions $\sigma_i^n \colon S_i \to S$.

Stochastic memory update strategies have the same power as deterministic memory update but are exponentially more succinct than deterministic memory update [66]. Stochastic memory update strategies can be determinised if their memory is not restricted [10, 9]; in fact, a finite stochastic memory update strategy can result in a finite or infinite deterministic update strategy. Memory elements of the determinised strategies are probability distributions over memory elements, which can be interpreted as the belief that the player has about the memory element, knowing only the history and rules to update them, while the actual memory based on sampling is kept secret.

For a given strategy profile $\sigma = \sigma_1, ...\sigma_{|\Pi|}$, the behaviour of an SMG $\mathcal{G}$ is fully probabilistic and we use $\mathrm{Path}_{\mathcal{G},s}^\sigma$, $\mathrm{Path}_{\mathcal{G}}^\sigma$, $\mathrm{FPath}_{\mathcal{G},s}^\sigma$ and $\mathrm{FPath}_{\mathcal{G}}^\sigma$, respectively, to denote paths obtained under the strategy profile σ. Following standard methods [13], we can define a probability measure $\Pr_{\mathcal{G},s}^\sigma$ over the set of infinite paths $\mathrm{Path}_{\mathcal{G},s}^\sigma$. Given a random variable ρ over this probability space, the expected value of ρ is defined as $\mathbb{E}_{\mathcal{G},s}^\sigma(\rho) = \int_{\mathrm{Path}_{\mathcal{G},s}^\sigma} \rho \, \mathrm{d}\Pr_{\mathcal{G},s}^\sigma$.

We will sometimes require restrictions on SMGs. One such restriction to so called *stopping games* was introduced to avoid infinite accumulation of rewards.

▶ **Definition 4** (Stopping game)**.** A state $s_f \in S$ of a stochastic multi-player game $\mathcal{G} = (\Pi, S, (S_\Pi, S_p), s_{\text{init}}, \Delta, \mathrm{A}, L)$ is called terminal if and only if $\Delta(s_f, s_f) = 1$ and $\Delta(s_f, s') = 0$ for all $s' \neq s_f, s' \in S$. A game is called *stopping* if it has at least one terminal state and if it holds that, for every strategy profile $\sigma = \sigma_1, ...\sigma_{|\Pi|}$ and starting from the initial state, with probability 1 the game eventually stops, *i.e.,* a terminal state is reached.

3 Property Specification

We now introduce a notation for specifying temporal and reward-based properties of stochastic games. The presentation employs a variant of the probabilistic temporal logic called RPATL, based on Probabilistic Computation Tree Logic (PCTL) [12] with rewards [41], enhanced with the coalition operator from Alternating Temporal Logic (ATL) [1]. We discuss both single and multi-objective properties. The full logics RPATL and RPATL* are studied in [23, 29].

▶ **Definition 5** (RPATL property). A *single-objective*, respectively *multi-objective*, RPATL property is a formula ϕ, respectively Φ, in the following grammar:

$$\phi ::= \langle\langle C\rangle\rangle \mathtt{P}_{\bowtie p}[\psi] \mid \langle\langle C\rangle\rangle \mathtt{R}^{r}_{\bowtie x}[\rho] \mid \langle\langle C\rangle\rangle \mathtt{R}^{r/c}_{\bowtie x}[\mathtt{S}]$$

$$\Phi ::= \langle\langle C\rangle\rangle \Big(\bigwedge_{i=1}^{n} \mathtt{P}_{\bowtie p_i}[\psi_i]\Big) \mid \langle\langle C\rangle\rangle \Big(\bigwedge_{j=1}^{m} \mathtt{R}^{r_j}_{\bowtie x_j}[\rho_j]\Big) \mid \langle\langle C\rangle\rangle \Big(\bigwedge_{j=1}^{m} \mathtt{R}^{r_j/c_j}_{\bowtie x_j}[\mathtt{S}]\Big)$$

$$\psi ::= \mathtt{F}\, a$$

$$\rho ::= \mathtt{C} \mid \mathtt{S}$$

where $a \in \mathrm{AP}$ is an atomic proposition, $C \subseteq \Pi$ is a coalition of players, $\bowtie \in \{\leq, \geq\}$, $p \in [0,1]$, r, c are reward structures, and $x \in \mathbb{R}$.

The operator $\mathtt{P}_{\bowtie p}[\psi]$ is the PCTL probabilistic operator, where ψ is a CTL path formula. To simplify the presentation we only consider the reachability path formulas $\mathtt{F}\ a$. The operators $\mathtt{R}^{r}_{\bowtie x}[\mathtt{C}]$ and $\mathtt{R}^{r}_{\bowtie x}[\mathtt{S}]$ respectively denote the expected *total reward* and *longrun average reward* (mean payoff), whereas $\mathtt{R}^{r/c}_{\bowtie x}[\mathtt{S}]$ is a *longrun average ratio* reward. We combine these operators with the coalition operator $\langle\langle\cdot\rangle\rangle$ of ATL as follows. Intuitively, $\langle\langle C\rangle\rangle \mathtt{P}_{\bowtie p}[\psi]$ means that the players in coalition C can collectively ensure that $\mathtt{P}_{\bowtie p}[\psi]$ is satisfied, no matter what the players outside the coalition do. For example, in a game with players 1, 2 and 3, the property $\langle\langle\{1,3\}\rangle\rangle \mathtt{P}_{\geq 1}[\mathtt{F\ end}]$ states that players 1 and 3 have a strategy to ensure that the game reaches an `end`-state almost surely, irrespective of what player 2 does. $\langle\langle C\rangle\rangle \mathtt{R}^{r}_{\bowtie x}[\mathtt{C}]$ means that the players in coalition C can collectively ensure that the expected total reward is in relation $\bowtie$ to x. $\langle\langle C\rangle\rangle \mathtt{R}^{r}_{\bowtie x}[\mathtt{S}]$ and $\langle\langle C\rangle\rangle \mathtt{R}^{r/c}_{\bowtie x}[\mathtt{S}]$ are defined similarly, except that they respectively concern expected average rewards cumulated over infinite paths and expected ratio rewards. Both probabilistic and reward properties can be interpreted in *quantitative* fashion, e.g. $\langle\langle C\rangle\rangle \mathtt{P}_{\max=?}[\mathtt{F}\ a]$, meaning the maximum probability of reaching an a-state that players in C can ensure, regardless of the other players, and similarly for the minimum.

We also allow for *multi-objective* properties Φ defined as conjunctions of coalition properties of the same type, with the interpretation that all conjuncts are required to be satisfied simultaneously; see [26, 10] for a more general definition of multi-objective properties as Boolean combinations. Intuitively, the property $\langle\langle C\rangle\rangle(\bigwedge_{i=1}^{n} \mathtt{P}_{\bowtie p_i}[\psi_i])$ means that players in coalition C have a *collective* strategy to ensure that, for all $i = 1, \ldots, n$, we have that $\mathtt{P}_{\bowtie p_i}[\psi_i]$ holds, no matter what the other players do. $\langle\langle C\rangle\rangle(\bigwedge_{j=1}^{m} \mathtt{R}^{r_j}_{\bowtie x_j}[\rho_j])$ is defined similarly, so, for example, $\langle\langle\{4,5\}\rangle\rangle(\mathtt{R}^{\mathtt{profit}}_{\geq 5}[\mathtt{S}] \wedge \mathtt{R}^{\mathtt{fuel}}_{\leq 10}[\mathtt{S}])$ states that players 4 and 5 have a collective strategy to ensure that expected longrun average profit is at least 5 *and* expected longrun average fuel usage is at most 10, no matter what the other players do. Property $\langle\langle C\rangle\rangle(\bigwedge_{j=1}^{m} \mathtt{R}^{r_j/c_j}_{\geq x_j}[\rho_j])$ is a ratio reward, so for example $\langle\langle\{1,2\}\rangle\rangle(\mathtt{R}^{\mathtt{fuel/time}}_{\leq 10}[\mathtt{S}] \wedge \mathtt{R}^{\mathtt{visit/time}}_{\geq 20}[\mathtt{S}])$ states that players 1 and 2 have a collective strategy to ensure that expected longrun fuel consumption per time unit is at most 10 and expected longrun number of visits is at least 20, no matter what the other players do.

In order to define the semantics of the coalition and multi-objective properties, we require the notion of a coalition game based on the analogous notion for ATL.

▶ **Definition 6** (Coalition game). Given an SMG $\mathcal{G} = (\Pi, S, (S_\Pi, S_p), s_{\text{init}}, \Delta, \mathrm{A}, L)$ and coalition of players $C \subseteq \Pi$, the *coalition game* of $\mathcal{G}$ induced by C is the two-player stochastic game $\mathcal{G}_C = (\Pi, S, (S'_1, S'_2), s_{\text{init}}, \Delta, \mathrm{A}, L)$, where $S'_1 = \bigcup_{i \in C} S_i$ and $S'_2 = \bigcup_{i \in \Pi/C} S_i$. The sets of strategies of player 1 and 2 in the coalition game are respectively denoted $\Sigma^1_{\mathcal{G}_C}$ and $\Sigma^2_{\mathcal{G}_C}$.

▶ **Definition 7** (RPATL semantics). Let $\mathcal{G} = (\Pi, S, (S_\Pi, S_p), s_{\text{init}}, \Delta, \mathrm{A}, L)$ be an SMG whose states are labelled with atomic propositions $a \in AP$. We identify a proposition a with the set of states $S_a = \{s \in S \mid s \models a\}$ satisfying a, also denoted a. The satisfaction relation $\models$ is defined as follows, where formulas ϕ and Φ are interpreted over states of the game, whereas temporal formulas ψ and reward functions ρ are interpreted over paths:

$$\begin{aligned}
\mathcal{G}, \lambda \models \mathtt{F}\, a \quad &\iff \quad \exists i \geq 0 : (\lambda_i \lambda_{i+1} \ldots \models a) \\
\mathcal{G}, s \models \langle\!\langle C \rangle\!\rangle \mathtt{P}_{\bowtie p}[\psi] \quad &\Leftrightarrow \quad \exists\, \sigma_1 \in \Sigma^1_{\mathcal{G}_C} \text{ s.t. } \forall\, \sigma_2 \in \Sigma^2_{\mathcal{G}_C}. \\
&\qquad \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}\{\lambda \in \mathrm{Path}_{\mathcal{G}_C,s} \mid \mathcal{G}_C, \lambda \models \psi\} \bowtie p \\
\mathcal{G}, s \models \langle\!\langle C \rangle\!\rangle \mathtt{R}^r_{\bowtie x} \rho \quad &\Leftrightarrow \quad \exists\, \sigma_1 \in \Sigma^1_{\mathcal{G}_C} \text{ s.t. } \forall\, \sigma_2 \in \Sigma^2_{\mathcal{G}_C}. \\
&\qquad \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(rew(r,\rho)) \bowtie x \\
\mathcal{G}, s \models \langle\!\langle C \rangle\!\rangle \mathtt{R}^{r/c}_{\bowtie x} \mathtt{S} \quad &\Leftrightarrow \quad \exists\, \sigma_1 \in \Sigma^1_{\mathcal{G}_C} \text{ s.t. } \forall\, \sigma_2 \in \Sigma^2_{\mathcal{G}_C}. \\
&\qquad \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(rew(r,\mathtt{S})) \bowtie x \\
\mathcal{G}, s \models \langle\!\langle C \rangle\!\rangle (\textstyle\bigwedge_{i=1}^n \mathtt{P}_{\bowtie p_i}[\psi_i]) \quad &\Leftrightarrow \quad \exists\, \sigma_1 \in \Sigma^1_{\mathcal{G}_C} \text{ s.t. } \forall\, \sigma_2 \in \Sigma^2_{\mathcal{G}_C},\, 1 \leq i \leq n. \\
&\qquad \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}\{\lambda \in \mathrm{Path}_{\mathcal{G}_C,s} \mid \mathcal{G}_C, \lambda \models \psi_i\} \bowtie p_i \\
\mathcal{G}, s \models \langle\!\langle C \rangle\!\rangle (\textstyle\bigwedge_{j=1}^m \mathtt{R}^{r_j}_{\bowtie x_j}[\rho_j]) \quad &\Leftrightarrow \quad \exists\, \sigma_1 \in \Sigma^1_{\mathcal{G}_C} \text{ s.t. } \forall\, \sigma_2 \in \Sigma^2_{\mathcal{G}_C},\, 1 \leq j \leq m. \\
&\qquad \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(rew(r,\rho_j)) \bowtie x_j \\
\mathcal{G}, s \models \langle\!\langle C \rangle\!\rangle (\textstyle\bigwedge_{j=1}^m \mathtt{R}^{r_j/c_j}_{\bowtie x_j}[\mathtt{S}]) \quad &\Leftrightarrow \quad \exists\, \sigma_1 \in \Sigma^1_{\mathcal{G}_C} \text{ s.t. } \forall\, \sigma_2 \in \Sigma^2_{\mathcal{G}_C},\, 1 \leq j \leq m. \\
&\qquad \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(rew(r/c,\mathtt{S})) \bowtie x_j
\end{aligned}$$

where $\mathcal{G}_C{=}(\Pi, S, (S'_1, S'_2), s_{\text{init}}, \Delta, \mathrm{A}, L)$ is the coalition game of $\mathcal{G}$ induced by C, r is a reward structure for $\mathcal{G}c$, c is a nonnegative reward structure for $\mathcal{G}_C$ s.t. the probability of its mean payoff under any strategy profile is at least some constant value, and

$$\begin{aligned}
rew^k(r)(\lambda) &= \textstyle\sum_{i=0}^k r(\lambda_i) \\
rew(r,\mathtt{C})(\lambda) &= \liminf_{k\to\infty} rew^k(r)(\lambda) \\
rew(r,\mathtt{S})(\lambda) &= \liminf_{k\to\infty} \tfrac{rew^k(r)(\lambda)}{k+1} \\
rew(r/c,\mathtt{S})(\lambda) &= \liminf_{k\to\infty} \tfrac{rew^k(r)(\lambda)}{1+rew^k(c)(\lambda)}
\end{aligned}$$

The properties defined above can be employed for verification as well as strategy synthesis for stochastic multi-player games.

4 Single-objective Properties

Model checking and strategy synthesis for single-objective RPATL properties in stochastic games reduces to checking the existence of, respectively finding (if it exists), a winning strategy in two-player stochastic games, and specifically coalition games. Formally, given a stochastic multi-player game $\mathcal{G}$ and a single-objective RPATL property ϕ, e.g., $\phi = \langle\!\langle C \rangle\!\rangle \mathtt{P}_{\bowtie p}[\psi]$ or

$\phi = \langle\langle C \rangle\rangle \mathtt{R}_{\bowtie x}[\rho]$, the model checking problem is to establish whether $s_{\text{init}} \models \phi$ in the coalition game $\mathcal{G}_C$. The strategy synthesis problem, on the other hand, for a stochastic game $\mathcal{G}$ and a property ϕ as above is to construct a strategy σ_1 for player 1 in the coalition game $\mathcal{G}_C$ (if it exists) that is a witness to the satisfaction $s_{\text{init}} \models \phi$.

Similarly to MDPs, deciding the verification and strategy synthesis problems involves computing the *optimal values* of path formulas ψ and reward functions ρ defined for the minimum in the coalition game $\mathcal{G}_C$ as follows:

$$\begin{aligned} \Pr^{\min}_{\mathcal{G}_C,s}(\psi) &= \inf_{\sigma_1 \in \Sigma^1_{\mathcal{G}_C}} \sup_{\sigma_2 \in \Sigma^2_{\mathcal{G}_C}} \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\psi), \\ \mathbb{E}^{\min}_{\mathcal{G}_C,s}(rew(r,\rho)) &= \inf_{\sigma_1 \in \Sigma^1_{\mathcal{G}_C}} \sup_{\sigma_2 \in \Sigma^2_{\mathcal{G}_C}} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(rew(r,\rho)), \end{aligned} \tag{1}$$

where we swap infimum and supremum to compute the maximum value. A strategy $\sigma_1 \in \Sigma^1_{\mathcal{G}_C}$ of player 1 starting from state s is called *optimal* if it achieves the optimal value, *e.g.,* $\sup_{\sigma_2 \in \Sigma^2_{\mathcal{G}_C}} \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\psi) = \Pr^{\min}_{\mathcal{G}_C,s}(\psi)$. Similarly, the strategy is called ε-*optimal*, for $\varepsilon > 0$, if it achieves a value deviating by at most ε from the optimum, *e.g.,* $\sup_{\sigma_2 \in \Sigma^2_{\mathcal{G}_C}} \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\psi) \geq \Pr^{\min}_{\mathcal{G}_C,s}(\psi) + \varepsilon$.

The optimal values and strategies can be used to solve the RPATL single-objective model checking problem in the following way. For example, to establish the verification problem for $\mathcal{G}, s$ and property $\langle\langle C \rangle\rangle \mathtt{P}_{\geq p}[\psi]$, it suffices to verify that in the coalition game $\mathcal{G}_C$ player 1 can ensure $\Pr^{\max}_{\mathcal{G}_C,s}(\psi) \geq p$. The remaining single-objective properties can be addressed in a similar fashion. To solve the strategy synthesis problem, we compute an optimal or a suitable ε-optimal strategy for the coalition, that is, for player 1. Since the games are zero-sum, for every single-objective RPATL property ϕ there exists a winning strategy for one of the players.

The problems of computing the optimal values and strategies for the variant of RPATL discussed here are in NP $\cap$ coNP [26, 9]. No polynomial-time algorithm is known, and the method used in practice to compute optimal values is a *value iteration* algorithm [30]. For probabilistic reachability properties, given the quantitative probabilistic reachability property $\langle\langle C \rangle\rangle \mathtt{P}_{\max=?}[\mathtt{F}\, a]$, the value iteration algorithm [30] computes the optimal values for player 1 states $s \in S_1$ in the coalition game as

$$\Pr^{\max}_{\mathcal{G}_C,s}(\psi) = v^*(s) = \lim_{n \to \infty} v^*_n(s), \tag{2}$$

where $v^*_n(s)$ is computed iteratively as indicated in Fig. 1, with the computation terminated when a suitable precision threshold α is reached, i.e. the maximum difference between $v^*_n(s)$ and $v^*_{n+1}(s)$, for $s \in S$, is not more than α. The computation of minimum values proceeds analogously.

It has been shown that for single-objective probabilistic reachability properties both players have optimal strategies and memoryless deterministic strategies suffice, and an optimal player 1 strategy can be constructed from the optimal values in time linear in the size of the game [4]. For more complex RPATL* properties not discussed here, where ψ is an LTL formula, pure finite-memory strategies may be required, see [4, 21, 23] and references therein. LTL properties ψ involve converting the formula to a Rabin or parity automaton, building the product of the automaton with the coalition game, and computing optimal values for a probabilistic reachability property on the product.

For single-objective cumulative reward properties $\rho = \mathtt{C}$, if a game is non-stopping then the set of states that receive infinite total reward can be computed by solving the game with

$$v_n^*(s) = \begin{cases} 1 & \text{if } s \models a \\ 0 & \text{if } s \not\models a \text{ and } n = 0, \\ \max\{v_{n-1}^*(s), v_n'(s)\} & \text{if } n > 0, \end{cases}$$

$$v_n'(s) = \begin{cases} \max_{s' \in S}\{\Delta(s,s') \cdot v_{n-1}^*(s')\} & \text{if } s \not\models a \text{ and } s \in S_1, \\ \min_{s' \in S}\{\Delta(s,s') \cdot v_{n-1}^*(s')\} & \text{if } s \not\models a \text{ and } s \in S_2, \\ \sum_{s' \in S} \Delta(s,s') \cdot v_{n-1}^*(s') & \text{if } s \not\models a \text{ and } s \in S_p. \end{cases}$$

Figure 1 Value iteration algorithm for the quantitative probabilistic reachability property $\langle\langle C \rangle\rangle \mathtt{P}_{\max=?}[\mathtt{F}\, a]$ computed on the coalition game $\mathcal{G}_C$.

respect to a parity condition [66]. After removing these states, value iteration algorithm similar to that for probabilistic reachability can be applied to compute the (bounded) optimal values for the remaining states. An optimal (memoryless deterministic) player 1 strategy for a stopping game can be constructed from the optimal values in time linear in the size of the game [4].

Longrun average reward properties $\rho = \mathtt{S}$ are more involved since average reward disregards all transient behaviour. Nevertheless, memoryless deterministic strategies still suffice for both players to win [43, 58] and an optimal strategy can be constructed by reduction to the discounted reward problem. An alternative, more practical, method, which also extends to expected and almost sure ratio rewards, was formulated for multi-objective properties [10, 9] under certain restriction on the models. The method employs stochastic memory update strategies and reduction to expected energy objectives, and is discussed in the next section.

5 Multi-objective Properties

In this section, we discuss the problem of multi-objective verification and strategy synthesis for stochastic multi-player games, previously also studied for MDPs [38, 42], where the goal is to simultaneously satisfy a certain combination of properties. Recall that we defined a multi-objective RPATL property Φ as a conjunction of properties of the same type, e.g. $\langle\langle C \rangle\rangle(\bigwedge_{i=1}^{n} \mathtt{P}_{\bowtie p_i}[\psi_i])$ or $\langle\langle C \rangle\rangle(\bigwedge_{j=1}^{m} \mathtt{R}^{r_j}_{\bowtie x_j}[\rho_j])$ (note that in [26] any positive Boolean combinations are allowed). As for single-objective properties, for a given coalition C we reduce the analysis of a multi-player stochastic game $\mathcal{G}$ to the coalition game $\mathcal{G}_C$, and thus it suffices to consider two-player stochastic games.

Let Φ_C be a multi-objective property for coalition C involving m probabilistic or reward properties and $\mathcal{G}_C$ be the induced two-player coalition game. Let $\mathbf{r} = (r_1, \ldots, r_m)$ denote the vector of reward structures and $\mathbf{r}(s) = (r_1(s), \ldots, r_m(s))$, for every $s \in S$. Similarly, $\mathbf{p} = (p_1, \ldots, p_m)$ and $\mathbf{x} = (x_1, \ldots, x_m)$ denote the vectors of probability and reward bounds. We say that the vector of bounds $(\mathbf{p}, \mathbf{x})$ for Φ_C, denoted $\Phi_C(\mathbf{p}, \mathbf{x})$, is *achievable* if and only if there exists a winning strategy for player 1 that guarantees all properties in Φ_C with bounds $\mathbf{p}, \mathbf{x}$. The optimal achievable vectors of bounds are called Pareto vectors.

▶ **Definition 8** (Pareto set)**.** Let Φ_C be a multi-objective property for coalition C involving n probabilistic or reward properties. A vector $(\mathbf{p}, \mathbf{x}) \in \mathbb{R}^m$ is called a *Pareto vector* if the property $\Phi_C(\mathbf{p} - \varepsilon, \mathbf{x} - \varepsilon)$ is achievable in $\mathcal{G}_C$ for every $\varepsilon > 0$ and $\Phi_C(\mathbf{p} + \varepsilon, \mathbf{x} + \varepsilon)$ is not achievable for any $\varepsilon > 0$. The set P of all Pareto vectors for Φ_C is called a *Pareto set*.

$$V_n^*(s) = \begin{cases} \{\mathbf{x} \in \mathbb{R}^m_{\geq 0} \mid \mathbf{x} \leq \mathbf{r}(s)\} & \text{if } n = 0, \\ \mathsf{dwc}(\mathbf{r}(s) + \mathsf{conv}(\bigcup_{\Delta(s,s')=1} V_{n-1}^*(s'))) & \text{if } n > 0 \text{ and } s \in S_1, \\ \mathsf{dwc}(\mathbf{r}(s) + \bigcap_{\Delta(s,s')=1} V_{n-1}^*(s')) & \text{if } n > 0 \text{ and } s \in S_2, \\ \mathsf{dwc}(\mathbf{r}(s) + \sum_{\Delta(s,s')>0} \Delta(s,s') \cdot V_{n-1}^*(s')) & \text{if } n > 0 \text{ and } s \in S_p, \end{cases}$$

$$x \cdot X = \{x \cdot \mathbf{x} \mid \mathbf{x} \in X\},$$

$$\mathbf{x} + X = \{\mathbf{x} + \mathbf{x}' \mid \mathbf{x}' \in X\},$$

$$\mathsf{dwc}(X) = \{\mathbf{y} \mid \exists \mathbf{x} \in X : \mathbf{y} \leq \mathbf{x}\},$$

$$\mathsf{conv}(X) = \{\mathbf{y} \mid \exists \mathbf{x}, \mathbf{x}' \in X, \alpha \in [0,1] : \mathbf{y} = \alpha\mathbf{x} + (1-\alpha)\mathbf{x}'\}.$$

Figure 2 Iterative computation of an ε-approximation of the Pareto set for a multi-objective expected total reward property in a two-player stochastic game. Here, $x \in \mathbb{R}_{\geq 0}$ is a real number, $\mathbf{x} \in \mathbb{R}^m_{\geq 0}$ is a vector, $X \subseteq \mathbb{R}^m_{\geq 0}$ is a set, $\leq$ is the componentwise partial order on $\mathbb{R}^m_{\geq 0}$, $\mathsf{dwc}(X)$ is the downward closure of the set X, and $\mathsf{conv}(X)$ is the convex closure of X. Given a two-player stopping game $\mathcal{G}$ with multiple reward structures $\mathbf{r}$ and a multi-objective total reward property $\Phi(\mathbf{x})$, the approximation is computed for every state $s \in S$ in $k = |S| + \lceil |S| \cdot \frac{\ln(\varepsilon \cdot (n \cdot M)^{-1})}{\ln(1-\delta)} \rceil$ iterations, where $M = |S| \cdot \frac{\max_{i, s \in S} r_i(s)}{\delta}$, $\delta = \Delta_{\min}^{|S|}$, and $\Delta_{\min}$ is the smallest positive probability in $\mathcal{G}$.

The problems of multi-objective verification and strategy synthesis are formulated similarly to the single-objective case discussed in the previous section. However, unlike in the single-objective case, optimal strategies might not exist, as shown in [27] for conjunctions. Further, an infinite-memory strategy may be required, even for stopping games with reachability objectives [26]. Existing solutions therefore compute ε-approximations of Pareto sets and the corresponding ε-optimal strategies.

▶ **Definition 9** (Pareto set approximation)**.** For $\varepsilon > 0$, an ε-approximation of the Pareto set is a set of vectors Q such that for every $(\mathbf{q}, \mathbf{y}) \in Q$ there exists a Pareto vector $(\mathbf{p}, \mathbf{x}) \in P$ with $\|(\mathbf{q}, \mathbf{y}) - (\mathbf{p}, \mathbf{x})\| \leq \varepsilon$, and vice versa, for every Pareto vector $(\mathbf{p}, \mathbf{x}) \in P$ there exists a vector $(\mathbf{q}, \mathbf{y}) \in Q$ with $\|(\mathbf{q}, \mathbf{y}) - (\mathbf{p}, \mathbf{x})\| \leq \varepsilon$, where $\|\cdot\|$ is the Manhattan distance defined as the sum of componentwise differences.

The ε-approximation of the Pareto set for a stopping coalition game $\mathcal{G}_C$ and a multi-objective property $\Phi_C(\mathbf{x})$ can be computed using the iteration algorithm in Fig. 2. The algorithm successively computes, for each state $s \in S$, the sets $V_n^*(s)$, where the nth such set is the downward closure of vectors of bounds achievable by the coalition (player 1), from s, in up to n steps. Since player 1 can randomise between its successor states, the set $V_n^*(s)$ for $s \in S_1$ is computed as a downward, convex closure of the union of $V_{n-1}^*(s')$, for all s' such that $\Delta(s, s') = 1$. For $s \in S_2$, the bounds must be achievable for all successor states, and hence we take the intersection. Finally, for probabilistic states $s \in S_p$, we consider the sum weighted by the corresponding probabilistic distribution.

Once an ε-approximation of the Pareto set has been computed, the corresponding ε-optimal player 1 strategy can be constructed [29, 26, 70], where the stochastic memory update, $\sigma_1 = (M, \sigma_1^u, \sigma_1^n, \sigma_1^{\mathsf{init}})$, representation is employed. In the construction, the vertices of approximation sets $V_n^*(s), s \in S$, act as memory elements M and represent the vector of reward bounds that the strategy currently aims to achieve. The distributions in functions σ_1^u and σ_1^{init} are constructed so that the expected value of the next memory element is an ε-approximation of the target reward bounds $\mathbf{x}$. Employing stochastic memory update

representation enables a reduction in the memory required from up to infinite to finite for stopping games [26].

Since probabilistic reachability properties can be reduced to total reward properties, the iterative algorithm in Fig. 2 can be adapted to compute ε-approximations of Pareto sets for any stopping stochastic game with a multi-objective property that involves only probabilistic reachability properties. This can be extended, for stopping games, to probabilistic LTL properties [29] by employing reduction to Rabin automata and building a synchronous product of all the automata and the original SMG $\mathcal{G}$, with a new terminal state which is entered after $\mathcal{G}$ enters any of its terminal states. However, the strategy synthesis problem for multi-objective probabilistic LTL properties in general stochastic games remains open.

Unfortunately, this method cannot be used for multi-objective strategy synthesis for longrun average reward properties. This is because the algorithm in Fig. 2 approximates the Pareto set in a finite number of iterations by combining the achievable values of successive states, but expected average reward disregard all transient behaviour. Nevertheless, for the special case of almost sure average reward properties [10], strategy synthesis for multi-objective properties of this type reduces to synthesis for multi-objective expected energy properties. The corresponding decision problem is in co-NP, as shown in [9] and [19]. In addition, [9] also formulate an algorithm to construct a strategy, if it exists, where stochastically updated memory strategies are generated, which can yield exponentially more compact representations than deterministically updated strategies used in [19]. Further, [9] identify a general class of games for which the synthesis algorithm can be extended to arbitrary Boolean combinations of expected mean-payoff objectives.

Finally, in [10, 9] a method to handle multi-objective ratio reward properties is provided. To solve the strategy synthesis problem for a single-objective ratio reward property in a coalition game, it suffices to solve the problem for an almost sure average reward property, and this can be extended to (conjunctive) multi-objective properties using vectors.

6 Compositional Strategy Synthesis

One difficulty with multi-objective strategy synthesis, in view of high computational complexity [26], is its scalability. To deal with this problem, [11, 70] formulate a *compositional* framework for strategy synthesis, which allows one to derive a strategy for the composed system by synthesising only for the (smaller) individual components. Firstly, recall that probabilistic automata of [63] correspond to coalition games with only one player present. In verification, the nondeterminism that is present in the probabilistic automaton models an adverse, uncontrollable, environment. By applying a coalition strategy to a game to resolve the controllable nondeterminism, we are left with a probabilistic automaton where only uncontrollable nondeterminism for the remaining players remains. This observation allows us to reuse rules for compositional verification of probabilistic automata, such as those in [56], to derive strategy synthesis rules for SMGs.

The compositional framework of [11, 9, 70] is based on a parallel composition operation for stochastic games, which is closely related to that of probabilistic automata [63], except that the identity of the player is preserved through composition. When composed, the component SMGs $\mathcal{G} = (\Pi, S, (S_\Pi, S_p), s_{\text{init}}, \Delta, \mathrm{A}, L)$ and $\mathcal{G}' = (\Pi', S', (S'_\Pi, S'_p), s'_{\text{init}}, \Delta', \mathrm{A}', L')$ synchronise on shared actions $A \cup A'$, yielding the composed game $\mathcal{G}'' = \mathcal{G} \parallel \mathcal{G}'$. Properties of the component SMGs, as well as the composed game, are then defined on traces, that is, sequences of actions that label the probabilistic states in the path, rather than over paths. Under the assumption that the component games are compatible, *i.e.,* all actions of player 1

in each composite game are enabled and fully controlled by player 1, the player 1 strategy $\sigma_1'' = \sigma_1 \parallel \sigma_1'$ for $\mathcal{G}''$ that is a composition of player 1 strategies for component games preserves all properties. More precisely, if strategies σ_1 and σ_1' guarantee (possibly multi-objective) properties Φ, Φ' in component games, then the composed strategy σ_1'' guarantees property Φ'' in $\mathcal{G}''$, where Φ'' is *any* property for the composed game that can be derived from Φ and Φ' using, for example, assume-guarantee rules in [56]. In particular, player 1 of different component games can cooperate to achieve a common goal: if in one component game player 1 guarantees a property Φ_2 under some assumption Φ_1 on the environment, *i.e.*, $\Phi_1 \Rightarrow \Phi_2$, and player 1 in a different component game ensures Φ_1, then the composition satisfies property Φ_2. A broad range of such assume-guarantee contracts can be supported for both probabilistic and reward multi-objective properties.

The method for compositional strategy synthesis [11] first computes an under-approximation Q of the Pareto set for Φ'' based on ε-approximations Q, Q' of Pareto sets for Φ, Φ'. For a chosen achievable vector of bounds $(\mathbf{p}, \mathbf{x})$ for Φ'', player 1 strategies σ_1, σ_i' are synthesised for component games that achieve $\Phi(\mathbf{p}, \mathbf{x})$, $\Phi'(\mathbf{p}', \mathbf{x}')$, where $(\mathbf{p}, \mathbf{x})$, $(\mathbf{p}', \mathbf{x}')$ are the respective bounds obtained by projecting $(\mathbf{p}, \mathbf{x})$ from Q'' to Q, Q'. The composed strategy $\sigma_1'' = \sigma_1 \parallel \sigma_1'$ then achieves $\Phi''(\mathbf{p}'', \mathbf{x}'')$. Note that, since assume-guarantee contracts may involve implication, to be able to apply this framework and, in particular, to take full advantage of assume-guarantee rules, we would need to be able to synthesise strategies for arbitrary Boolean combinations of properties, which is possible [9, 70] under certain restrictions on models and properties.

7 Tool Implementation and Applications

All techniques overviewed in this paper, including single- and multi-objective, as well as compositional, strategy synthesis problems for turn-based stochastic multi-player games, have been implemented in the open-source tool called PRISM-games [25, 54], developed as an extension of the probabilistic model checker PRISM [52]. PRISM-games can be used to model, verify, solve and simulate stochastic multi-player games with complex properties. As a modelling notation, PRISM-games uses an extension of PRISM's modelling language based on reactive modules. The specification notation is based on RPATL [23], and includes support for the coalition operator; single-objective properties, namely probabilistic reachability, total reward properties for stopping games, average reward and ratio properties for a special class of games called controllable multichain games (for details, see [70]), and almost sure average reward and ratio properties; and multi-objective properties, and specifically Boolean combinations of the same type of reward properties, except for the almost sure average and ratio reward properties for which only conjunctions are supported.

Currently, PRISM-games is an explicit state model checker, which extends the Java-based engine of PRISM, and relies on the Parma Polyhedra Library [6] for symbolic manipulation of convex sets during ε-approximate computation of Pareto sets, see [66, 70].

Below we report on a variety of case studies of autonomous systems that employed stochastic game models and were analysed using PRISM-games. For more information, see [66, 70, 68, 67] and the PRISM-games website [61].

Microgrid demand-side management [66]. The example models a decentralised energy management protocol for smart grids that draw energy from a variety of sources. The system consists of a set of households, where each household follows a simple probabilistic protocol to execute a load if the current energy cost is below a pre-agreed limit, and otherwise it only

executes the load with a pre-agreed probability. The energy cost to execute a load for a single time unit is the number of loads currently being executed in the grid. The analysis of the protocol with respect to the expected load per cost unit for a household, formulated as a single-objective total reward property, exposed a protocol weakness. The weakness was then corrected by disincentivising non-cooperative behaviour.

Human-in-the-loop UAV mission planning [40]. This case study concerns autonomous unmanned aerial vehicles (UAV) performing road network surveillance and reacting to inputs from a human operator. The UAV performs most of the piloting functions, such as selecting the waypoints and flying the route. The operator primarily performs sensor tasks at waypoints but may also pick a road for the UAV at waypoints. The optimal UAV piloting strategy depends on mission objectives, *e.g.,* safety, reachability, coverage, and operator characteristics, *i.e.,* workload, proficiency, and fatigue. The main focus of the case study is on studying a multi-objective property to analyse the trade-off between the mission completion time and the number of visits to restricted operating zones, which have been investigated by computing Pareto sets.

Autonomous urban driving [29]. A stochastic game model of an autonomous car is developed, which considers the car driving through an urban environment and reacting to hazards such as pedestrians, obstacles, and traffic jams. The car does not only decide on the reactions to hazards, which are adversarial, but also chooses the roads to take in order to reach a target location. The presence and probability of hazards is based on statistical information for the road. Through multi-objective strategy synthesis, strategies with optimal trade-off between the probability of reaching the target location, the probability of avoiding accidents and the overall quality of roads on the route are identified.

Aircraft power distribution [10]. An aircraft electrical power network is considered, where power is to be routed from generators to buses through controllable switches. The generators can exhibit failures and switches have delays. The system consists of several components, each containing buses and generators, and the components can deliver power to each other. The network is modelled as a composition of stochastic games, one for each component. These components are physically separated for reliability, and hence allow limited interaction and communication. Compositional strategy synthesis is applied to find strategies with good trade-off between uptime of buses and failure rate. By employing stochasticity, we can faithfully encode the reliability specifications in quantitative fashion, thus improving over previous results. The property is modelled as a conjunction of ratio reward properties.

Self-adaptive software architectures [44, 16]. Self-adaptive software automatically adapts its structure and behaviour according to changing requirements and quantitative goals. Several self-adaptive software architectures, such as adaptive industrial middleware used to monitor and manage sensor networks in renewable energy production plants, have been modelled as stochastic games and analysed. Both single- and multi-objective verification of multi-player stochastic games has been applied to to evaluate their resilience properties and synthesise pro-active adaptation policies.

DNS Bandwidth Amplification Attack [35]. The Domain Name System (DNS) is an Internet-wide hierarchical naming system for assigning IP addresses to domain names, and any disruption of the service can lead to serious consequences. A notable threat to DNS,

namely the bandwidth amplification attack, where an attacker attempts to flood a victim DNS server with malicious traffic, is modelled as a stochastic game. Verification and strategy synthesis is used to analyse and generate counter-measures to defend against the attack.

Attack-defence scenarios in RFID goods management system [5]. This case study considers complex attack-defence scenarios, such as an RFID goods management system, translating attack-defence trees to two-player stochastic games. Probabilistic verification is then employed to check security properties of the attack-defence scenarios and to synthesise strategies for attackers or defenders which guarantee or optimise some quantitative property. The properties considered include single-objective properties such as the probability of a successful attack or the incurred cost, as well as their multi-objective combinations.

8 Challenges

Clearly, there has been much progress towards quantitative verification and aspects of quantitative synthesis for autonomy, with a wide variety of relevant case studies serving as proof of concept. However, a number of significant challenges have yet to be overcome. We briefly review a selection of these below.

Partial information. Practical quantitative verification, as exemplified by PRISM, has so far mostly been limited to complete information systems. This restriction is not applicable to many autonomous scenarios, where agents in the system only have partial information. Partial observability [18, 20] raises a number of algorithmic challenges that need to be tackled.

Modelling social interactions. Autonomous systems are increasingly often employed to assist and interact with humans, and operator models have to be taken into account. Though some progress has been made, for example in the context of UAVs [40], models that incorporate cognitive processes and social interactions, such as those based on trust [39, 46], and the corresponding verification techniques are needed.

Model learning and adaptation from data. Quantitative verification has so far mainly focused on modelling system dynamics, but the behaviour of many autonomous and semi-autonomous systems, such as those involving perception, is data-driven [62]. Techniques that integrate model learning from data in order to inform adaptation in real-time and programming with uncertain data within quantitative verification methodologies are needed.

Model synthesis from specifications. Though correct-by-construction synthesis of strategies has been tackled in a range of models, model synthesis from quantitative specifications requires further study. A possible approach is combining template-based and parameter synthesis methods already developed for Markov chains and MDPs [34, 28, 17] and via discretisation for timed and hybrid automata [36, 55], but more effort is required to tackle autonomous systems.

Scalability, efficiency and precision. Existing model checking and strategy synthesis tools for stochastic games are in early stages of development and substantial effort is necessary to ensure their effectiveness in industrial applications. Compositional approaches, symbolic techniques and methodologies based on induction, deduction and machine learning, as well as their judicious combinations, have great potential.

9 Conclusion

As autonomous systems are becoming an integral part of our society, their failure carries potentially unacceptable and life-endangering risks. Rigorous model-based verification technologies incorporated within the design process can improve their safety and reliability and reduce development costs. This paper has briefly summarised quantitative verification and strategy synthesis techniques developed for autonomous systems modelled as turn-based stochastic multi-player games as implemented in the tool PRISM-games and outlined future research challenges in this challenging yet exciting field.

References

1 R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

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

3 Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Ansgar Fehnker, Thomas Hune, Bertrand Jeannet, Kim G. Larsen, M. Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. Uppaal – Now, Next, and Future. In F. Cassez, C. Jard, B. Rozoy, and M. Ryan, editors, *Modelling and Verification of Parallel Processes*, number 2067 in Lecture Notes in Computer Science Tutorial, pages 100–125. Springer-Verlag, 2001.

4 Daniel Andersson and Peter Bro Miltersen. The Complexity of Solving Stochastic Games on Graphs. In *Algorithms and Computation*, volume 5878 of *LNCS*, pages 112–121. 2009.

5 Zaruhi Aslanyan, Flemming Nielson, and David Parker. Quantitative Verification and Synthesis of Attack-Defence Scenarios. In *Proc. of Computer Security Foundations Symposium CSF*, 2016. to appear.

6 Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

7 C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proc. 24th Int. Colloq. Automata, Languages and Programming (ICALP'97)*, volume 1256 of *LNCS*, pages 430–440. Springer, 1997.

8 S. Basagiannis, P. Katsaros, A. Pombortsis, and N. Alexiou. Probabilistic model checking for the quantification of DoS security threats. *Computers & Security*, 2009.

9 N. Basset, M. Kwiatkowska, and C. Wiltsche. Compositional strategy synthesis for stochastic games with multiple objectives. Technical Report CS-RR-16-03, Department of Computer Science, University of Oxford, 2016.

10 Nicolas Basset, Marta Z. Kwiatkowska, Ufuk Topcu, and Clemens Wiltsche. Strategy Synthesis for Stochastic Games with Multiple Long-Run Objectives. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, pages 256–271, 2015.

11 Nicolas Basset, Marta Z. Kwiatkowska, and Clemens Wiltsche. Compositional Controller Synthesis for Stochastic Games. In *Proc. of Concurrency Theory CONCUR*, pages 173–187, 2014.

12 Andrea Bianco and Luca de Alfaro. Model checking of probabilistic and nondeterministic systems. In *Proc. of Foundations of Software Technology and Theoretical Computer Science FSTTCS*, volume 1026 of *LNCS*, pages 499–513, 1995.

13 P. Billingsley. *Probability and Measure*. Wiley, 1995.

14 Tomás Brázdil, Václav Brozek, Vojtech Forejt, and Antonín Kucera. Stochastic Games with Branching-Time Winning Objectives. In *Proc. of Logic in Computer Science LICS*, pages 349–358, 2006.

15 R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola. Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM*, 55(9):69–77, 2012.

16 Javier Cámara, Gabriel A. Moreno, and David Garlan. Stochastic Game Analysis and Latency Awareness for Proactive Self-adaptation. In *Proc. of Software Engineering for Adaptive and Self-Managing Systems SEAMS*, pages 155–164, 2014.

17 M. Ceska, F. Dannenberg, N. Paoletti, M. Kwiatkowska, and L. Brim. Precise parameter synthesis for stochastic biochemical systems. *Acta Informatica, to appear*, 2016.

18 Krishnendu Chatterjee and Laurent Doyen. Partial-Observation Stochastic Games: How to Win when Belief Fails. *Transactions on Compuational Logic*, 15(2):16, 2014.

19 Krishnendu Chatterjee and Laurent Doyen. Perfect-information Stochastic Games with Generalized Mean-Payoff Objectives. In *Proc. of LICS*. 2016. To appear.

20 Krishnendu Chatterjee, Laurent Doyen, Sumit Nain, and Moshe Y. Vardi. The Complexity of Partial-Observation Stochastic Parity Games with Finite-Memory Strategies. In *Proc. of Foundations of Software Science and Computation Structures FOSSACS*, pages 242–257, 2014.

21 Krishnendu Chatterjee and Thomas A. Henzinger. A survey of stochastic ω-regular games. *Journal of Computer and System Sciences*, 78(2):394–413, 2012.

22 Krishnendu Chatterjee and Rasmus Ibsen-Jensen. Qualitative analysis of concurrent mean-payoff games. *Information and Computation*, 242:2–24, 2015.

23 T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. Automatic Verification of Competitive Stochastic Systems. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, volume 7214 of *LNCS*, pages 315–330, 2012.

24 T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. Automatic verification of competitive stochastic systems. *Formal Methods in System Design*, 43(1):61–92, 2013.

25 T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. PRISM-games: A Model Checker for Stochastic Multi-Player Games. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, volume 7795 of *LNCS*, pages 185–191, 2013.

26 Taolue Chen, Vojtech Forejt, Marta Z. Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. On Stochastic Games with Multiple Objectives. In *Proc. of Mathematical Foundations of Computer Science MFCS*, pages 266–277, 2013.

27 Taolue Chen, Vojtěch Forejt, Marta Kwiatkowska, Aistis Simaitis, Ashutosh Trivedi, and Michael Ummels. Playing Stochastic Games Precisely. In *Proc. of Concurrency Theory CONCUR*, volume 7454 of *LNCS*, pages 348–363. 2012.

28 Taolue Chen, Ernst Moritz Hahn, Tingting Han, Marta Kwiatkowska, Hongyang Qu, and Lijun Zhang. Model repair for Markov decision processes. In *Proc. 7th Int. Symp. Theoretical Aspects of Software Engineering (TASE'13)*, pages 85–92. IEEE Computer Society Press, 2013.

29 Taolue Chen, Marta Z. Kwiatkowska, Aistis Simaitis, and Clemens Wiltsche. Synthesis for Multi-objective Stochastic Games: An Application to Autonomous Urban Driving. In *Proc. of Quantitative Evaluation of Systems QEST*, pages 322–337, 2013.

30 Anne Condon. The Complexity of Stochastic Games. *Information and Computation*, 96:203–224, 1992.

31 C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite state probabilistic programs. In *Proc. 29th Annual Symp. Foundations of Computer Science (FOCS'88)*, pages 338–345. IEEE Computer Society Press, 1988.

32 F. Dannenberg, M. Kwiatkowska, C. Thachuk, and A. J. Turberfield. Dna walker circuits: Computational potential, design and verification. *Natural Computing*, 14(2):195–211, 2015.

33 L. de Alfaro, M. Kwiatkowska, G. Norman, D. Parker, and R. Segala. Symbolic model checking of probabilistic processes using MTBDDs and the Kronecker representation. In S. Graf and M. Schwartzbach, editors, *Proc. 6th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'00)*, volume 1785 of *LNCS*, pages 395–410. Springer, 2000.

34 C. Dehnert, S. Junges, N. Jansen, F. Corzilius, M. Volk, H. Bruintjes, J-P. Katoen, and E. Ábrahám. PROPhESY: A PRObabilistic ParamEter SYnthesis tool. In *Proc. 27th Int. Conf. Computer Aided Verification (CAV'15)*, volume 9206 of *LNCS*, pages 214–231. Springer, 2015.

35 T. Deshpande, P. Katsaros, S.A. Smolka, and S.D. Stoller. Stochastic Game-Based Analysis of the DNS Bandwidth Amplification Attack Using Probabilistic Model Checking. In *Proc. of European Dependable Computing Conference EDCC*, pages 226–237, 2014.

36 M. Diciolla, C. H. P. Kim, M. Kwiatkowska, and A. Mereacre. Synthesising optimal timing delays for timed I/O automata. In *Proc. 14th International Conference on Embedded Software (EMSOFT'14)*. ACM, 2014.

37 M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker. A formal analysis of Bluetooth device discovery. *Int. Journal on Software Tools for Technology Transfer*, 8(6):621–632, 2006.

38 K. Etessami, M. Kwiatkowska, M. Vardi, and M. Yannakakis. Multi-objective model checking of Markov decision processes. *Logical Methods in Computer Science*, 4(4):1–21, 2008.

39 R. Falcone and C. Castelfranchi. Social trust: A cognitive approach. In *Trust and Deception in Virtual Societies*, pages 55–90. Kluwer, 2001.

40 Lu Feng, Clemens Wiltsche, Laura Humphrey, and Ufuk Topcu. Controller Synthesis for Autonomous Systems Interacting with Human Operators. In *Proc. of Int. Conf. on Cyber-Physical Systems ICCPS*, pages 70–79, 2015.

41 V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated Verification Techniques for Probabilistic Systems. In *Proc. of Formal Methods for Eternal Networked Software System SFM*, volume 6659 of *LNCS*, pages 53–113, 2011.

42 V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Quantitative multi-objective verification for probabilistic systems. In P. Abdulla and K. Leino, editors, *Proc. 17th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'11)*, volume 6605 of *LNCS*, pages 112–127. Springer, 2011.

43 D. Gillette. Stochastic games with zero stop probabilities. *Contributions to the Theory of Games*, 39:179–187, 1957.

44 T.J. Glazier, J. Camara, B. Schmerl, and D. Garlan. Analyzing Resilience Properties of Different Topologies of Collective Adaptive Systems. In *Proc. of Self-Adaptive and Self-Organizing Systems Workshops SASOW*, pages 55–60, 2015.

45 J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. *Theoretical Computer Science*, 319(3):239–257, 2008.

46 X. Huang and M. Kwiatkowska. Reasoning about cognitive trust in stochastic multiagent systems. Technical Report CS-RR-16-02, Department of Computer Science, University of Oxford, 2016.

47 M. Kattenbelt, M. Kwiatkowska, G. Norman, and D. Parker. Abstraction refinement for probabilistic software. In N. Jones and M. Muller-Olm, editors, *Proc. 10th Int. Conf. Verification, Model Checking, and Abstract Interpretation (VMCAI'09)*, volume 5403 of *LNCS*, pages 182–197. Springer, 2009.

48 M. Kwiatkowska. Model checking for probability and time: From theory to practice. In *Proc. 18th Annual IEEE Symp. Logic in Computer Science (LICS'03)*, pages 351–360. IEEE Computer Society Press, 2003. Invited Paper.

49 M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In T. Field, P. Harrison, J. Bradley, and U. Harder, editors, *Proc. 12th Int. Conf. Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS'02)*, volume 2324 of *LNCS*, pages 200–204. Springer, 2002.

50 M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128–142, 2004.

51 M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd Int. Conf. Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

52 M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. of Computer Aided Verification CAV*, volume 6806 of *LNCS*, pages 585–591, 2011.

53 M. Kwiatkowska, G. Norman, and R. Segala. Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. 13th Int. Conf. Computer Aided Verification (CAV'01)*, volume 2102 of *LNCS*, pages 194–206. Springer, 2001.

54 M. Kwiatkowska, D. Parker, and C. Wiltsche. PRISM-games 2.0: A Tool for Multi-Objective Strategy Synthesis for Stochastic Games. In *Proc. of Tools and Algorithms for the Construction and Analysis of Systems TACAS*, 2016. to appear.

55 Marta Kwiatkowska, Alexandru Mereacre, Nicola Paoletti, and Andrea Patanè. Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques. In *Proceedings of the 4th International Workshop on Hybrid Systems and Biology (HSB 2015)*, volume 9271 of *LNCS/LNBI*, pages 119–140. Springer, 2015.

56 Marta Kwiatkowska, Gethin Norman, David Parker, and Hongyang Qu. Compositional probabilistic verification through multi-objective model checking. *Information and Computation*, 232:38–65, 2013.

57 M. Lakin, D. Parker, L. Cardelli, M. Kwiatkowska, and A. Phillips. Design and analysis of DNA strand displacement devices using probabilistic model checking. *Journal of the Royal Society Interface*, 9(72):1470–1485, 2012.

58 Thomas M. Liggett and Steven A. Lippman. Stochastic Games with Perfect Information and Time Average Payoff. *SIAM Review*, 11(4):604–607, 1969.

59 G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla. Evaluating the reliability of NAND multiplexing with PRISM. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(10):1629–1637, 2005.

60 D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.

61 PRISM-games website. `http://www.prismmodelchecker.org/games/`.

62 Dorsa Sadigh, Katherine Driggs-Campbell, Alberto Puggelli, Wenchao Li, Victor Shia, Ruzena Bajcsy, Alberto L. Sangiovanni-Vincentelli, S. Shankar Sastry, and Sanjit A. Seshia. Data-driven probabilistic modeling and verification of human driver behavior. In *Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symposium*, 2014.

63 R. Segala. *Modelling and Verification of Randomized Distributed Real Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

64 L. S. Shapley. Stochastic games. In *National Academy of Sciences*, pages 1095–1100, 1953.

65 V. Shmatikov. Probabilistic analysis of anonymity. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, pages 119–128. IEEE Computer Society Press, 2002.

66 A. Simaitis. *Automatic Verification of Competitive Stochastic Systems.* PhD thesis, Department of Computer Science, University of Oxford, 2014.

67 M. Svorenova and M. Kwiatkowska. Quantitative verification and strategy synthesis for stochastic games. *European Journal of Control*, 2016. To appear.

68 M. Ujma. *On Verification and Controller Synthesis for Probabilistic Systems at Runtime.* PhD thesis, University of Oxford, 2015.

69 Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite state programs. *Foundations of Computer Science, IEEE Annual Symposium on*, 0:327–338, 1985.

70 C. Wiltsche. *Assume-Guarantee Strategy Synthesis for Stochastic Games.* PhD thesis, Department of Computer Science, University of Oxford, 2015.

Fine-Grained Complexity Analysis of Two Classic TSP Variants*

Mark de Berg[1], Kevin Buchin[2], Bart M. P. Jansen[3], and Gerhard Woeginger[4]

1 **Eindhoven University of Technology, Eindhoven, The Netherlands**
`m.t.d.berg@tue.nl`
2 **Eindhoven University of Technology, Eindhoven, The Netherlands**
`k.a.buchin@tue.nl`
3 **Eindhoven University of Technology, Eindhoven, The Netherlands**
`b.m.p.jansen@tue.nl`
4 **Eindhoven University of Technology, Eindhoven, The Netherlands**
`g.woeginger@tue.nl`

Abstract

We analyze two classic variants of the Traveling Salesman Problem using the toolkit of fine-grained complexity.

Our first set of results is motivated by the Bitonic tsp problem: given a set of n points in the plane, compute a shortest tour consisting of two monotone chains. It is a classic dynamic-programming exercise to solve this problem in $\mathcal{O}(n^2)$ time. While the near-quadratic dependency of similar dynamic programs for Longest Common Subsequence and Discrete Fréchet Distance has recently been proven to be essentially optimal under the Strong Exponential Time Hypothesis, we show that bitonic tours can be found in subquadratic time. More precisely, we present an algorithm that solves bitonic tsp in $\mathcal{O}(n \log^2 n)$ time and its bottleneck version in $\mathcal{O}(n \log^3 n)$ time. In the more general pyramidal tsp problem, the points to be visited are labeled $1, \ldots, n$ and the sequence of labels in the solution is required to have at most one local maximum. Our algorithms for the bitonic (bottleneck) tsp problem also work for the pyramidal tsp problem in the plane.

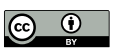
Our second set of results concerns the popular k-opt heuristic for tsp in the graph setting. More precisely, we study the k-opt decision problem, which asks whether a given tour can be improved by a k-opt move that replaces k edges in the tour by k new edges. A simple algorithm solves k-opt in $\mathcal{O}(n^k)$ time for fixed k. For 2-opt, this is easily seen to be optimal. For $k = 3$ we prove that an algorithm with a runtime of the form $\widetilde{\mathcal{O}}(n^{3-\varepsilon})$ exists if and only if All-Pairs Shortest Paths in weighted digraphs has such an algorithm. For general k-opt, it is known that a runtime of $f(k) \cdot n^{o(k/\log k)}$ would contradict the Exponential Time Hypothesis. The results for $k = 2, 3$ may suggest that the actual time complexity of k-opt is $\Theta(n^k)$. We show that this is not the case, by presenting an algorithm that finds the best k-move in $\mathcal{O}(n^{\lfloor 2k/3 \rfloor + 1})$ time for fixed $k \geq 3$. This implies that 4-opt can be solved in $\mathcal{O}(n^3)$ time, matching the best-known algorithm for 3-opt. Finally, we show how to beat the quadratic barrier for $k = 2$ in two important settings, namely for points in the plane and when we want to solve 2-opt repeatedly.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory, I.2.8 Problem Solving, Control Methods, and Search

Keywords and phrases Traveling salesman problem, fine-grained complexity, bitonic tours, k-opt

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.5

* This work was supported by NWO Gravitation grant 024.002.003 "Networks" (all authors), NWO grant 612.001.207 "A framework for progressive, user-steered algorithms in visual analytics" (Buchin), and NWO Veni grant "Frontiers in Parameterized Preprocessing" (Jansen).



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 5; pp. 5:1–5:14

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

1 Introduction

1.1 Motivation

We analyze two classic variants of the Traveling Salesman Problem (TSP) by applying the modern toolkit of fine-grained complexity analysis. The first TSP variant can for instance be found in Chapter 15 of the well-known textbook *"Introduction to Algorithms"* by Cormen, Leiserson, Rivest, and Stein [15]. The chapter discusses dynamic programming, and its problem section poses the following classic exercise:

> ***15-3 Bitonic euclidean traveling-salesman problem***
> In the ***euclidean traveling-salesman problem***, we are given a set of n points in the plane, and we wish to find the shortest closed tour that connects all n points. The general problem is NP-complete, and its solution is therefore believed to require more than polynomial time. J. L. Bentley has suggested that we simplify the problem by restricting our attention to ***bitonic tours***, that is, tours that start at the leftmost point, go strictly rightward to the rightmost point, and then go strictly leftward back to the starting point. In this case, a polynomial-time algorithm is possible. Describe an $\mathcal{O}(n^2)$-time algorithm for determining an optimal bitonic tour.

This exercise already showed up in the very first edition of the book in 1991. Since then, thousands of students pondered about it and (hopefully) found the solution. One might wonder whether $\mathcal{O}(n^2)$ runtime is best possible for this problem. As one of our main contributions, we will show that in fact it is not.

The second TSP variant concerns k-OPT, a popular local search heuristic that attempts to improve a suboptimal solution by a k-OPT *move* (or: *k-move* for short), which is an operation that removes k edges from the current tour and reconnects the resulting pieces into a new tour by inserting k new edges. The cases $k = 2$ [16] and $k = 3$ have been studied extensively with respect to various aspects such as experimental performance [7, 24, 27], (smoothed) approximation ratio [13, 26], rate of convergence [13, 17], and algorithm engineering [19, 21, 29, 30]. The decision problem associated with k-OPT asks, given a tour in an edge-weighted graph, whether it is possible to obtain a tour of smaller weight by replacing k edges. There are $\Theta(n^k)$ possibilities to choose k edges that leave the current tour, and for each choice the number of ways to reconnect the resulting pieces back into a tour is constant (for fixed k). As the weight change for each reconnection pattern can be evaluated in $\mathcal{O}(k)$ time, this simple algorithm finds the best k-OPT improvement in time $\mathcal{O}(n^k)$ for each fixed k. The survey chapter [25] by Johnson and McGeoch extensively discusses k-OPT. On page 233 they write:

> To complete our discussion of running times, we need to consider the time per move as well as the number of moves. This includes the time needed to *find* an improving move (or verify that none exists), together with the time needed to *perform* the move. In the worst case, 2-opt and 3-opt require $\Omega(n^2)$ and $\Omega(n^3)$ time respectively to verify local optimality, assuming all possible moves must be considered.

The two lower bounds in the last sentence are stated without further justification. It is clear that finding an improving k-move takes $\Omega(n^k)$ time, if we require that all possible moves must be enumerated *explicitly*. However, one might wonder whether there are other,

faster algorithmic approaches that proceed without enumerating all moves. As one of our main contributions, we will show that such faster approaches do not exist for $k = 3$ (under the All-Pairs Shortest Paths conjecture), but do exist for all $k \geq 4$.

1.2 Our contributions

We investigate whether the long-standing runtimes of $\mathcal{O}(n^2)$ for bitonic tours and $\mathcal{O}(n^k)$ for finding k-opt improvements are optimal. Such optimality investigations usually involve two ingredients: fast algorithms and runtime lower bounds. While proving unconditional lower bounds is far out of reach, in recent years there has been an influx of techniques for establishing lower bounds on the running time of a given problem, based on a hypothesis about the best-possible running time for another problem. Recent results in this direction consider the problems of computing the Longest Common Subsequence [1, 10] of two length-n strings, the Edit Distance [5, 10] from one length-n string to another, or the Discrete Fréchet Distance [9] between two polygonal n-vertex curves in the plane. If one of these problems allows an algorithm with running time $\mathcal{O}(n^{2-\varepsilon})$, then this would yield an algorithm to test the satisfiability of an n-variable CNF formula ϕ in time $(2-\varepsilon)^n \cdot |\phi|^{\mathcal{O}(1)}$. As decades of research have not led to algorithms with such a running time for cnf-sat, this gives evidence that the classic $\mathcal{O}(n^2)$-time algorithms for these problems are optimal up to $n^{o(1)}$ factors.

Pyramidal tours in the plane. Consider a symmetric tsp instance that is defined by an edge-weighted complete graph. For a linear ordering $1, \ldots, n$ of the vertices in the graph, a *pyramidal* tour has the form $(1, i_1, \ldots, i_r, n, j_1, \ldots, j_{n-r-2})$, where $i_1 < i_2 < \ldots < i_r$ and $j_1 > j_2 > \ldots > j_{n-r-2}$. A *bitonic* tour for a Euclidean tsp instance is pyramidal with respect to the left-to-right order on the points in the plane. Bitonic and pyramidal tours play an important role in the combinatorial optimization literature on the tsp; see [6, 11, 20]. They form an exponentially large set of tours over which we can optimize efficiently, and they lead to well-solvable special cases of the tsp. Combined with a procedure for generating suitable permutations of the vertices, heuristic solutions to tsp can be obtained by computing optimal pyramidal tours with respect to the generated orders [12].

We will show that the classic $\mathcal{O}(n^2)$ dynamic program for finding bitonic tours in the Euclidean plane is far from optimal: by an appropriate use of dynamic geometric data structures, the running time can be reduced to $\mathcal{O}(n \log^2 n)$. To the best of our knowledge, this presents the first improvement in finding bitonic tours since the problem was popularized in *Introduction to Algorithms* [15] in 1991. In fact, we prove the stronger result that an optimal *pyramidal* tour among n points in the plane can be computed in $\mathcal{O}(n \log^2 n)$ time with respect to any given linear order on the points. Our techniques extend to the related Bottleneck Pyramidal tsp problem in the plane, where the goal is to find a pyramidal tour among the cities that minimizes the length of the longest edge. We prove that the underlying decision problem (given a linearly ordered set of points and a bottleneck value B, is there a pyramidal tour of the points whose longest edge has length at most B?) can be solved in $\mathcal{O}(n \log n)$ time, while the underlying optimization version (given a linearly ordered set of points, compute a bitonic tour that minimizes the length of the longest edge) can be solved in $\mathcal{O}(n \log^3 n)$ time. For the decision version of the bottleneck problem, we prove a matching $\Omega(n \log n)$ time lower bound in the algebraic computation tree model by a reduction from Set Disjointness with integer inputs [34]; this reduction even applies to the bitonic setting where the points are ordered from left to right.

***k*-OPT in the graph setting.** The complexity of k-OPT has been analyzed using the framework of parameterized complexity theory. Marx [28] proved that deciding whether there is a k-move that improves a given tour is W[1]-hard parameterized by k, giving evidence that there is no algorithm with runtime $f(k) \cdot n^{\mathcal{O}(1)}$. Guo *et al.* [22] refined this result and proved that, under the *Exponential Time Hypothesis* [23], there is no algorithm that determines whether a tour in a weighted complete graph can be improved by a k-move in time $f(k) \cdot n^{o(k/\log k)}$ for any function f. This lower bound shows that the exponent of n in the runtime of any k-OPT algorithm must grow almost linearly with k. The next question that we settle in this paper is: can one do better than $\mathcal{O}(n^k)$ for finding a k-OPT improvement? The answer turns out to depend on the value of k. For 2-OPT, an easy adversarial argument shows that any deterministic algorithm must inspect all the edge weights. This gives a trivial lower bound of $\Omega(n^2)$, matching the upper bound. For larger values of k, the question becomes more interesting.

The 3-OPT DETECTION problem asks whether the weight of a given tour can be reduced by some 3-move. We show that it is unlikely that 3-OPT DETECTION with weights in the range $[-M, \ldots, M]$ allows an algorithm with a *truly subcubic* runtime of $\mathcal{O}(n^{3-\varepsilon} \operatorname{polylog}(M))$ for $\varepsilon > 0$. We prove that the NEGATIVE EDGE-WEIGHTED TRIANGLE problem (given an edge-weighted graph, is there a triangle of negative weight?) reduces to 3-OPT DETECTION by a reduction that takes $\mathcal{O}(n^2)$ time and increases the size of the graph by only a constant factor. As NEGATIVE EDGE-WEIGHTED TRIANGLE is equivalent to ALL-PAIRS SHORTEST PATHS in weighted digraphs (APSP) with respect to having truly subcubic algorithms [33], a truly subcubic algorithm for 3-OPT DETECTION would contradict the APSP conjecture [2, 3] which states that APSP cannot be solved in truly subcubic time. We also give a reduction in the other direction: finding a 3-OPT improvement reduces to finding a negative edge-weighted triangle. Consequently, 3-OPT DETECTION is *equivalent* to NEGATIVE EDGE-WEIGHTED TRIANGLE and APSP with respect to truly subcubic runtimes. This adds yet another classic problem to the growing list of such equivalent problems [2, 33].

As a final result in this direction, we design an algorithm that finds the best k-OPT improvement in weighted n-vertex complete graphs in $\mathcal{O}(n^{\lfloor 2k/3 \rfloor + 1})$ time for each fixed value of k. For $k = 2$ and $k = 3$, this expression simply boils down to the straightforward time complexities of $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ for 2-OPT and 3-OPT respectively. For $k \geq 4$, however, our result yields a substantial improvement over the trivial $\mathcal{O}(n^k)$ time bound. For example, 4-OPT can be solved in $\Theta(n^3)$ time, matching the best-known algorithm for 3-OPT. The algorithm mixes enumeration of partial solutions with a simple dynamic program.

Faster 2-OPT in the repeated setting and in the planar setting. For the 2-OPT problem in graphs, the runtime for finding a single tour improvement cannot be improved below the trivial $\Theta(n^2)$. However, in the context of local search we are often interested in *repeatedly* finding tour improvements. It is therefore natural to consider whether speedups can be obtained when repeatedly finding improving tours on the same TSP instance. We prove that this is indeed the case: after $\mathcal{O}(n^2)$ preprocessing time, one can repeatedly find the best 2-OPT improvement in $\mathcal{O}(n \log n)$ time per iteration.

The quadratic lower bound for 2-OPT applies only in the graph setting. This raises the question: can we solve 2-OPT faster for points in the plane? We show the answer is yes, by giving an algorithm for 2-OPT DETECTION with runtime $\mathcal{O}(n^{8/5+\varepsilon})$ for all $\varepsilon > 0$. Similarly, 3-OPT DETECTION can be solved in expected time $\mathcal{O}(n^{80/31+\varepsilon})$.

2 Faster pyramidal TSP

In this section we show that the pyramidal TSP and the bottleneck pyramidal TSP problem in the plane can be solved in subquadratic time. For simplicity we only show how to compute the value of an optimal solution; the actual tour can be computed in the standard manner.

Let P be the ordered input set of n points with distinct x-coordinates in the plane. Our algorithm will consider the points in P in order, and maintain a collection of partial solutions that are locally optimal. To make this precise, define $P_i := \{p_1, \ldots, p_i\}$ to be the first i points in P. A *partial solution* for P_i, for some $1 \leq i \leq n$, is a pair P', P'' of monotone paths (w.r.t. the order on P) that together visit all the points in P_i and that only share p_1. We call a partial solution for P_i an (i,j)*-partial tour*, for some $1 \leq j < i$, if one of the paths ends at p_i – this is necessarily the case in a partial solution for P_i – and the other path ends at p_j.

Our starting point is the standard dynamic-programming solution. It uses a 2-dimensional table[1] $A[1..n, 1..n]$, where $A[i,j]$, for $1 \leq j < i \leq n$, is defined as the minimum length of an (i,j)-partial tour; for $i \leq j \leq n$ the entries $A[i,j]$ are undefined. We can compute the entries in the table row by row, using the recursive formula

$$A[i+1,j] = \begin{cases} A[i,j] + |p_i p_{i+1}| & \text{if } 1 \leq j < i \\ \min_{1 \leq k < i} \left(A[i,k] + |p_k p_{i+1}|\right) & \text{if } j = i \end{cases} \tag{1}$$

where $A[2,1] = |p_1 p_2|$. Let us briefly verify this recurrence. For $(i+1,j)$-partial tours with $j < i$, the path P' that visits p_{i+1} must also visit p_i: the other path P'' ends at index $j < i$ and the monotonicity requirement ensures P'' cannot visit i and go back to j. So for $j < i$ any $(i+1,j)$-partial tour consists of an (i,j)-partial tour together with the segment $p_i p_{i+1}$. For $(i+1,i)$-partial tours, the predecessor of p_{i+1} cannot be p_i, since a path ends at p_i. Hence an $(i+1,i)$-partial tour consists of an (i,k)-partial tour for some $1 \leq k < i$ together with the segment $p_k p_{i+1}$. The cheapest combination yields the best partial tour.

After computing the last row of A, the minimum length of a pyramidal tour can be found by computing $\min_{1 \leq k < n} \left(A[n,k] + |p_k p_n|\right)$. There are $\mathcal{O}(n^2)$ entries in A of the first type that each take constant time to evaluate. There are $\mathcal{O}(n)$ entries of the second type that need time $\Theta(n)$. Hence the dynamic program can be evaluated in $\mathcal{O}(n^2)$ time.

Our subquadratic algorithm is based on the following two observations. First, any two subsequent rows $A[i, 1..n]$ and $A[i+1, 1..n]$ are quite similar: the entries $A[i+1,j]$, for $j < i$, can all be obtained from $A[i,j]$ by adding the same value, namely $|p_i p_{i+1}|$. Second, the computation of $A[i+1,i]$ can be sped up using appropriate geometric data structures. Thus our algorithm will maintain a data structure that implicitly represents the current row and allows for fast queries and so-called bulk updates (see below).

Recall that $P_i := \{p_1, \ldots, p_i\}$. The point that defines $\min_{1 \leq k < i} \left(A[i,k] + |p_k p_{i+1}|\right)$ is the point $p_k \in P_{i-1}$ closest to the query point $q := p_{i+1}$ if we use the additively weighted distance function

$$\text{dist}(p_k, q) := w_k + |p_k q|, \tag{2}$$

where $w_k := A[i,k]$ is the weight of p_k. Thus we need a data structure for storing a weighted point set that supports the following operations:

[1] Some of our results can also be obtained from an alternative DP with n states. As we need the 2-dimensional approach for Theorem 4, we present all our results in this setting.

- perform a *nearest-neighbor query* with a query point q, which reports the point p_k closest to q according to the additively weighted distance function,
- perform a *bulk update* of the weights, which adds a given value Δ to the weights of all the points currently stored in the data structure;
- *insert* a new point with a given weight into the data structure.

Answering nearest-neighbor queries for the weighted point set P can be done by performing point location in the *additively weighted Voronoi diagram* [18] of P augmented by a point location data structure [32]. This (static) data structure has size $\mathcal{O}(n)$, can be computed in $\mathcal{O}(n \log n)$ time, and allows for $\mathcal{O}(\log n)$-time queries. To allow for insertions we use the logarithmic method [8]. The logarithmic method makes a data structure semi-dynamic by storing $\mathcal{O}(\log n)$ static data structures of increasing size (resulting in an additional log-factor in the query time). The main observation is that we can handle bulk updates by storing a correction term for the weights with each of the static additively weighted Voronoi diagrams. The additively-weighted nearest neighbor structure does not change when adding the same constant to each point weight, which means we do not have to update the Voronoi diagrams when performing bulk updates. This leads to an implementation that supports each operation in $\mathcal{O}(\log^2 n)$ amortized time. The details are deferred to the full version. Using the data structure we obtain the following theorem.

▶ **Theorem 1.** *Let P be an ordered set of n points in the plane. Then we can compute a minimum-length pyramidal tour for P in $\mathcal{O}(n \log^2 n)$ time and using $\mathcal{O}(n)$ storage.*

Proof. We aim to speed up the classic dynamic-programming algorithm using the data structure described above. Instead of computing the entire dynamic programming table A explicitly, we maintain an implicit representation of one row of the table and compute the rows one by one. The i-th row of A has $i-1$ well-defined entries. We define an implicit representation of row i to be an instance of the data structure storing the weighted point set $P_{i-1} = \{p_1, \ldots, p_{i-1}\}$ such that $w(p_j) = A[i,j]$. The first nontrivial row in A is the second row, $A[2, 1..n]$. An implicit representation for that row consists of the point p_1 of weight $A[2,1] = |p_1 p_2|$.

If we have an implicit representation of row i, we can efficiently obtain an implicit representation of row $i+1$, as we describe next. By our choice of implicit representation, the value of $A[i+1, i]$ according to (1) is exactly the distance from p_{i+1} to its closest neighbor in the data structure under the additively weighted distance function. Hence, the value of k that minimizes the lower expression in (1) can be found by a nearest neighbor query with p_{i+1}. We can therefore transform a representation of row i into a representation for row $i+1$ as follows:

1. Query with point p_{i+1} to find the value $A[i+1, i]$ and remember this value.
2. Perform a bulk update to increase the weight of the points $p_1, \ldots, p_{i-1}$ that are already in the structure by $\Delta := |p_i p_{i+1}|$. Recall that for cells j with $1 \leq j < i$ their value in row $i+1$ is obtained from their value in row i by adding $|p_i p_{i+1}|$.
3. Insert point p_i of weight $A[i+1, i]$ into the structure.[2]

It is easy to verify that this yields an implicit representation of row $i+1$. Since a representation of the first nontrivial row can be found in constant time, and each successive row can be computed from the previous using three data structure operations that take $\mathcal{O}(\log^2 n)$

[2] We could also insert p_i with weight $A[i+1, i] - \Delta$. This way we would not have to subtract Δ from the weights of $p_1, \ldots, p_{i-1}$ in Step 2, and the bulk updates are not needed. As they are trivial in our data structure, we prefer the version that keeps the correspondence between weights and $A[i,j]$ values.

amortized time each, it follows that an implicit representation of the final row can be computed in $\mathcal{O}(n \log^2 n)$ time. The minimum cost of a pyramidal tour is $\min_{1 \leq k < n} (A[n,k] + |p_k p_n|)$, which can be found by querying the representation of the final row with point p_n. ◀

Bottleneck pyramidal TSP. Using a similar global approach but different supporting data structures we can also solve the bottleneck version of the problem – here the goal is to minimize the length of the longest edge in the tour – in subquadratic time. For the decision version of the problem we need the following result.

▶ **Theorem 2.** *We can maintain a collection $\mathcal{D}$ of n congruent disks in a data structure such that we can decide in $\mathcal{O}(\log n)$ time if a query point q lies in* Union($\mathcal{D}$). *The data structure uses $\mathcal{O}(n)$ storage and a new disk can be inserted into $\mathcal{D}$ in $\mathcal{O}(\log n)$ amortized time.*

This result is obtained as follows. Assume the disks have radius $\sqrt{2}$ and consider the integer grid. Let $\mathcal{D}(C) \subseteq \mathcal{D}$ be the set of disks whose centers lie inside a grid cell C. To decide if $q \in$ Union($\mathcal{D}$) we need to test if $q \in$ Union($\mathcal{D}(C)$) for $O(1)$ grid cells C that are sufficiently close to q. Now consider a cell C with $\mathcal{D}(C) \neq \emptyset$. Obviously C itself is completely covered by Union($\mathcal{D}(C)$). Let $\ell_{\text{top}}(C)$ be the line containing the top edge of C. Then the part of Union($\mathcal{D}(C)$) above $\ell_{\text{top}}(C)$ – the other parts are handled similarly – is x-monotone. Moreover, we can show that each disk $D_i \in \mathcal{D}(C)$ contributes at most one arc to the boundary of Union($\mathcal{D}(C)$) above $\ell_{\text{top}}(C)$, and the left-to-right order of the contributed arcs is consistent with the left-to-right order of the corresponding disk centers. Using this fact, we can do point locations and insertions in $O(\log n)$ time. Details can be found in the full version.

Combining the global technique of the previous section with Theorem 2 we obtain the following theorem.

▶ **Theorem 3.** *Let P be an ordered set of n points in the plane, and let $B > 0$ be a given parameter. Then we can decide in $\mathcal{O}(n \log n)$ time and using $\mathcal{O}(n)$ storage if P admits a pyramidal tour whose longest edge has length at most B. This problem requires $\Omega(n \log n)$ time in the algebraic computation tree model of computation.*

The algorithm for the decision version does not easily extend to solve the minimization version of the problem. We therefore design a specialized data structure – a tree storing unions of disks and (regular) Voronoi diagrams – that allows us to obtain the following result.

▶ **Theorem 4.** *Let P be an ordered set of n points in the plane. Then we can compute a pyramidal tour whose bottleneck edge has minimum length in $O(n \log^3 n)$ time and using $\mathcal{O}(n \log n)$ storage.*

3 The k-OPT problem in general graphs

In this section we change the perspective from Euclidean problems to the TSP in general graphs. A *tour* of an undirected graph G is a Hamiltonian cycle in the graph. Depending on the context, we may treat a tour as a permutation of the vertex set or as a set of edges. We consider undirected, weighted complete graphs to model symmetric TSP inputs. The weight of a tour is simply the sum of the weights of its edges. Recall that a *k-move* of a tour T is an operation that replaces a set of k edges in T by another set of k edges from G in such a way that the result is a valid tour. In degenerate cases, such an operation may delete and reinsert the same edge. The associated decision problem is defined as follows.

k-opt Detection
Input: A complete undirected graph G along with a (symmetric) distance function $d\colon E(G) \to \mathbb{N}$, an integer k, and a tour $T \subseteq E(G)$.
Question: Is there a k-move that strictly improves the cost of T?

The optimization problem k-opt Optimization is to compute, given a tour in a graph, a k-move that gives the largest cost improvement, or report that no improving k-move exists.

3.1 On truly subcubic algorithms for 3-OPT

We say that an algorithm for n-vertex graphs with integer edge weights in the range $[-M, \ldots, M]$ runs in *truly subcubic time* if its runtime is bounded by $\mathcal{O}(n^{3-\varepsilon}\,\mathrm{polylog}(M))$ for some constant $\varepsilon > 0$. Vassilevska-Williams and Williams [33] introduced a framework for relating the truly subcubic solvability of several classic problems to each other. We use it to show that the existence of a truly subcubic algorithm for 3-opt is unlikely. Their framework uses a notion of subcubic reducibility based on Turing reducibility [33, §IV] that solves one instance of problem A by repeatedly solving inputs of problem B. For our applications, simple reductions suffice that transform one input of problem A into one input of problem B of roughly the same size, in $\mathcal{O}(n^2)$ time.[3] Such reductions preserve the existence of truly subcubic algorithms, so we take this simpler viewpoint. The following problem is the starting point for our reductions.

Negative Edge-Weighted Triangle
Input: An undirected, complete graph G and a weight function $w\colon E(G) \to \mathbb{Z}$.
Question: Does G contain a triangle whose total edge-weight is negative?

Vassilevska-Williams and Williams [33, Thm. 1.1] proved that Negative Edge-Weighted Triangle has a truly subcubic algorithm if and only if the All-Pairs Shortest Paths problem on digraphs with non-negative integral edge weights has a truly subcubic algorithm.

▶ **Lemma 5.** Negative Edge-Weighted Triangle *can be reduced to* 3-opt Detection *in time* $\mathcal{O}(n^2)$*, increasing the size of the graph and the largest weight by a constant factor.*

Proof. Consider an instance (G, w) of Negative Edge-Weighted Triangle, and let $v_1, \ldots, v_n$ be an enumeration of the vertices of G. Let M be the largest absolute value of an edge weight. We introduce an instance of 3-opt Detection that consists of $2n$ vertices $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, where the starting tour T uses the ordering $a_1, b_1, a_2, b_2, \ldots, a_n, b_n$. The (symmetric) distances $d(\cdot, \cdot)$ between these vertices are defined as follows:

- $d(a_i, b_i) = 0$ for $1 \leq i \leq n$;
- $d(b_n, a_1) = -3M$, and $d(b_i, a_{i+1}) = -3M$ for $1 \leq i \leq n-1$;
- $d(a_i, b_j) = w(\{v_i, v_j\})$ for $1 \leq i < j \leq n$;
- $d(b_i, a_j) = w(\{v_i, v_j\})$ for $1 \leq i < j - 1 \leq n - 1$;
- $d(a_i, a_j) = d(b_i, b_j) = 3M$ for $1 \leq i \neq j \leq n$.

(For convenience, we allow distances to be negative in this construction. One easily moves to non-negative distances by adding the constant $4M$ to all distances.)

▶ **Claim 6.** *The constructed instance of* 3-opt Detection *allows an improving* 3-opt *move, if and only if the graph G contains a triangle of negative edge-weight.*

[3] We assume that simple arithmetic on weights can be done in constant time. The $\mathrm{polylog}(M)$ factors used in the framework originate from repeated executions to perform binary search on weight values.

Proof. ($\Leftarrow$) Assume that the vertices v_i, v_j, v_k span a triangle of negative edge-weight in G for $i < j < k$. We remove the three edges $\{a_i, b_i\}$, $\{a_j, b_j\}$, and $\{a_k, b_k\}$ from tour T, and we reconnect the resulting pieces by the three edges $\{a_i, b_j\}$, $\{a_j, b_k\}$, and $\{a_k, b_i\}$. The three removed edges have total length 0, while the three inserted edges have negative total length.

($\Rightarrow$) Now assume that there exists an improving 3-move for tour T. This improving move cannot remove any edge $\{b_i, a_{i+1}\}$ or $\{b_n, a_1\}$, as these edges have length $-3M$ while all newly inserted edges have non-negative length. Consequently, the three removed edges will be $\{a_i, b_i\}$, $\{a_j, b_j\}$, and $\{a_k, b_k\}$ for some $i < j < k$. As these three edges have total length 0, the total length of the three inserted edges must be strictly negative. The edges $\{a_x, a_y\}$ and $\{b_x, b_y\}$ all have length $3M$, while the edges $\{a_x, b_y\}$ all have length between $-M$ and M. This implies that every inserted edge is either of the type $\{a_x, b_y\}$, or coincides with one of the removed edges. Suppose for the sake of contradiction that one of the inserted edges coincides with a removed edge $\{a_k, b_k\}$, so that we are actually dealing with a 2-move. Then the two inserted edges in the 2-move must be $\{a_i, a_j\}$ and $\{b_i, b_j\}$, so that the new tour is by $6M$ *longer* than the old tour T. This contradiction leaves only two possibilities for the three inserted edges: either $\{a_i, b_j\}$, $\{a_j, b_k\}$, $\{a_k, b_i\}$, or $\{a_i, b_k\}$, $\{a_k, b_j\}$, $\{a_j, b_i\}$ (of which the latter is actually not a valid 3-move). Since the total length of the three inserted edges is strictly negative, the three vertices v_i, v_j, v_k form a triangle of strictly negative weight in G. ◀

The claim shows the correctness of the reduction. It is easy to perform in $\mathcal{O}(n^2)$ time. ◀

There is an analogous reduction in the other direction, which can be found in the full version. Together, these lemmata show the equivalence of finding negative-weight triangles and detecting improving 3-OPT moves. From our reductions and the results of Vassilevska-Williams and Williams [33, Thm. 1.1], we obtain the following theorem.

▶ **Theorem 7.** *There is a truly subcubic algorithm for* 3-OPT DETECTION *if and only if there is such an algorithm for* ALL-PAIRS SHORTEST PATHS *on weighted digraphs.*

3.2 A fast k-OPT algorithm

We will prove that the k-OPT OPTIMIZATION problem can be solved significantly faster than $\Theta(n^k)$ when $k \geq 4$. To this end, we first analyze the structure of k-OPT moves. Consider a k-move for a given tour $T \subseteq E(G)$, and let $e_1, \ldots, e_k$ be the removed edges with $e_i = \{v_{2i-1}, v_{2i}\}$. We assume throughout that these vertices (and edges) are indexed in such a way that T traverses the vertices v_i in order of increasing index. We assume furthermore that the vertices $v_1, \ldots, v_{2n}$ are pairwise distinct; all our arguments also go through without this assumption, but the notation becomes more complicated in the equality case. The k edges that are then inserted into T are denoted $f_1, \ldots, f_k$. The *signature* of this k-move is a permutation π of $\{1, \ldots, 2k\}$, such that v_j and $v_{\pi(j)}$ form the endpoints of one of the edges $f_1, \ldots, f_k$; see Fig. 1.

Note that the removed edges $e_1, \ldots, e_k$ together with the signature π fully determine the k-move (and in particular determine the inserted edges $f_1, \ldots, f_k$).

Note furthermore that not every permutation π yields a feasible signature that corresponds to some k-move: First, in a feasible signature $\pi(i) = j$ always implies $\pi(j) = i$, and we will always have $\pi(i) \neq i$. Secondly, in a feasible signature the edge set that results from T by removing $e_1, \ldots, e_k$ and by inserting $f_1, \ldots, f_k$ must form a single Hamiltonian cycle – it must never form a collection of two or more cycles. It is easy to check whether a given permutation π constitutes a feasible signature, and to enumerate all feasible signatures.

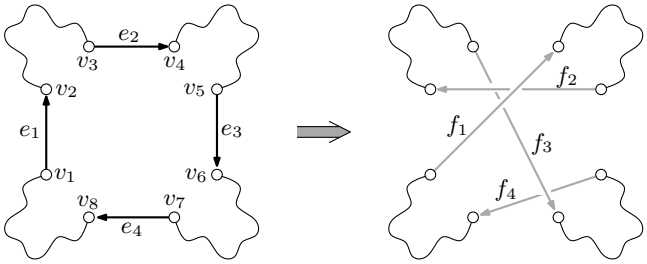


Figure 1 A 4-change with signature 4,5,7,1,2,8,3,6. Edges e_1 and e_4 are non-interfering. As we work on symmetric TSP, the graph and distance function are undirected; the arc directions merely indicate the traversal direction with respect to an arbitrary orientation of the tour.

We say that two of the removed edges e_i and e_j *interfere* with each other in a k-move, if there exists an inserted edge f that connects one of the endpoints of e_i to an endpoint of e_j.

▶ **Lemma 8.** *For any signature π, we can find a subset $E_\pi \subseteq \{e_1, \ldots, e_k\}$ of at least $\lceil k/3 \rceil$ removed edges that are pairwise non-interfering.*

Proof. The $2k$ edges $e_1, \ldots, e_k$ and $f_1, \ldots, f_k$ induce a set of cycles on the vertices $v_1, \ldots, v_{2k}$. If such a cycle contains an even number of removed edges, say 2ℓ, we put every other removed edge along this cycle into E_π; this yields ℓ out of 2ℓ edges for E_π. If the cycle contains only a single removed edge, we put this single edge into E_π; this yields one out of one edge for E_π. If the cycle contains an odd number of removed edges, say $2\ell + 1 \geq 3$, we ignore the first removed edge and then put every other removed edge along the cycle into E_π; this yields ℓ out of $2\ell + 1$ edges for E_π. The weakest contribution to E_π comes from cycles with three removed edges, which yield only one out of three edges for E_π. The claimed bound $\lceil k/3 \rceil$ follows. ◀

▶ **Theorem 9.** *For every fixed $k \geq 3$, the k-opt Optimization problem on an n-vertex graph can be solved in $\mathcal{O}(n^{\lfloor 2k/3 \rfloor + 1})$ time.*

Proof. For computing the best k-move for tour T, it is sufficient to compute for every feasible signature π – for fixed k there are only $\mathcal{O}(1)$ such signatures – the best k-move for tour T with that particular signature. This is done as follows. We first determine a set E_π of pairwise non-interfering edges according to the above lemma. Then we enumerate and handle all possible cases for the locations of the $\lfloor 2k/3 \rfloor$ removed edges not in E_π along T. This yields $\mathcal{O}(n^{\lfloor 2k/3 \rfloor})$ cases to handle, and every such case will be handled in $\mathcal{O}(n)$ time; note that this yields the claimed complexity. In handling a case, the positions of the removed edges not in E_π are frozen, while the edges in E_π have to be embedded into T. The cost of a k-move with signature π decomposes into two parts:

- The first part consists of the total weight of all frozen edges (which is subtracted) and the total weight of inserted edges between frozen edges (which is added).
- The second part consists of the individual contributions of the edges in E_π. For an edge $e \in E_\pi$ and an edge $e' \in T$, the cost of embedding e into e' equals the weight of the two inserted edges adjacent to e minus the weight of e'. As the edges in E_π are pairwise non-interfering, their individual cost contributions do not interact with each other.

As the cost of the first part is fixed in every considered case, our goal is to minimize the total cost of the second part. The frozen edges subdivide the tour T into a number of tour pieces, and we have to find the cheapest way of embedding the corresponding edges from E_π

into such a tour piece. The following paragraph sketches a straightforward dynamic program for finding the optimal embedding for each tour piece in time proportional to the length of the piece. As the length of all tour pieces combined is $\mathcal{O}(n)$, every case is indeed handled in time $\mathcal{O}(n)$.

We are essentially dealing with the following optimization problem. There are r locations $L_1, \ldots, L_r$ (the edges along tour T between two consecutive frozen edges) and s objects $O_1, \ldots, O_s$ (the edges in E_π that should be embedded between the two considered frozen edges). The objects are to be embedded into the locations, so that the location of object O_i always precedes the location of object O_{i+1}. The cost of embedding object O_i into location L_j is denoted $c(i, j)$. For $1 \leq x \leq s$ and $1 \leq y \leq r$, let $V(x, y)$ denote the smallest possible cost incurred by embedding the first x objects $O_1, \ldots, O_x$ into the first y locations $L_1, \ldots, L_y$. As $V(x, y)$ equals the minimum of $V(x, y-1)$ and $V(x-1, y-1) + c(x, y)$, all these values $V(x, y)$ can easily be computed in $\mathcal{O}(rs)$ time. In our situation, r is the length of the considered tour piece and $s \leq k$ is a constant that does not depend on the input; hence the complexity is indeed proportional to the length of the considered tour piece. ◀

4 Faster 2-OPT

In this section we show that it is possible to beat the quadratic barrier for 2-OPT in two important settings, namely when we want to apply 2-moves repeatedly, and in the Euclidean setting in the plane.

Repeated 2-OPT. In the repeated 2-OPT problem, we apply 2-OPT repeatedly (e.g. until no further improvements are possible). One can considerably speed up the 2-OPT computations at each of the iterations, except the first one. The following theorem gives our improvement for the 2-OPT OPTIMIZATION problem, where the goal is to find the best 2-move (rather than any 2-move that improves the tour).

▶ **Theorem 10.** *After $\mathcal{O}(n^2)$ preprocessing and using $\mathcal{O}(n^2)$ storage we can repeatedly solve the* 2-OPT OPTIMIZATION *problem in $\mathcal{O}(n \log n)$ time per iteration.*

The speedup claimed in the theorem relies on a tour representation that supports efficient 2-moves. To apply a 2-move that removes two edges e and e' and replaces them by the appropriate diagonal connections, one effectively has to reverse the part of the tour between e and e', or the part between e' and e. It can therefore take $\Omega(n)$ time to apply a 2-move to a tour represented as a sequence of vertices in an array. Chrobak *et al.* [14] give a speedup by storing the cities on the tour in an ordered balanced binary search tree. Each node in the tree stores a bit indicating whether the tour order is given by an in-order traversal of the subtree rooted there, or by the *reverse* of the in-order traversal. This allows a 2-move to be applied in $\mathcal{O}(\log n)$ time by manipulating reversal bits.

Our approach for repeated 2-OPT OPTIMIZATION is based on a similar data structure that represents tours in balanced search trees. However, instead of having only one tree that stores the current tour, we have n trees; one for each edge $e_1, \ldots, e_n$ in the current tour. A query in the tree $\mathcal{T}(e_i)$ corresponding to edge e_i can be used to determine which edge e_j yields the most profitable 2-move together with e_i. After initializing these n trees, which takes $\mathcal{O}(n^2)$ time, an iteration of 2-OPT OPTIMIZATION can be performed as follows. For each e_i on the current tour, we query in tree $\mathcal{T}(e_i)$ to find the best 2-move that removes e_i and some unknown edge e_j in $\mathcal{O}(\log n)$ time. In this way we find the best overall 2-move which removes, say, edges e_i and e_j. We can update all trees $\mathcal{T}(e_\ell)$ for $\ell \neq i, j$ by deleting e_i

and e_j, and inserting the appropriate replacement edges. Using the reversal bits this can be done in $O(\log n)$ time. Trees $\mathcal{T}(e_i)$ and $\mathcal{T}(e_j)$ are destroyed; we build two new trees from scratch for the two new edges $e_{i'}$ and $e_{j'}$ that enter the tour. This gives $\mathcal{O}(n \log n)$ time per iteration.

It is likely that these techniques can be extended to speed up repeated 3-OPT as well. As the technical details become substantially more cumbersome, we do not pursue this direction.

The planar case. For points in the plane (and under the Euclidean metric) we can speed up 2-OPT computations by using suitable geometric data structures for semi-algebraic range searching; the details had to be omitted from this extended abstract. (Note that we do not consider the repeated version of the problem, but the single-shot version.) A similar approach can be used to speed up 3-OPT in the Euclidean setting in the plane. This leads to the following theorem.

▶ **Theorem 11.** *For any fixed $\varepsilon > 0$,* 2-OPT DETECTION *in the plane can be solved in $\mathcal{O}(n^{8/5+\varepsilon})$ time, and* 3-OPT DETECTION *in the plane can be solved in $\mathcal{O}(n^{80/31+\varepsilon})$ expected time.*

5 Conclusion

Revisiting the worst-case complexity of k-OPT and pyramidal TSP led to a number of new results on these classic problems. Some, such as the equivalence between 3-OPT and APSP with respect to having truly subcubic algorithms, rely on very recent work. Other results, such as the near-linear time algorithm for finding bitonic tours, and the k-OPT algorithm that beats the trivial $\mathcal{O}(n^k)$ upper bound, are obtained using classic techniques. In this respect, it is surprising that these results were not found earlier. These examples show that the availability of new lower bound machinery can inspire new algorithms.

Our findings suggest several directions for further research, both theoretical and applied. An interesting open problem regarding k-OPT DETECTION is whether the problem is fixed-parameter tractable when improving a given tour in an edge-weighted planar graph. This question was also asked by Marx [28] and Guo et al. [22]. Similarly, it is open whether the problem is fixed-parameter tractable when improving a given tour among points in the Euclidean plane. It would be interesting to settle the exact complexity of k-OPT in general weighted graphs. Is $\Theta(n^{\lfloor \frac{2k}{3} \rfloor + 1})$ the optimal running time for k-OPT DETECTION? When all weights lie in the range $[-M, \ldots, M]$, one can detect a negative triangle in an edge-weighted graph in time $\mathcal{O}(M \cdot n^\omega)$ using fast matrix multiplication [4, 31, 35]. By our reduction, this gives an algorithm for 3-OPT DETECTION with weights $[-M, \ldots, M]$ in time $\mathcal{O}(M \cdot n^\omega)$. Can similar speedups be obtained for k-OPT for larger k?

Given the great industrial interest in TSP, establishing the practical applicability of these theoretical results is an important follow-up step. Several of our results rely on data structures that are efficient in theory, but which are currently impractical. These include the additively-weighted Voronoi diagram used for pyramidal tours on points in the plane, and the semi-algebraic range searching data structures used to speed up 2-OPT DETECTION. In contrast, the $\mathcal{O}(n^{\lfloor 2k/3 \rfloor + 1})$ algorithm for finding the best k-move improvement is self-contained, easy to implement, and may have practical potential.

Acknowledgments. We are grateful to Hans L. Bodlaender, Karl Bringmann, and Jesper Nederlof for insightful discussions, an anonymous referee for the observation in Footnote 1, and Christian Knauer for the observation in Footnote 2.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proc. 56th FOCS*, pages 59–78, 2015. doi:10.1109/FOCS.2015.14.

2 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska-Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proc. 26th SODA*, pages 1681–1697, 2015. doi:10.1137/1.9781611973730.112.

3 Amir Abboud, Virginia Vassilevska-Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proc. 47th STOC*, pages 41–50, 2015. doi:10.1145/2746539.2746594.

4 Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *J. Comput. Syst. Sci.*, 54(2):255–262, 1997. doi:10.1006/jcss.1997.1388.

5 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. 47th STOC*, pages 51–58, 2015. doi:10.1145/2746539.2746612.

6 Md. Fazle Baki and Santosh N. Kabadi. Pyramidal traveling salesman problem. *Computers & OR*, 26(4):353–369, 1999. doi:10.1016/S0305-0548(98)00067-7.

7 Jon Louis Bentley. Experiments on traveling salesman heuristics. In *Proc. 1st SODA*, pages 91–99, 1990.

8 J.L. Bently and J.B. Saxe. Decomposable searching problems I: Static-to-dynamic transformation. *J. Algorithms*, 1:301–358, 1980. doi:10.1016/0196-6774(80)90015-2.

9 Karl Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th FOCS*, pages 661–670, 2014. doi:10.1109/FOCS.2014.76.

10 Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In Venkatesan Guruswami, editor, *Proc. 56th FOCS*, pages 79–97. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.15.

11 Rainer E. Burkard, Vladimir G. Deineko, René van Dal, Jack A. A. van der Veen, and Gerhard J. Woeginger. Well-solvable special cases of the traveling salesman problem: A survey. *SIAM Review*, 40(3):496–546, 1998. doi:10.1137/S0036144596297514.

12 J. Carlier and P. Villon. A new heuristic for the travelling salesman problem. *RAIRO – Operations Research*, 24:245–253, 1990.

13 Barun Chandra, Howard J. Karloff, and Craig A. Tovey. New results on the old k-OPT algorithm for the traveling salesman problem. *SIAM J. Comput.*, 28(6):1998–2029, 1999. doi:10.1137/S0097539793251244.

14 M. Chrobak, T. Szymacha, and A. Krawczyk. A data structure useful for finding hamiltonian cycles. *Theoretical Computer Science*, 71(3):419–424, 1990. doi:10.1016/0304-3975(90)90053-K.

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

16 G. A. Croes. A method for solving traveling-salesman problems. *Operations Research*, 6:791–812, 1958. doi:10.1287/opre.6.6.791.

17 Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-opt algorithm for the TSP. *Algorithmica*, 68(1):190–264, 2014. doi:10.1007/s00453-013-9801-4.

18 Steven Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica*, 2:153–174, 1987. doi:10.1007/BF01840357.

19 Michael L. Fredman, David S. Johnson, Lyle A. McGeoch, and G. Ostheimer. Data structures for traveling salesmen. *J. Algorithms*, 18(3):432–479, 1995. doi:10.1006/jagm.1995.1018.

20 P.C. Gilmore, E.L. Lawler, and D.B. Shmoys. Well-solved special cases. In E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys, editors, *The Traveling Salesman Problem*, pages 87–143. Wiley, New York, 1985.

21 Fred Glover. Finding a best traveling salesman 4-Opt move in the same time as a best 2-Opt move. *J. Heuristics*, 2(2):169–179, 1996. doi:10.1007/BF00247211.

22 Jiong Guo, Sepp Hartung, Rolf Niedermeier, and Ondrej Suchý. The parameterized complexity of local search for TSP, more refined. *Algorithmica*, 67(1):89–110, 2013. doi:10.1007/s00453-012-9685-8.

23 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

24 D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In G. Gutin and A. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 369–443. Kluwer Academic Publishers, Dordrecht, 2002.

25 D.S. Johnson and L.A McGeoch. The traveling salesman problem: A case study in local optimization. In E Aarts and J.K. Lenstra, editors, *Local search in combinatorial optimization*, pages 215–310. Wiley, Chichester, 1997.

26 Marvin Künnemann and Bodo Manthey. Towards understanding the smoothed approximation ratio of the 2-opt heuristic. In *Proc. 42nd ICALP*, pages 859–871, 2015. doi:10.1007/978-3-662-47672-7_70.

27 Shen Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965. doi:10.1002/j.1538-7305.1965.tb04146.x.

28 Dániel Marx. Searching the k-change neighborhood for TSP is W[1]-hard. *Oper. Res. Lett.*, 36(1):31–36, 2008. doi:10.1016/j.orl.2007.02.008.

29 Ioannis Mavroidis, Ioannis Papaefstathiou, and Dionisios N. Pnevmatikatos. A fast FPGA-based 2-opt solver for small-scale euclidean traveling salesman problem. In *IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 13–22, 2007. doi:10.1109/FCCM.2007.40.

30 Molly A. O'Neil and Martin Burtscher. Rethinking the parallelization of random-restart hill climbing: a case study in optimizing a 2-opt TSP solver for GPU execution. In *Proceedings of the 8th Workshop on General Purpose Processing using GPUs*, pages 99–108, 2015. doi:10.1145/2716282.2716287.

31 Liam Roditty and Virginia Vassilevska-Williams. Minimum weight cycles and triangles: Equivalences and algorithms. In *Proc. 52nd FOCS*, pages 180–189, 2011. doi:10.1109/FOCS.2011.27.

32 Jack Snoeyink. Point location. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry (2nd ed.)*. CRC Press, 2004.

33 Virginia Vassilevska-Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Proc. 51th FOCS*, pages 645–654, 2010. doi:10.1109/FOCS.2010.67.

34 Andrew Chi-Chih Yao. Lower bounds for algebraic computation trees with integer inputs. *SIAM J. Comput.*, 20(4):655–668, 1991. doi:10.1137/0220041.

35 Gideon Yuval. An algorithm for finding all shortest paths using $n^{2.81}$ infinite-precision multiplications. *Inf. Process. Lett.*, 4(6):155–156, 1976. doi:10.1016/0020-0190(76)90085-5.

Bicovering: Covering Edges With Two Small Subsets of Vertices

Amey Bhangale[*1], **Rajiv Gandhi**[†2], **Mohammad T. Hajiaghayi**[‡3], **Rohit Khandekar**[4], **and Guy Kortsarz**[§5]

1 **Department of Computer Science, Rutgers University, New Brunswick, USA**
amey.bhangale@rutgers.edu
2 **Department of Computer Science, Rutgers University, Camden, USA**
rajivg@camden.rutgers.edu
3 **Department of Computer Science, University of Maryland, College Park, USA**
hajiagha@cs.umd.edu
4 **KCG holdings Inc., Jersey City, USA**
rkhandekar@gmail.com
2 **Department of Computer Science, Rutgers University, Camden, USA**
guyk@camden.rutgers.edu

Abstract

We study the following basic problem called Bi-Covering. Given a graph $G(V, E)$, find two (not necessarily disjoint) sets $A \subseteq V$ and $B \subseteq V$ such that $A \cup B = V$ and that every edge e belongs to either the graph induced by A or to the graph induced by B. The goal is to minimize $\max\{|A|, |B|\}$. This is the most simple case of the Channel Allocation problem [Gandhi et. al, Networks, 2006]. A solution that outputs $V, \emptyset$ gives ratio at most 2. We show that under the similar *Strong* Unique Game Conjecture by [Bansal-Khot, FOCS, 2009] there is no $2 - \epsilon$ ratio algorithm for the problem, for any constant $\epsilon > 0$.

Given a bipartite graph, Max-bi-clique is a problem of finding largest $k \times k$ complete bipartite sub graph. For Max-bi-clique problem, a constant factor hardness was known under random 3-SAT hypothesis of Feige [Feige, STOC, 2002] and also under the assumption that NP $\not\subseteq \cap_{\epsilon>0} \mathsf{BPTIME}(2^{n^\epsilon})$ [Khot, SIAM J. on Comp., 2011]. It was an open problem in [Ambühl et. al., SIAM J. on Comp., 2011] to prove inapproximability of Max-bi-clique assuming weaker conjecture. Our result implies similar hardness result assuming the Strong Unique Games Conjecture.

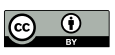
On the algorithmic side, we also give better than 2 approximation for Bi-Covering on numerous special graph classes. In particular, we get 1.876 approximation for Chordal graphs, exact algorithm for Interval Graphs, $1 + o(1)$ for Minor Free Graph, $2 - 4\delta/3$ for graphs with minimum degree δn, $2/(1 + \delta^2/8)$ for δ-vertex expander, $8/5$ for Split Graphs, $2 - (6/5) \cdot 1/d$ for graphs with minimum constant degree d etc. Our algorithmic results are quite non-trivial. In achieving these results, we use various known structural results about the graphs, combined with the techniques that we develop tailored to getting better than 2 approximation.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases Bi-covering, Unique Games, Max Bi-clique.

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.6

[*] Research supported in part by NSF grant CCF-1253886.
[†] Supported in part by NSF grant number 1218620.
[‡] Supported in part by NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant.
[§] Supported in part by NSF grant number 1218620 and by NSF grant 1540547.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 6; pp. 6:1–6:12

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

1 Introduction

We study the Bi-Covering problem - Given a graph $G(V, E)$, find two (not necessarily disjoint) sets $A, B \subseteq V$ such that $A \cup B = V$ and that every edge $e \in E$ belongs to either the graph induced by A or to the graph induced by B. The goal is to minimize $\max\{|A|, |B|\}$.

The problem we study is closely related to the problem of *Channel Allocation* which was studied in [10]. The Channel Allocation Problem can be described as follows: there is a universe of topics, a fixed number of channels and a set of requests where each request is a subset of topics. The task is to send a subset of topics through each channel such that each request is satisfied by set of topics from one of the channel i.e. for every request there must exists at least one channel such that the set of topics present in that channel is a superset of the set of topics from the request. Of course, one can achieve this task trivially by sending all topics through one channel. But, the optimization version of Channel Allocation Problem asks for a way to satisfy all the request by minimizing the maximum number of topics sent through a channel.

Any *connected* undirected graph $G(V, E)$ on n vertices and m edges along with an integer k can be viewed as a special case of channel allocation problem - The set of topics is a set of n vertices, each edge represents a request, where the requested set of topics corresponding to an edge is a pair of its endpoints and the number of channels is k. If we fix the number of channels to $k = 2$ then the optimization problem exactly corresponds to the Bi-Covering problem. Specifically, the optimization problem asks for two subsets A and B of V minimizing $\max\{|A|, |B|\}$ such that $A \cup B = V$ and every edge is totally contained in a graph induced by either A or B.

2 Our Results

Getting 2 approximation for Bi-Covering problem is trivial (by setting A = B = V). We show that Bi-Covering problem is hard to approximate within any factor strictly less than 2 assuming a strong Unique Games Conjecture (UGC) similar to the one in [5] (see Conjecture 12).

▶ **Theorem 1.** *Let $\epsilon > 0$ be any small constant. Assuming a strong Unique Games Conjecture (Conjecture 12), given a graph $G(V, E)$, it is* NP*-hard to distinguish between following two cases:*
1. *G has* Bi-Covering *of size at most $(1/2 + \epsilon)|V|$.*
2. *Any* Bi-Covering *of G has size at least $(1 - \epsilon)|V|$.*
In particular, it is NP-hard (assuming strong UGC) to approximate Bi-Covering *within a factor $2 - \epsilon$ for every $\epsilon > 0$.*

Given this structural hardness result, we get a $\frac{3}{2} - \epsilon$ hardness of Bi-Covering restricted to bipartite graphs by transforming a hard instance from Theorem 1 into a bipartite graph in a natural way (getting a $\frac{3}{2}$-approximation is easy - given a bipartite graph on X and Y with $|X| \geq |Y|$, one can take arbitrary partition X into two equal sized parts X_1 and X_2 and set the Bi-Covering to be $X_1 \cup Y$ and $X_2 \cup Y$).

▶ **Theorem 2.** *Assuming the strong Unique Games Conjecture, for every $\epsilon > 0$,* Bi-Covering *is* NP*-hard to approximate within a factor $\frac{3}{2} - \epsilon$ for bi-partite graphs.*

Our Theorem 1 implies hardness result for the following well known problem:

Max-Bi-Clique problem is as follows:

Input: A bipartite graph $G(X, Y, E)$ with $|X| = |Y| = n$.
Output: Find largest k such that there exists two subsets $A \subseteq X, B \subseteq Y$ of size k and the graph induced on (A, B) is a complete bipartite graph.

Inapproximability of Max-Bi-Clique problem has been studied extensively [1, 6, 8, 13]. Feige[8] showed that using an assumption of average case hardness of 3SAT instance, Max-Bi-Clique cannot be approximated within any constant factor in polynomial time (and hence within n^δ for some $\delta > 0$ using known amplification technique [1, 6]). Feige-Kogan [9] showed that assuming $SAT \notin DTIME(2^{n^{3/4+\epsilon}})$ there is no $2^{(\log n)^\delta}$ approximation for Max-Bi-Clique. They also showed that it is NP-hard to approximate Max-Bi-Clique within any constant factor assuming Max−Clique (finding a maximum sized clique in a graph) does not have a $n/2^{c\sqrt{\log n}}$-approximation. Khot [13] later proved a similar inapproximability result but assuming NP $\nsubseteq \cap_{\epsilon>0}$ BPTIME(2^{n^ϵ}) using a *quasi-random* PCP. It is an important open problem to extend similar hardness results based on weaker complexity assumptions [2]. In particular, it is still not known if UGC implies a constant factor hardness for Max-Bi-Clique. A straightforward corollary from Theorem 1 (see 4.2.2) implies that we get similar hardness results for Max-Bi-Clique based on Conjecture 12.

▶ **Corollary 3.** *Assuming strong Unique Games Conjecture, it is* NP*-hard to approximate* Max-Bi-Clique *within any constant factor.*

As mentioned above, the hardness factor can be boosted to n^δ for some $\delta > 0$ using known techniques. (such as described in [1, 6])

UGC and strong UGC

Unique games conjecture so far helped in understanding the tight inapproximability factors of many problems including, but not limited to, Vertex Cover [14], optimal algorithm for every Max-CSP[16], Ordering CSPs[11], characterizing strong approximation resistance of CSPs[15] etc. The inherent difficulty in showing hardness results assuming Unique Games Conjecture for the problems that we study is that we need some kind of expansion property on the unique games instance which it lacks. It is shown that Unique Games are easy when the constraint graph is an expander[4]. In general, in [3] it is shown that Unique Games are easy when a normalized adjacency matrix of a constraint graph has very few eigenvalues close to 1. So the natural direction is to modify the unique games instance to get some expansion property but weak enough so that it is not tractable by the techniques of [4], [3]. A similar Strong Unique Games Conjecture, which has a *weak expansion property*, has been used earlier in [5] and [17] to show inapproximability results for minimizing weighted completion time on a single machine with precedence constraints and minimizing makespan in precedence constrained scheduling on identical machines respectively. Our result adds another interesting implication of Unique Games Conjecture with weak expansion property, namely inapproximability of Max-Bi-Clique and Bi-Covering. We hope that our results will help motivate study of Strong Unique Games Conjecture and ultimately answering the question about its equivalence to the Unique Games Conjecture.

Algorithmic Results

We give better than 2 approximation for Bi-Covering on numerous special graph classes.

Graph types. A *δ-vertex expander* is a graph so that for every S of size $|S| \leq n/2$, $N_1(S) \geq \delta n$, where $N_1(S)$ is the set of neighbors of S not in S. A *chordal graph* is a graph

that does not contain a cycle of size at least 4 as an induced subgraph. A *split graph* is a graph whose vertex set is a union of a Clique and and independent set, with arbitrarily connections between the clique and the independent set.

A minor of a graph is any subgraph G' that can be derived from G by contracting and removing edges. A *minor free graph* is a graph that does not contain some constant size graph H as a minor.

An *interval graph* is the intersection graph of a family of intervals on the real line. It has one vertex for each interval in the family, and an edge between every pair of vertices corresponding to intervals that intersect.

The algorithmic results can be summarized in the following theorem.

▶ **Theorem 4.** *The* Bi-Covering *problem admits polynomial time algorithms that attain the following ratios (Graph type: approximation ratio):*

1. *Chordal graphs :* 1.876.
2. *Interval Graphs: exact* $O(n^5)$ *time algorithm.*
3. *Minor Free Graph:* $1 + o(1)$.
4. *Graph with minimum degree* δn*:* $2 - 4\delta/3$.
5. δ*-vertex expander Graph:* $2/(1 + \delta^2/8)$.
6. *Split Graphs :* $8/5$.
7. *Graphs with minimum degree* d*:* $2 - (6/5) \cdot 1/d$.

Our algorithms are quite non-trivial. Most of our algorithmic results relies on the fact that if we can find two disjoint sets each of size at least ϵn with no edges in between, then this itself gives $2 - \epsilon$ approximation. To get better bound on ϵ in some special cases we use known theorems related to the structural results of graphs, size of separator, lower bound on independent set size etc. In some of the cases, we create a bipartite graph from a given graph instance and show that the vertex cover in the bipartite graph is small. We then use the bound on the size of vertex cover to find a better bi-covering of the edges in a graph.

3 Organization

In Section 4, we prove the main inapproximability of Bi-Covering and related problems. We refer to the full version of the paper for algorithmic results.

4 Inapproximability of Bi-Covering

The Bi-Covering problem is:

Input: A graph $G(V, E)$
Output: Two subsets $A, B \subseteq V$ such that $A \cup B = V$ and every edge $(u, v) \in E$ either $\{u, v\} \subseteq A$ or $\{u, v\} \subseteq B$. Minimize $\max\{|A|, |B|\}$.

The optimal value of a Bi-Covering on instance $G(V, E)$ is always at least $|V|/2$ and hence getting a 2-approximation for this problem is *trivial* by setting $A = V$ and $B = \emptyset$. In order to beat 2-approximation, one should be able to solve the following weaker problem.

Problem

For small enough $\epsilon > 0$, given an undirected graph $G(V, E)$, distinguish between the following two cases:

1. There exists two disjoint sets $A, B \subseteq V$, $|A|, |B| \geq (1/2 - \epsilon)|V|$ such that there are no edges between A and B.
2. There exists no two disjoint sets $A, B \subseteq V$ $|A|, |B| \geq \epsilon|V|$ such that there are no edges between A and B.

In this section, we show that it is UG-Hard to distinguish between (1) and (2) for any constant $\epsilon > 0$ proving Theorem 1.

4.1 Preliminaries

Let q be any prime for convenience. We are interested in space of functions from $\mathbb{F}_q^n$ to $\mathbb{R}$. Define inner product on this space as $\langle f, g \rangle = \frac{1}{q^n} \sum_{x \in \mathbb{F}_q^n} f(x)g(x)$. Let ω_q be the q^{th} root of unity. For a vector $\alpha \in \mathbb{F}_q^n$, we will denote α_i the i^{th} coordinate of vector α. The *Fourier decomposition* of a function $f : \mathbb{F}_q^n \to \mathbb{R}$ is given as

$$f(x) = \sum_{\alpha \in \mathbb{F}_q^n} \hat{f}(\alpha)\chi_\alpha(x)$$

where $\chi_\alpha(x) := \omega_q^{\langle \alpha, x \rangle}$ and a *Fourier coefficient* $\hat{f}(\alpha) := \langle f, \chi_\alpha \rangle$.

▶ **Definition 5** (Symmetric Markov Operator)**.** Symmetric Markov operator on $\mathbb{F}_q$ can be thought of as a random walk on an undirected graph with the vertex set $\mathbb{F}_q$. It can be represented as a $q \times q$ matrix T where (i, j) th entry is the probability of moving to vertex j from i.

▶ **Definition 6.** For a symmetric Markov operator T, let $1 = \lambda_0 \geq \lambda_1 \geq \lambda_2 \ldots \geq \lambda_{q-1}$ be the eigenvalues of T in a non-increasing order. The spectral radius of T, denoted by $r(T)$, is defined as:

$$r(T) = \max\{|\lambda_1|, |\lambda_{q-1}|\}.$$

For a Markov operator T the condition $r(T) < 1$ is equivalent to saying that the induced regular graph (self-loop allowed) on $\mathbb{F}_q$ is non-bipartite and connected.

For T as above, we also define a Markov operator $T^{\otimes n}$ on $[q]^n$ in a natural way i.e applying a Markov operator $T^{\otimes n}$ to $x \in [q]^n$ is same as applying the Markov operator T on each x_i independently. Note that if T is symmetric then $T^{\otimes n}$ is also symmetric and $r(T^{\otimes n}) = r(T)$.

▶ **Definition 7** (Influence)**.** Let $f : \mathbb{F}_q^n \to \mathbb{R}$ be a function. the influence of the $i'th$ variable on f, denoted by $\mathbf{Inf}_i(f)$ is defines as:

$$\mathbf{Inf}_i(f) = \mathbb{E}[\mathbf{Var}_{x_i}[f(x)|x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]]$$

where $x_1, \ldots, x_n$ are uniformly distributed. In terms of Fourier coefficients, it has the following formula:

$$\mathbf{Inf}_i(f) = \sum_{\alpha_i \neq 0} \hat{f}(\alpha)^2.$$

The low-level (level k) influence of $i'th$ variable is defined as:

$$\mathbf{Inf}_i^{\leq k}(f) = \sum_{\alpha_i \neq 0, |\alpha| \leq k} \hat{f}(\alpha)^2.$$

where $|\alpha|$ is the number of non-zero co-ordinates in α.

We will need the following Gaussian stability measure in our analysis:

▶ **Definition 8.** Let $\phi : \mathbb{R} \to [0,1]$ be the cumulative distribution function of the standard Gaussian random variable. For a parameter $\rho, \mu, \nu \in [0,1]$, we define the following two quantities:

$$\underline{\Gamma}_\rho(\mu,\nu) = \Pr[X \leq \phi^{-1}(\mu), Y \geq \phi^{-1}(1-\nu)]$$

$$\overline{\Gamma}_\rho(\mu,\nu) = \Pr[X \leq \phi^{-1}(\mu), Y \leq \phi^{-1}(\nu)]$$

where X and Y are two standard Gaussian variables with covariance ρ.

We are now ready to state the invariance principle from [7] that we need for our reduction.

▶ **Theorem 9** ([7]). *Let T be a symmetric Markov operator on $\mathbb{F}_q$ such that $\rho = r(T) < 1$. Then for any $\tau > 0$ there exists $\delta > 0$ and $k \in \mathbb{N}$ such that if $f, g : \mathbb{F}_q^n \to [0,1]$ are two functions satisfying*

$$\min(\mathbf{Inf}_i^{\leq k}(f), \mathbf{Inf}_i^{\leq k}(g)) \leq \delta$$

for all $i \in [n]$, then it holds that

$$\langle f, T^{\otimes n} g \rangle \geq \underline{\Gamma}_\rho(\mu,\nu) - \tau$$

where $\mu = \mathbb{E}[f]$, $\nu = \mathbb{E}[g]$.

Our hardness result is based on a variant of Unique Games conjecture. First, we define what the Unique game is:

▶ **Definition 10** (Unique Game). An instance $G = (U, V, E, [L], \{\pi_e\}_{e \in E})$ of the Unique Game constraint satisfaction problem consists of a bi-regular bipartite graph (U, V, E), a set of alphabets $[L]$ and a permutation map $\pi_e : [L] \to [L]$ for every edge $e \in E$. Given a labeling $\ell : U \cup V \to [L]$, , an edge $e = (u, v)$ is said to be satisfied by ℓ if $\pi_e(\ell(v)) = \ell(u)$.

G is said to be *at most δ-satisfiable* if every labeling satisfies at most a δ fraction of the edges.

The following is a conjecture by Khot [12] which has been used to prove many *tight* inapproximability results.

▶ **Conjecture 11** (Unique Games Conjecture [12]). *For every sufficiently small $\delta > 0$ there exists $L \in \mathbb{N}$ such that the following holds. Given a an instance $\mathcal{G} = (U, V, E, [L], \{\pi_e\}_{e \in E})$ of* Unique Game *it is NP-hard to distinguish between the following two cases:*

- *YES case: There exist an assignment that satisfies at least $(1-\delta)$ fraction of the edges.*
- *NO case: Every assignment satisfies at most δ fraction of the edge constraints.*

Our hardness results are based on the following stronger conjecture which is similar to the one in Bansal-Khot [5]. We refer readers to [5] for more discussion on comparison between these two conjectures.

▶ **Conjecture 12** (Strong Unique Games Conjecture). *For every sufficiently small $\delta, \gamma, \eta > 0$ there exists $L \in \mathbb{N}$ such that the following holds: Given an instance $\mathcal{G} = (U, V, E, [L], \{\pi_e\}_{e \in E})$ of* Unique Game *which is bi-regular, it is* NP*-hard to distinguish between the following two cases:*

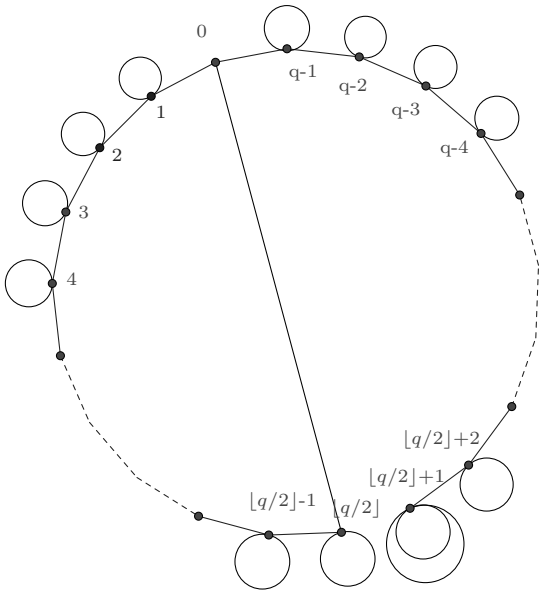


Figure 1 Gadget.

- *YES case: There exist sets $V' \subseteq V$ such that $|V'| \geq (1-\eta)|V|$ and an assignment that satisfies all edges connected to V'.*
- *NO case: Every assignment satisfies at most γ fraction of the edge constraints. Moreover, the instance satisfies the following expansion property. For every set $S \subseteq V$, $|S| = \delta|V|$, we have $|\Gamma(S)| \geq (1-\delta)|U|$, where $\Gamma(S) := \{u \in U \mid \exists v \in S \mathtt{s.t.} (u,v) \in E\}$.*

▶ Remark. We would like to point out that the above conjecture differs from the one in [5] in the completeness case. In [5], the Yes instance has a guarantee that there exists sets $V' \subseteq V, U' \subseteq U$ with $|V'| \geq (1-\eta)|V|, |U'| \geq (1-\eta)|U|$ such that all edges between V' and U' are satisfied.

4.2 $(2-\epsilon)$-inapproximability

In order to prove the $(2-\epsilon)$ hardness, we first start with a *dictatorship test* that we will use as a gadget in the actual reduction.

4.2.1 Dictatorship Test

We design a dictatorship test for the problem Bi-Covering. We are interested in functions $f : \mathbb{F}_q^n \to \mathbb{R}$. f is called a dictator if it is of the form $f(x_1, x_2, \ldots, x_n) = x_i$ for some $i \in [n]$.

4.2.1.1 Dictatorship gadget

For convenience, we will let $q > 2$ be any prime number for the description of the dictatorship gadget. Let $G(\mathbb{F}_q, \mathcal{E})$ be a 3-regular graph on $\mathbb{F}_q$ (where we identify the elements of $\mathbb{F}_q$ by $\{0, 1, \ldots, q-1\}$) with self loops as shown in figure 1:

It is constructed as follows : Take a cycle on $0, 1, 2, \ldots, q-1, 0$, then add a self loop to every vertex except to the vertex 0. Remove the edge $(\lfloor q/2 \rfloor, \lfloor q/2 \rfloor + 1)$, add an edge

$(0, \lfloor q/2 \rfloor)$. Finally, to make it 3-regular, add a self loop to the vertex $\lfloor q/2 \rfloor + 1$. This completes the description of graph G. Since the graph G is connected and non-bipartite, the symmetric Markov operator T defined by the random walk in G has $r(T) < 1$. One crucial thing about G is that it has two large disjoint subsets of vertices, namely $\{1, 2, \ldots, \lfloor q/2 \rfloor\}$ and $\{\lfloor q/2 \rfloor + 1 . \lfloor q/2 \rfloor + 2, \ldots, q-1\}$, with no edges in between.

Consider the vertex set $V = \mathbb{F}_q^R$ for some constant R. We will construct a graph H on V as follows : $(x, y) \in (\mathbb{F}_q^R)^2$ forms an edge in H iff they satisfy the following condition:

$$\forall i \in [R], (x_i, y_i) \in \mathcal{E},$$

x is adjacent to y iff $T^{\otimes R}(x \leftrightarrow y) \neq 0$.

4.2.1.2 Completeness

Let $f : \mathbb{F}_q^R \to \mathbb{R}$ be any dictator, say i^{th} dictator i.e. $f(x) = x_i$. By letting set A to be $f^{-1}(0) \cup f^{-1}(1) \cup \ldots \cup f^{-1}(\lfloor q/2 \rfloor)$ and set B to be $f^{-1}(0) \cup f^{-1}(\lfloor q/2 \rfloor + 1) \cup f^{-1}(\lfloor q/2 \rfloor + 2) \cup \ldots \cup f^{-1}(q-1)$, it can be seen easily that there is no edge between sets $A \setminus B$ and $B \setminus A$. More precisely,

$$\begin{aligned} A \setminus B &= \{x \in \mathbb{F}_q^R \quad | \quad x_i \in \{1, 2, \ldots, \lfloor q/2 \rfloor\}\} \\ B \setminus A &= \{y \in \mathbb{F}_q^R \quad | \quad y_i \in \{\lfloor q/2 \rfloor + 1, \lfloor q/2 \rfloor + 2, \ldots, q-1\}\} \end{aligned}$$

By the property of Markov operator $T^{\otimes R}$, (x, y) are not adjacent if $(x_i, y_i) \notin \mathcal{E}$ for some $i \in [R]$. Hence, there are no edges between $A \setminus B$ and $B \setminus A$. Thus, the optimal value is at most

$$\frac{1}{|V|} \cdot \max\{|A|, |B|\} = \frac{1}{2} + \frac{1}{2q}.$$

4.2.1.3 Soundness

Let $A, B \subseteq V$ such that $A \cup B = V$ and $f, g : \mathbb{F}_q^R \to \{0, 1\}$ be the indicator functions of sets $A \setminus B$ and $B \setminus A$ respectively. Suppose $|A \setminus B| = \epsilon |V|$ and $|B \setminus A| = \epsilon |V|$ for some $\epsilon > 0$ and that there are no edges in between $A \setminus B$ and $B \setminus A$. We will show that in this case, f and g must have a common influential co-ordinate. Since, there are no edges between these sets, we have

$$\mathop{\mathbb{E}}_{\substack{x \sim \mathbb{F}_q^R, \\ y \sim T^{\otimes R}(x)}} [f(x) g(y)] = \langle f, T^{\otimes R} g \rangle = 0 \,.$$

For the application of Invariance principle, Theorem 9, in our case we have $\mathbb{E}[f] = \mathbb{E}[g] = \epsilon > 0$ and $\rho = r(T) < 1$. Thus, for small enough $\tau := \tau(\rho, \epsilon) > 0$,

$$\underline{\Gamma}_\rho(\epsilon, \epsilon) - \tau > 0.$$

We can now apply Theorem 9 to conclude that there exists $i \in [R]$ and $k \in \mathbb{N}$ independent of R such that

$$\min(\mathbf{Inf}_i^{\leq k}(f), \mathbf{Inf}_i^{\leq k}(g)) \geq \delta,$$

for some $\delta(\tau) > 0$. Hence, unless f and g have a common influential co-ordinate, $\frac{1}{|V|} \cdot \max\{|A|, |B|\} \geq 1 - \epsilon$. Thus, the optimum value is at least $1 - \epsilon$

4.2.2 Actual Reduction

The above dictatorship test for large enough q can be composed with the Unique Games instance having some stronger guarantee (Conjecture 12) in a straightforward way that gives $(2-\epsilon)$ hardness for every constant $\epsilon > 0$ assuming UGC. Details as follows:

Let $\mathcal{G} = (U, V, E, [L], \{\pi_e\}_{e\in E})$ be the given instance of Unique Game with parameters $\delta < \frac{\epsilon}{4}, \gamma, \eta > 0$ from Conjecture 12 . We replace each vertex $v \in V$ by a block of q^L vertices, namely by a hypercube $[q]^L$. We will denote this block by $[v]$. As defined in the dictatorship test, let G be the graph on $\mathbb{F}_q$ and T be the induced symmetric Markov operator. For every pair of edges $e_1(u, v_1)$ and $e_2(u, v_2)$ in $\mathcal{G}$, we will add the following edges between $[v_1]$ and $[v_2]$: Let π_1 and π_2 be the permutation constraint associated with e_1 and e_2 respectively. $x \in [v_1]$ and $y \in [v_2]$ are connected by an edge iff $T^{\otimes L}((x \circ \pi_1^{-1}) \leftrightarrow (y \circ \pi_2^{-1})) \neq 0$ (where $(x \circ \pi^{-1})_i = x_{\pi^{-1}(i)}$ for all $i \in [L]$) i.e. for every $i \in [L]$, $x_{\pi_1^{-1}(i)}$ and $y_{\pi_2^{-1}(i)}$ are connected by an edge in graph G. This completes the description of a graph. Let's denote this graph by H.

▶ **Lemma 13** (Completeness). *If there exists an assignment to vertices in $\mathcal{G}$ that satisfies all edges connected to $(1-\eta)$ fraction of vertices in V then H has a* Bi-Covering *of size at most $(1-\eta)(1/2 + 1/2q) + \eta$.*

Proof. Fix a labeling ℓ such that for at least $(1-\eta)$ fraction of vertices in V in $\mathcal{G}$, all edges attached to them are satisfied. Suppose X be the set of remaining η fraction of vertices of V in $\mathcal{G}$. For every vertex $v \in V$, consider the following two partitions of $[v]$:

$$A_v = \{x \in [q]^L : x_{\ell(v)} \in \{1, \ldots, \lfloor q/2 \rfloor\}\}$$
$$B_v = \{x \in [q]^L : x_{\ell(v)} \in \{\lfloor q/2 \rfloor\} + 1, \lfloor q/2 \rfloor\} + 2, \ldots, q\}\}$$
$$C_v = \{x \in [q]^L : x_{\ell(v)} = 0\}$$

Let $A = \cup_{v\in V}(A_v \cup C_v) \cup_{z\in X} [z]$ and $B = \cup_{v\in V}(B_v \cup C_v) \cup_{z\in X} [z]$. The claim is that this is the required edge separating sets. To see this, consider any vertex pair (a, b) such that $a \in A \setminus B$ and $b \in B \setminus A$. We need to show that (a, b) must not be adjacent in H. Suppose $a \in [v_1]$ and $b \in [v_2]$. If v_1 and v_2 don't have a common neighbor then clearly, there is no edge between a and b. Suppose they have a common neighbor u and let $e_1 = (u, v_1)$ and $e_2 = (u, v_2)$ be the edges and π_1 and π_2 be the associated permutation constraints. Since $X \subseteq A \cap B$, $v_1, v_2 \notin X$. Hence ℓ satisfies all constraints associated with v_1 and v_2. In particular, $\pi_1(\ell(v_1)) = \pi_2(\ell(v_2)) =: j$ for some $j \in [L]$. Since $a \in A_{v_1}$, we have $a_{\pi_1^{-1}(j)} = a_{\ell(v_1)} \in \{1, \ldots, \lfloor q/2 \rfloor\}\}$. Similarly, $b_{\pi_2^{-1}(j)} \in \{\lfloor q/2 \rfloor\} + 1, \lfloor q/2 \rfloor\} + 2, \ldots, q\}$. By the construction of edges in H, a and b are not adjacent.

For any v, $|A_v \cup C_v| = |B_v \cup C_v| = (\frac{1}{2} + \frac{1}{2q})q^L$. Thus,

$$|A| = |B| \leq \left(\eta + (1-\eta)\left(\frac{1}{2} + \frac{1}{2q}\right)\right)|V|q^L\,. \qquad \blacktriangleleft$$

▶ **Lemma 14** (Soundness). *For every constant $\epsilon > 0$, there exists a constant γ such that, if $\mathcal{G}$ is at most γ-satisfiable then H has* Bi-Covering *of size at least $1-\epsilon$.*

Proof. Suppose for contradiction, there exists an Bi-Covering of size at most $(1-\epsilon)$. This means there exists two disjoint sets X, Y of size at least ϵ fraction of vertices in H such that there are no edges in between X and Y. Let X^* be the set of vertices in $v \in V$ such that $[v] \cap X \geq \frac{\epsilon}{2}|[v]|$. Similarly, Y^* be the set of vertices in $v \in V$ such that $[v] \cap Y \geq \frac{\epsilon}{2}|[v]|$. By simple averaging argument, $|X^*| \geq \frac{\epsilon}{2}|V|$ and $|Y^*| \geq \frac{\epsilon}{2}|V|$.

▶ **Lemma 15.** *The total fraction of edges connected to X^* whose other end point is in $\Gamma(X^*) \cap \Gamma(Y^*)$ is at least $\frac{1}{2}$.*

Proof. Let $\mathcal{G}$ has left-degree d_1 and right-degree d_2. We have $d_1 = \frac{d_2|V|}{|U|}$. Suppose the claim is not true, then at least $\frac{1}{2}$ fraction of edges have there endpoint in $U \setminus \Gamma(Y^*)$. As, $|U \setminus \Gamma(Y^*)| \leq \delta|U|$, the average degree of a vertex in $U \setminus \Gamma(Y^*)$ is at least $\frac{(1/2)d_2|X^*|}{\delta|U|} \geq \frac{(d_2/2)\cdot(\epsilon/2)|V|}{\delta|U|}$ which is greater than d_1 as $\epsilon > 4\delta$. ◀

For $v \in X^* \cup Y^*$, let $f_v : [q]^L \to \{0,1\}$ be the indicator function of a set $[v] \cap (X \cup Y)$. Define the following label set for $v \in X^* \cup Y^*$ for some $\tau' > 0$ and $k \in \mathbb{N}$:

$$\mathcal{F}(v) := \{i \in [L] \mid \mathbf{Inf}_i^{\leq k}(f_v) \geq \tau'\}.$$

We have $|\mathcal{F}(v)| \leq \frac{\tau'}{k}$ as $\sum_i \mathbf{Inf}_i^{\leq k}(f_v) \leq k$.

▶ **Lemma 16.** *There exists a constant $\tau' := \tau'(q,\epsilon)$ and $k := k(q,\epsilon)$ such that for every $u \in U$ and edges $e_1(u,v), e_2(u,w)$ such that $v \in X^*$ and $w \in Y^*$, we have*

$$\pi_{e_1}(\mathcal{F}(v)) \cap \pi_{e_2}(\mathcal{F}(w)) \neq \emptyset.$$

Proof. As there are no edges between X and Y, we have

$$\mathop{\mathbb{E}}_{\substack{(x\circ\pi_{e_1}^{-1})\sim\mathbb{F}_q^L,\\ (y\circ\pi_{e_2}^{-1})\sim T^{\otimes L}(x\circ\pi_{e_1}^{-1})}} [f_v(x\circ\pi_{e_1}^{-1}) f_w(y\circ\pi_{e_2}^{-1})] = 0.$$

By the soundness analysis of the dictatorship test, it follows that there exists $i \in [L]$ such that

$$\min(\mathbf{Inf}^{\leq k}_{\pi_{e_1}^{-1}(i)}(f_v), \mathbf{Inf}^{\leq k}_{\pi_{e_2}^{-1}(i)}(f_w)) \geq \tau',$$

for some τ', k as a function of q and ϵ. Thus, $i \in \pi_{e_1}(\mathcal{F}(v))$ and $i \in \pi_{e_2}(\mathcal{F}(w))$. ◀

◀

4.2.2.1 Labeling

Fix τ' and k from Lemma 16. We now define a labeling ℓ to vertices in $X^* \subseteq V$ and in $\Gamma(X^*) \cap \Gamma(Y^*) \subseteq U$ as follows: For a vertex $v \in X^*$ set $\ell(v)$ to be an uniformly random label from $\mathcal{F}(v)$. For $u \in \Gamma(X^*) \cap \Gamma(Y^*)$, select an arbitrary neighbor w of u in Y^* and set $\ell(u)$ to be an uniformly random label from the set $\pi_{(u,w)}(\mathcal{F}(w))$ of size at most $\frac{k}{\tau'}$. Fix an edge (u,v) such that $u \in \Gamma(X^*) \cap \Gamma(Y^*)$ and $v \in X^*$. By Lemma 16, for any $w \in Y^*$ since $\pi_{(u,w)}(\mathcal{F}(w)) \cap \pi_{(u,v)}(\mathcal{F}(v)) \neq \emptyset$, The probability that the edge is satisfied by the randomized labeling is at least $\left(\frac{\tau'}{k}\right)^2$. Thus in expectation, at least $\left(\frac{\tau'}{k}\right)^2$ fraction of edges between X^* and $\Gamma(X^*) \cap \Gamma(Y^*)$ are satisfied. By Lemma 15, at least $\frac{1}{2}$ fraction of edges connected to X^* are in between X^* and $\Gamma(X^*) \cap \Gamma(Y^*)$. Finally using bi-regularity, this labeling satisfies at least $\frac{1}{2}\frac{\epsilon}{2}\left(\frac{\tau'}{k}\right)^2$ fraction of edges in $\mathcal{G}$. Setting $\gamma < \frac{1}{2}\frac{\epsilon}{2}\left(\frac{\tau'}{k}\right)^2$ completes the proof. ◀

Proof of Theorem 1

The proof follows from Lemma 13, Lemma 14 and Conjecture 12.

Proof of Theorem 2

Given an input as a bipartite graph, there is a trivial 3/2 approximation for Bi-Covering - Take set A to be the union of a smaller part and half of the larger bi partition and B to be union of smaller part and remaining half of the larger part. It is easy to see these two sets A and B satisfy the property of being a Bi-Covering. As $\max\{|A|, |B|\} \leq \frac{3}{4}|V|$, this is a $\frac{3}{2}$ approximation as OPT is at least $\frac{|V|}{2}$.

The $\frac{3}{2} + \epsilon$ inapproximability follows easily from the above $(2 - \epsilon)$ inapproximability for the general case. The reduction is as follows: Let $G(V, E)$ be the given instance of a Bi-Covering. Construct a natural bipartite graph G' between $V \times V$ where (i, j) forms an edge if $(i, j) \in E$ (or $(j, i) \in E$). Fix a small enough constant $\epsilon > 0$. It is easy to see that if G has a solution of fractional size $1/2 + \epsilon$ then so does G'. Next, if there are sets A' and B' where $\frac{1}{2|V|}\max\{|A'|, |B'|\} \leq \frac{3}{4} - \epsilon$ which satisfy the Bi-Covering property, we have $\frac{1}{2|V|}|A' \setminus B'| = \frac{1}{2|V|}(2|V| - |B'|) \geq 1 - (\frac{3}{4} - \epsilon) = \frac{1}{4} + \epsilon$ and similarly $\frac{1}{2|V|}|B' \setminus A'| \geq \frac{1}{4} + \epsilon$. Note that $A' \setminus B'$ and $B' \setminus A'$ are two disjoint sets whose size of union is at least $(1 + 2\epsilon)|V|$. Thus, we can find two sets, say X' and Y' (namely X' is intersection of $A' \setminus B'$ with left part of the bipartite graph and Y' is the intersection of $B' \setminus A'$ with right part) of size at least $\epsilon|V|$ each, where X' is from left side and Y' is from right side with no edges in between. We now think of X' and Y' as a subset of V. Let $Z = X' \cap Y'$. Partition Z into Z_1 and Z_2 of equal sizes. Take $X = Z_1 \cup (X' \setminus Y')$ and $Y = Z_2 \cup (Y' \setminus X')$. It is now easy to verify that there are no edges in between X and Y in G and $\frac{1}{|V|}\min\{|X|, |Y|\} \geq \frac{\epsilon}{2}$. Hence, if we can find a solution of fractional cost $\frac{3}{4} - \epsilon$ in G' in polynomial time then we can also find a solution of fractional cost $1 - \frac{\epsilon}{2}$ in G in polynomial time and this gives a polynomial time algorithm with approximation factor $2 - \frac{\epsilon}{2}$ for small enough constant $\epsilon > 0$. As Bi-Covering is UG hard to approximate within $(2 - \epsilon)$ for all $\epsilon > 0$ for general graph, this gives a $\frac{3}{2} + \epsilon$ hardness for Bi-Covering in bipartite graph.

Proof of Corollary 3

We prove it by giving reduction from Bi-Covering. Let $G(V, E)$ be the given instance of Bi-Covering. Construct a bipartite graph H between $V \times V$ where (i, j) forms an edge if $(i, j) \notin E$. Fix a small enough constant $\epsilon > 0$. In one direction, if G has a Bi-Covering of fractional size at most $(1/2 + \epsilon)$ then H' contains a $(1/2 - \epsilon)|V| \times (1/2 - \epsilon)|V|$ bipartite clique. In other direction, if H' has a bipartite clique of size $2\epsilon|V| \times 2\epsilon|V|$ then let X' and Y' be the subset of vertices from left and right side of bipartite clique. As before, let $Z = X' \cap Y'$ and Z_1 and Z_2 be the partition of Z of equal size. Let $X = (X' \setminus Y') \cup Z_1$ and $Y = (Y' \setminus X') \cup Z_2$. It follows that $|X|, |Y|$ is at least $\epsilon|V|$ and are disjoint viewed as a subset of V. Also, there are no edges between X and Y. Therefore, $V \setminus X$ and $V \setminus Y$ each of size at most $(1 - \epsilon)|V|$ gives a Bi-Covering of G. Thus, Theorem 1 implies that it is hard to distinguish between Bi−Clique of size $(1/2 - \epsilon)|V|$ and $\epsilon|V|$ which completes the proof of corollary.

Acknowledgements. We would like to thank anonymous reviewers for their comments which significantly helped in improving the presentation of this paper.

References

1 Noga Alon, Uriel Feige, Avi Wigderson, and David Zuckerman. Derandomized graph products. *Computational Complexity*, 5(1):60–75, 1995.

2 Christoph Ambühl, Monaldo Mastrolilli, and Ola Svensson. Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM Journal on Computing*, 40(2):567–596, 2011.

3 Sanjeev Arora, Boaz Barak, and David Steurer. Subexponential algorithms for unique games and related problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 563–572. IEEE, 2010.

4 Sanjeev Arora, Subhash A Khot, Alexandra Kolla, David Steurer, Madhur Tulsiani, and Nisheeth K Vishnoi. Unique games on expanding constraint graphs are easy. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 21–28. ACM, 2008.

5 Nikhil Bansal and Subhash Khot. Optimal long code test with one free bit. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 453–462. IEEE, 2009.

6 Thang Nguyen Bui and Lisa C. Strite. An ant system algorithm for graph bisection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO'02, pages 43–51, 2002.

7 Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. *SIAM Journal on Computing*, 39(3):843–873, 2009.

8 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 534–543. ACM, 2002.

9 Uriel Feige and Shimon Kogan. *Hardness of approximation of the balanced complete bipartite subgraph problem*, 2004.

10 Rajiv Gandhi, Samir Khuller, Aravind Srinivasan, and Nan Wang. Approximation algorithms for channel allocation problems in broadcast networks. *Networks*, 47(4):225–236, 2006.

11 Venkatesan Guruswami, Johan Håstad, Rajsekar Manokaran, Prasad Raghavendra, and Moses Charikar. Beating the random ordering is hard: Every ordering csp is approximation resistant. *SIAM Journal on Computing*, 40(3):878–914, 2011.

12 Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 767–775. ACM, 2002.

13 Subhash Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

14 Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2-ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

15 Subhash Khot, Madhur Tulsiani, and Pratik Worah. A characterization of strong approximation resistance. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 634–643. ACM, 2014.

16 Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 245–254. ACM, 2008.

17 Ola Svensson. Conditional hardness of precedence constrained scheduling on identical machines. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 745–754. ACM, 2010.

Constant Congestion Routing of Symmetric Demands in Planar Directed Graphs

Chandra Chekuri[*1], **Alina Ene**[2], **and Marcin Pilipczuk**[†3]

1 **Department of Computer Science, University of Illinois, Urbana-Champaign, USA**
chekuri@illinois.edu

2 **Department of Computer Science and DIMAP, University of Warwick, Warwick, UK**
A.Ene@dcs.warwick.ac.uk

3 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
malcin@mimuw.edu.pl

Abstract

We study the problem of routing symmetric demand pairs in planar digraphs. The input consists of a directed planar graph $G = (V, E)$ and a collection of k source-destination pairs $\mathcal{M} = \{s_1t_1, \ldots, s_kt_k\}$. The goal is to maximize the number of pairs that are routed along disjoint paths. A pair s_it_i is routed in the symmetric setting if there is a directed path connecting s_i to t_i *and* a directed path connecting t_i to s_i. In this paper we obtain a randomized poly-logarithmic approximation with constant congestion for this problem in planar digraphs. The main technical contribution is to show that a planar digraph with directed treewidth h contains a constant congestion crossbar of size $\Omega(h/\text{polylog}(h))$.

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

Keywords and phrases Disjoint paths, symmetric demands, planar directed graph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.7

1 Introduction

Disjoint path problems are well-studied routing problems with several applications and fundamental connections to algorithmic and structural results in combinatorial optimization and graph theory. Canonical problems here are the edge-disjoint paths problem (EDP) and the node-disjoint paths problem (NDP) in undirected graphs. In both these problems the input consists of an undirected graph $G = (V, E)$ and k node-pairs $\{s_1t_1, \ldots, s_kt_k\}$. In EDP the goal is to connect the pairs by edge-disjoint paths and in NDP the goal is to connect the pairs by node-disjoint paths. The decision versions of these problems are NP-Complete when k is part of the input. The seminal work of Robertson and Seymour showed that both these problems are fixed parameter tractable when parameterized by k, the number of pairs. In this paper we are concerned with an optimization version of the problems where the goal is to maximize the number of input pairs that can be routed via edge or node-disjoint

* Supported in part by NSF grant CCF-1319376.
† Research done while the author was at University of Warwick, partially supported by DIMAP and by Warwick-QMUL Alliance in Advances in Discrete Mathematics and its Applications.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 7; pp. 7:1–7:14

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

paths. To avoid notational overload we will henceforth use EDP and NDP to refer to these maximization versions.

The approximability of EDP and NDP has been extensively studied but our understanding is still limited. The best known approximation for both these problems is $O(\sqrt{n})$ [9, 31] (here n is number of nodes in G) while current hardness of approximation results only rule out an $O(\log^{1/2-\varepsilon} n)$ approximation [2]. Even in planar graphs the best approximation up to very recently was $O(\sqrt{n})$, with a slight improvement just announced [16]. One of the reasons for this state of affairs is that the natural multicommodity flow relaxation has an integrality gap of $\Theta(\sqrt{n})$. On the other hand, two closely related relaxations of these problems have seen significant progress in the last decade. ANF is the relaxation of the disjoint paths problem where a subset of the input pairs $\mathcal{M}'$ is routed if there is a feasible multicommodity flow in the graph that routes one unit of flow for each pair in $\mathcal{M}'$. A second relaxation is to allow some small constant congestion c, i.e., instead of the pairs being routed on disjoint paths we allow up to c paths to use a given edge or node. ANF admits a poly-logarithmic approximation [11, 8]. A series of breakthroughs [33, 1, 13] culminated in a poly-logarithmic approximation for EDP with congestion 2 by Chuzhoy and Li [17]. These ideas have been extended to NDP as well [6, 4]. These results have been made possible by a number of non-trivial ideas and techniques at the intersection of algorithms, combinatorial optimization and graph theory. In particular, the results have been enabled by and contributed to a deeper understanding of the structure of undirected graphs via the notion of treewidth. Treewidth is a well-known graph parameter that plays a fundamental role in the graph-minor theory of Robertson and Seymour; see [3, 4, 5, 14] for some of the recent results.

It is natural to study disjoint paths problems also in *directed* graphs. Here the graph G is directed and the input pairs $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ are ordered and we seek to find a maximum cardinality subset of $\mathcal{M}$ that can be connected by disjoint paths[1]. Unfortunately, it has been shown that disjoint paths problems are highly intractable in directed graphs. It is known that even the simpler case of ANF and with congestion c allowed is hard to approximate to within a factor of $n^{\Omega(1/c)}$ [15]; moreover this holds in acyclic graphs.

A recent paper by a subset of the authors [7] initiated the study of maximum throughput routing problems in directed graphs where the demand pairs are *symmetric*. Here the graph G is directed but the input pairs are unordered as in the undirected setting. Routing a pair s_it_i requires finding a path that connects s_i to t_i *and* a path connecting t_i to s_i. We use Sym-Dir-EDP, Sym-Dir-NDP and Sym-Dir-ANF to denote the analogues of EDP, NDP and ANF respectively in this setting. A detailed motivation for the study of this model is given in [7]. Here we briefly outline some of the key points.

The model is motivated by both theoretical and practical considerations. On the theoretical side the model generalizes (modulo constant congestion) the edge and node disjoint paths problems in undirected graphs. Moreover, flow-cut gaps in this model have been studied in the past and have close connections to various problems including feedback edge/vertex set problems [30, 37, 21, 10]. From the more practical side there are several scenarios where the communication between users is symmetric while the underlying network that supports the communication may be asymmetric (hence modeled as a directed graph); see [26, 25] for instance.

Unlike the case of directed graph routing problems, the symmetric model exhibits tractability. In particular, the well-linked decomposition framework for undirected graphs extends to a large extent to this model [7].

[1] Although edge and node disjoint paths problems are equivalent in general directed graphs, this is not necessarily the case in restricted graph classes such as planar graphs.

To resolve the complexity of disjoint path problems in the symmetric model one needs to understand the structure of directed graphs as a function of their *directed treewidth* [23, 34], that we denote by $\mathrm{dtw}(G)$. As we mentioned, the interplay between algorithmic questions and graph structure theory for undirected graphs has been very successful in the recent past. There has been recent significant progress on the graph theoretic side on directed treewidth; in particular Kawarabayashi and Kreutzer recently established the excluded grid theorem in directed graphs [27, 28].

The main technical contribution of [7] is to generalize the well-linked decomposition framework of [8] to the symmetric demands setting in directed graphs. As a consequence, [7] obtained a poly-logarithmic approximation with constant congestion for Sym-Dir-ANF. The central open question they raised is the following: *Is there a poly-logarithmic approximation for* Sym-Dir-NDP *with constant congestion in general directed graphs?* It was shown in [7] that this can be answered in the positive by addressing the following question which is the analogue that was raised in [8] for undirected graphs: *If a directed graph G has directed treewidth h, does it have a constant congestion routing structure (crossbar) of size $\Omega(h/\mathrm{polylog}(h))$?* Note that grid-minor theorems establish such a connection between treewidth and routing structures, however, the quantitative relationship between the treewidth and the size of the grid is too weak to prove any meaningful approximation for the routing problem. On the other hand, the routing problem has the flexibility of allowing a large constant congestion which enables one to prove the existence of routing structures that are not as rigid as a grid; this relaxation has been the key to algorithmic success on routing. We also note that it is NP-Complete to decide whether a single pair can be routed without congestion in the symmetric setting [22]; thus a congestion of at least 2 is necessary for a non-trivial approximation ratio.

In this paper we take a step towards the general problem by addressing the important special case of planar graphs. Our main algorithmic result is the following.

▶ **Theorem 1.** *There is a randomized poly-logarithmic approximation for* Sym-Dir-NDP *in planar directed graphs with congestion* 5.

The approximation algorithm in the preceding theorem is derived via a natural multicommodity flow relaxation for the problem. The main new technical ingredient is a graph theoretic result that shows that if a planar digraph has directed treewidth h then it has a constant congestion crossbar of size $\Omega(h/\mathrm{polylog}(h))$. We remark that an undirected planar graph with treewidth h has a grid-minor (which is a congestion 2 crossbar) of size $\Omega(h)$. In contrast the known relationship between treewidth and grid-minors in directed planar graphs is much weaker; recent work [27, 28] only shows that there is a directed-grid of size $f(h)$ for some weakly growing function of h. We hope that our crossbar result could be used as a starting point to improve the quantitative bound on the grid-minor theorem for planar digraphs.

1.1 Overview of the Algorithm and Technical Contributions

Here we give a brief outline of the high-level details of the algorithm and some of our technical contributions. Let $(G, \mathcal{M})$ be an instance of Sym-Dir-NDP, where $G = (V, E)$ is a directed planar graph with unit node capacities, and $\mathcal{M} = \{s_1t_1, \ldots, s_kt_k\}$ is a collection of source-destination pairs. We refer to the nodes participating in $\mathcal{M}$ as terminals, and we use $\mathcal{T}$ to denote the set of terminals. It is convenient to assume that the pairs $\mathcal{M}$ form a matching on $\mathcal{T}$.

Well-linked sets: A key notion that we make use of is well-linkedness. Given a directed graph $G = (V, E)$ a subset of nodes $X \subseteq V$ is said to be well-linked if for any two disjoint subsets Y and Z of X of equal size, there exist $|Y|$ node-disjoint paths from Y to Z; note that the definition is symmetric since we can swap Y and Z. We need a relaxation of well-linkedness. For some parameter $\beta \in [0, 1]$, X is β-well-linked if for all disjoint $Y, Z \subset X$ of equal size there are $|Y|$ paths from Y to Z such that no node is in more than $\lceil 1/\beta \rceil$ of these paths; in other words, the node-congestion caused by the paths is at most $\lceil 1/\beta \rceil$. The case $\beta = 1$ corresponds to well-linkedness. It is well-known that in both directed and undirected graphs well-linkedness is closely connected to treewidth. More precisely, a graph has treewidth k iff it has a well-linked set of size $\Theta(k)$; see [34]. Moreover, if X is β-well-linked in G then the treewidth of G is $\Omega(\beta|X|)$.

Algorithm: Here we outline the high-level steps of our algorithm.

1. Solve a multicommodity flow based LP relaxation that routes each pair $s_i t_i$ fractionally to an amount $x_i \in [0, 1]$ to maximize $\sum_{i=1}^{k} x_i$. See Fig. 2 and the description in Section 2.
2. Use the LP relaxation and the well-linked decomposition framework from [7] to reduce the problem, at the loss of a poly-logarithmic factor in the approximation, to instances in which the terminals $\mathcal{T}$ are α-well-linked for some *fixed* constant α.
3. Assuming that $\mathcal{T}$ is α-well-linked in G we have $\mathrm{dtw}(G) = \Omega(k)$ where $k = |\mathcal{T}|$. Using this fact show that G has a large routing structure and use this structure to route a large number of terminal pairs. Use the following steps.
   a. From G obtain an *Eulerian multigraph* $H = (V, E_H)$ whose support is a subgraph of G such that (i) $\mathcal{T}$ is α'-well-linked in H for $\alpha' = \Omega(\frac{1}{\mathrm{polylog}(k)})$ and (ii) $\Delta(H)$, the maximum degree in H, is $\mathrm{polylog}(k)$.
   b. Using the fact that H is Eulerian, has treewidth $\Omega(k/\mathrm{polylog}(k))$, and has maximum degree $\mathrm{polylog}(k)$, show that it has a cylinder-like routing structure of size $\Omega(k/\mathrm{polylog}(k))$. See Fig. 1.
   c. Route terminals to the routing structure and use it to connect a large number of input pairs.

The preceding algorithm follows the general framework that has been very successful in the undirected graph setting in the recent past. The first two steps follows the well-linked decomposition framework from [8] that has been extended to the symmetric demand instances in directed graphs by [7]. This framework allows one to reduce, via the LP relaxation, general instances to instances in which the terminals are well-linked. This incurs a poly-logarithmic factor loss in the approximation. With this reduction in place we have the following property for our instance. The graph G has a terminal set $\mathcal{T}$ of size h and since $\mathcal{T}$ is α-well-linked for some fixed constant α, G has directed treewidth $\Omega(h)$. Now, the remaining task is to show a graph-theoretic result that any directed graph with treewidth h has a constant congestion *crossbar* routing structure of size $\Omega(h/\mathrm{polylog}(h))$. By crossbar we mean a directed graph H with an *interface* $I \subset V(H)$ with the following property: *any* matching on I can be routed in a symmetric fashion in H with constant congestion. The idea then is to route the terminals to the interface of the crossbar and use it to route the desired matching on the terminals.

In undirected planar graphs if G has treewidth h then it has grid-minor of size $\Omega(h)$ [36], and this grid-minor can be used as a crossbar to route $\Omega(h)$ input pairs (see [8] for instance). What about directed graphs? Johnson *et al.* [23], who introduced the notion of directed treewith, conjectured that any directed graph with sufficiently large treewidth contains a cylindrical grid (see Fig. 1) as a butterfly minor. The cylindrical grid can be used as a

crossbar. In an unpublished manuscript, Johnson *et al.* [24] outlined a proof for the case of planar graphs. Kawarabayashi and Kreutzer [27] recently gave a different proof for the planar and minor-free case, and very recently gave a proof for all graphs [28]. However, as we already mentioned, the quantitative relationship between the size of the cylindrical grid and treewidth is very weak. Hence, these results would not yield any meaningful results for our routing problem. Here, we build on the high-level ideas in the work of Johnson *et al.* [24] to establish our main result which gives a constant congestion crossbar of size $\Omega(h/\text{polylog}(h))$ where h is the treewidth of G; our result applies only to planar graphs and establishing a similar result for general graphs is a challenging open problem. Due to space constraints we only mention the key steps.

A key insight from [24] is that given directed graph G one can create an Eulerian multigraph H of bounded degree whose support is a subgraph of G such that $\text{dtw}(H) \geq f(\text{dtw}(G))$ for some function f. Eulerianness as well as small degree are critical for further manipulations. Our first contribution is to show that H can be chosen such that (i) $\text{dtw}(H) = \text{dtw}(G)/\text{polylog}(\text{dtw}(G))$ and (ii) the maximum degree in H, $\Delta(H) = O(\log^2 \text{dtw}(G))$. For this purpose we use Louis's extension of the cut-matching game of Khandekar, Rao and Vazirani [29] to directed graphs [32], combined with the well-linked decomposition framework of [7, 8].

▶ **Theorem 2.** *Suppose that there is a polynomial time algorithm for $\Omega(1)$-node-well-linked instances of* Sym-Dir-NDP *in planar directed Eulerian graphs of maximum degree Δ that achieves a $\beta(\Delta)$-approximation with congestion c. Then there is a polynomial time randomized algorithm that, with high probability, achieves a $\beta(O(\log^2 k)) \cdot O(\log^6 k)$ approximation with congestion c for arbitrary instances of* Sym-Dir-NDP *in planar directed graphs, where k is the number of pairs in the instance.*

Another key insight from [24] is to consider the undirected version of G, denoted by G^{UN}, to obtain a large undirected grid-minor using the fact that $\text{tw}(G^{\text{UN}}) = \Omega(\text{dtw}(G))$. In particular this allows the construction of several disjoint concentric directed cycles in G by exploiting the structure of the grid, Eulerianness, and planarity. We follow their ideas and show that the entire construction can be done in polynomial time to yield $\Omega(\text{dtw}(H)/\Delta(H))$ concentric disjoint cycles.

The final step is to find many disjoint paths that cross the concentric cycles from the inner cycle to the outer cycle and many disjoint paths from the outer cycle to the inner cycle. We show that we can find such paths via some ideas in [24] but with the additional property that these paths originate at the terminals. The collection of concentric cycles with these crossing paths is our desired crossbar and we also obtain the required property that the terminals are linked to this crossbar. We note that [24] have to do considerable work to obtain the cylindrical grid while we are satisfied with the constant congestion properties of the cycles plus paths (see Fig.1).

In the end, we arrive at the following statement whose proof is presented in Section 3.

▶ **Theorem 3.** *Given a plane directed Eulerian graph G of maximum in-degree at most Δ and an α-node-well-linked set X in G with $|X| = \Omega(\Delta^2/\alpha)$, one can in polynomial time find a set of $\Omega(\alpha|X|/\Delta)$ concentric cycles going in the same direction (i.e., all clockwise or all counter-clockwise), sets $Y^+, Y^- \subseteq X$ of size $|Y^+| = |Y^-| = \Omega(\alpha^2|X|/\Delta^2)$ each, and families $\mathcal{P}^+$ and $\mathcal{P}^-$ of node-disjoint paths, such that either*

1. *none of the cycles enclose any vertex of $Y^+ \cup Y^-$, the family $\mathcal{P}^+$ consists of $|Y^+|$ node-disjoint paths from Y^+ to the innermost cycle, and the family $\mathcal{P}^-$ consists of $|Y^-|$ node-disjoint paths from the innermost cycle to Y^-; or*

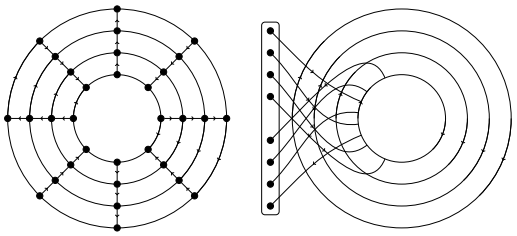


Figure 1 A cylinder (left) and a crossbar we obtain in our proof (right); the vertices in the rounded rectangle is a set of α-well-linked terminals.

$$\max \sum_{i=1}^{k} x_i$$

$$\sum_{p \in \mathcal{P}(s_i,t_i)} f(p) = x_i \qquad i \in [k]$$

$$\sum_{p \in \mathcal{P}(t_i,s_i)} f(p) = x_i \qquad i \in [k]$$

$$\sum_{p:\ v \in p} f(p) \leq 1 \qquad v \in V(G)$$

$$f(p) \geq 0 \qquad p \in \mathcal{P}$$

$$x_i \in [0,1] \qquad i \in [k]$$

Figure 2 Relaxation Sym-Dir-NDP LP.

2. *all cycles enclose $Y^+ \cup Y^-$, the family $\mathcal{P}^+$ consists of $|Y^+|$ node-disjoint paths from Y^+ to the outermost cycle, and the family $\mathcal{P}^-$ consists of $|Y^-|$ node-disjoint paths from the outermost cycle to Y^-.*

Although we are inspired by [24], in the proof of Theorem 3 we use different methodology based on well-linked sets. We also point out that there are significant technical hurdles in working with directed graphs and treewidth. For instance, one can prove that if an undirected graph has treewidth k then it has $\Omega(k/\log k)$ disjoint cycles. This is closely related to the well-known Erdos-Posa theorem [20]. Relating treewidth and disjoint cycles in directed graphs is significantly harder and was resolved in [35] (and also via the more recent result [28]) but the quantitative relationship is weak and far from the known lower bounds.

Using Theorem 3, we show the following statement, which in turn, together with Theorem 2, immediately yields Theorem 1.

▶ **Theorem 4.** *There is an $\mathcal{O}(\Delta^2/\alpha^3)$ approximation with congestion 5 for* Sym-Dir-NDP *in instances for which the input digraph is planar and Eulerian with maximum degree Δ, and the terminals are α-node-well-linked for some $\alpha \leq 1$.*

In this extended abstract, we focus on proving Theorem 3 on constructing the crossbar in Section 3, and we defer the remaining details and proofs to a longer version of this paper.

2 Preliminaries on LP Relaxation and plane Eulerian digraphs

LP relaxation. Our algorithm uses a standard multicommodity flow relaxation for the problem given in Figure 2. We use $\mathcal{P}(u,v)$ to denote the set of all paths in G from u to v, for each ordered pair (u,v) of nodes. Our assumption that the pairs $\mathcal{M}$ form a matching ensures that the sets $\mathcal{P}(s_i,t_i)$, $\mathcal{P}(t_i,s_i)$, $\mathcal{P}(s_j,t_j)$ and $\mathcal{P}(t_j,s_j)$ are pairwise disjoint. Let $\mathcal{P} = \bigcup_{i=1}^{k}(\mathcal{P}(s_i,t_i) \cup \mathcal{P}(t_i,s_i))$. The LP has a variable $f(p)$ for each path $p \in \mathcal{P}$ representing the amount of flow on p. For each (unordered) pair $s_it_i \in \mathcal{M}$, the LP has a variable x_i denoting the total amount of flow routed for the pair (in the corresponding IP, x_i denotes whether the pair is routed or not). The LP imposes the symmetry constraint that there is a flow from s_i to t_i of value x_i *and* a flow from t_i to s_i of value x_i. Additionally, the LP has

capacity constraints that ensure that the total amount of flow on paths using a given node is at most one.[2]

It is convenient to assume that the pairs $\mathcal{M}$ form a matching on $\mathcal{T}$ and each terminal is a leaf of G, i.e., it is attached to a single neighbor using an edge in each direction. As shown in [7], these properties can be ensured as follows. Given an instance $(G, \mathcal{M})$ with terminal $\mathcal{T}$, we create a new instance $(G', \mathcal{M}')$ by attaching a new leaf neighbor t' to every $t \in \mathcal{T}$ with arcs (t, t') and (t', t), and move the terminal t to t'. Given a solution to the LP relaxation on $(G, \mathcal{M})$, we can easily find a solution of at least half of the value by extending the flow along arcs (t, t') and (t', t); the loss of the flow is due to potential capacity violation at vertex t that is now counted twice along the flow paths. If we obtain an integral solution in $(G', \mathcal{M}')$ (i.e., a routing of some pairs from $\mathcal{M}'$) with congestion $c > 1$, by shortening the paths we obtain a routing with the same congestion in $(G, \mathcal{M})$.

Plane Eulerian Digraphs: First, let us recall the following lemma that encapsulates the main property of Eulerian digraphs that make them similar to undirected graphs.

▶ **Lemma 5.** *Let G be an Eulerian digraph of maximum in-degree Δ, let $A, B \subseteq V(G)$, and let ℓ be an integer. If there exist $(\Delta+1)\ell+1$ undirected vertex-disjoint paths from A to B in G, then there exist $\ell+1$ directed ones as well.*

We also need some notation with respect to planar embeddings. Let Π denote the euclidean plane. For a closed Jordan curve γ and a point $p \in \Pi \setminus \gamma$, by $\zeta_p(\gamma) \in \mathbb{Z}$ we denote the element of the fundamental group of $\Pi \setminus \{p\}$ where γ belongs (with the convention that a clockwise cycle around p is the $+1$ element). A Jordan curve γ is in *general position* with respect to the plane graph G if it has finite number of intersections with G, its starting point and ending point do not belong to G, and whenever a point p lies both on γ and in the interior of an edge $e \in E(G)$, then γ traverses the edge e at this point. *A face-edge curve* in a plane digraph G is a Jordan curve in general position that does not traverse any vertex of G.

For a curve γ in general position with respect to G, we introduce the following notions. Assume γ intersects an edge e while going from a face f to a face f'. If e has the face f on the right and the face f' on the left, then we say that e *crosses γ from left to right* and, otherwise, if e has the face f on the left and the face f' on the right, then we say that e *crosses γ from right to left*. By $\mathrm{cross}^{L\to R}(\gamma)$ and $\mathrm{cross}^{R\to L}(\gamma)$ we denote the number of times an edge crosses γ from left to right and from right to left, respectively; note that in these numbers we may count one edge multiple times, one for each moment γ crosses the edge.

For a vertex v in a digraph G, an *imbalance* of v is the number $\mathrm{imb}_G(v) := \mathrm{indeg}_G(v) - \mathrm{outdeg}_G(v)$. A graph is *balanced* if $\mathrm{imb}_G(v) = 0$ for every $v \in V(G)$, and *Eulerian* if it is additionally weakly connected. Furthermore, let the *imbalance* of a curve γ in a general position with respect to G be $\mathrm{imb}(\gamma) = \mathrm{cross}^{L\to R}(\gamma) - \mathrm{cross}^{R\to L}(\gamma)$. A standard argument shows the following:

[2] There is a subtle issue here with regards to the capacity usage at the endpoints of a path. In the integral solution, a pair of paths, one from s_i to t_i and one from t_i to s_i, is regarded as using the vertex s_i only once and using the vertex t_i only once; in other words, such a pair can be seen as a simple cycle passing through s_i and t_i. To simulate it in the LP relaxation, we consider that the starting vertex belongs to a flow path, but the ending vertex does not belong to it. Alternatively, we can assume that a flow path uses only half of the capacities at its endpoints; these interpretations are equivalent due to the symmetry of the demands.

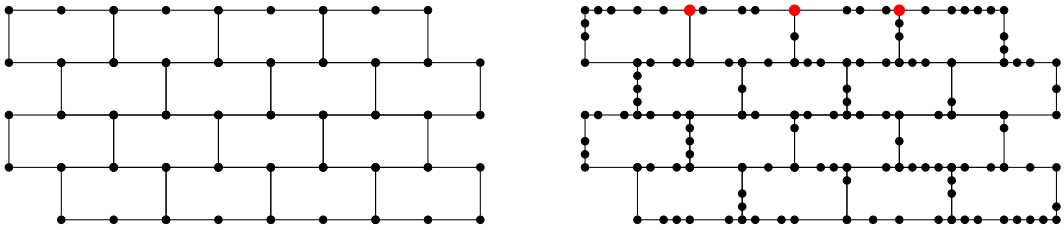


Figure 3 A wall (left) and a subdivided wall (right). The red vertices denote the interface of the wall.

▶ **Lemma 6.** *Let γ be a closed face-edge curve in a plane digraph G. Then*

$$\mathrm{imb}(\gamma) = \sum_{v \in V(G)} \zeta_v(\gamma) \cdot \mathrm{imb}_G(v).$$

We also need the following flow/cut duality.

▶ **Lemma 7.** *Given a plane digraph G, two distinguished faces f^{in} and f^{out}, and an integer k, one can in linear time find either:*

1. *a family of directed vertex-disjoint cycles $C_1, C_2, \ldots, C_k$, all having f^{in} to the right and f^{out} to the left;*
2. *a curve γ in general position with respect to G, that starts in f^{in}, ends in f^{out}, intersects at most k vertices, and satisfies $\mathrm{cross}^{L \to R}(\gamma) = 0$.*

3 The crossbar construction for Eulerian graphs

We first remark that the two outcomes of Theorem 3 are the same if one considers embeddings on the sphere, while on the plane they differ only by the choice of the outer face of the embedding. Furthermore, note that all paths in $\mathcal{P}^+$ and in $\mathcal{P}^-$ intersect every constructed concentric cycle due to planarity.

In the proof of Theorem 3, without loss of generality we assume that every vertex of X is incident to one outgoing arc and one incoming arc, and these two arcs have the same second endpoint: We can achieve it by creating a pendant vertex x' for every $x \in X$, connected to x with arcs (x, x') and (x', x); note that the well-linkedness of X may drop to $\alpha/(\alpha+1)$ in this manner. Consequently, every cycle and path has no vertices in X (except for possibly some endpoints); henceforth we will implicitly use this property multiple times.

Obtaining an undirected grid. We start by applying the construction for *undirected* planar graphs from [8]. Let G^{UN} be the underlying undirected graph of G. Clearly, X is α-node-well-linked in G^{UN} and thus G^{UN} has (undirected) treewidth $\Omega(\alpha|X|)$. Hence we can obtain a large grid minor linked to the terminals X using the following theorem of [8]. In what follows, it is notationally more convenient to work with *subdivided walls as subgraphs*, instead of minors. A $t \times t$ wall and a subdivided wall are shown in Figure 3. The $(t-1)$ vertices of degree three in the top row of a $t \times t$ wall Γ are called the *interface* of the wall, denoted I_Γ.

▶ **Theorem 8** (Theorem 4.5 of [8]). *For every constant $\alpha \leq 1$, given an undirected planar graph H and an α-node-well-linked set X in H, one can in polynomial time find an integer $t = \Omega(\alpha|X|)$, a subdivided $t \times t$ wall Γ in H, and a family of t node-disjoint paths connecting X and the interface of Γ.*

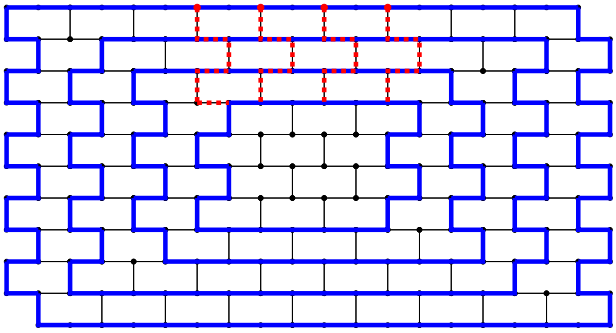


Figure 4 Illustration of Corollary 9. The cycles are blue, while the paths are dashed red.

In our construction we do not need the entire structure of a subdivided wall, but only part of it, as in the following immediate corollary (see Fig. 4).

▶ **Corollary 9.** *One can in polynomial time find an integer $r = \Omega(\alpha|X|)$ and a sequence of node-disjoint concentric undirected cycles $C_1, C_2, \ldots, C_r$ in G^{UN}, with C_1 being the outermost and C_r being the innermost cycle, with the additional property that for every $1 \leq i \leq r$ there exists r vertex-disjoint paths in G^{UN} from X to $V(C_i)$.*

Isles S^{out} and S^{in}. Let us fix a choice of r and cycles $C_1, C_2, \ldots, C_r$ stemming from Corollary 9. For a while, we work only with the undirected graph G^{UN}. Our goal is to strengthen the requirement of the existence of many undirected paths between X and the innermost and outermost cycles by getting more properties about their endpoints, so that we can use an argument similar to the one of [24] to reason about the existence of *directed* concentric cycles with similar connectivity towards X.

To this end, we identify two small connected parts of G^{UN}, S^{out} and S^{in}, one around C_1 and one around C_r. The parts will be large enough so that there is a substantial number of vertex-disjoint directed paths between them and X, but small enough so that they are placed very locally in the graph, and their boundary is small. This last property ensures that after deletion of these parts, the graph is close to Eulerian, and we can make use of Lemma 6.

For a vertex set $Q \subseteq V(G^{\mathrm{UN}})$, a vertex $v \notin Q$, and an integer $\ell \geq 2\Delta$, we say that a vertex set S is a (v, Q, ℓ)-*isle* if $v \in S$, $G^{\mathrm{UN}}[S]$ is connected, $S \cap Q = \emptyset$, and $|N_{G^{\mathrm{UN}}}(S)| \leq \ell$.[3] We will rely on the following greedy procedure, that is inspired by the enumeration algorithm for important separators in parameterized complexity (cf. [12] and [18, Chapter 8]).

▶ **Lemma 10.** *Given a set $Q \subseteq V(G^{\mathrm{UN}})$, a vertex $v \notin Q$, and an integer $\ell \geq 2\Delta$, one can in $O(\ell^3 n)$ time find an inclusion-wise maximal (v, Q, ℓ)-isle.*

Proof. We perform the following iterative procedure. Start with $S = \{v\}$; clearly, S is a (v, Q, ℓ)-isle, as $v \notin Q$ by assumption and the maximum in-degree of G is Δ. In an iterative step, we assume that S is a (v, Q, ℓ)-isle, and our goal is to check if S is an inclusion-wise maximal one, or produce a (v, Q, ℓ)-isle S' with $S \subsetneq S'$.

[3] We use the following notation with respect to neighborhoods. Let G be an undirected graph, $x \in V(G)$, and $S \subseteq V(G)$. Then $N_G(x)$ is the set of neighbors of x in G, $N_G[x] = \{x\} \cup N_G(x)$, $N_G[S] = \bigcup_{x \in S} N_G[x]$, $N_G(S) = N_G[S] \setminus S$, $N^2_G[S] = N_G[N_G[S]]$, and $N^2_G(S) = N_G(N_G[S])$.

To this end, consider every $w \in N_{G^{\text{UN}}}(S) \setminus Q$; note that, by the connectivity of S' and S, there exists such $w \in S' \setminus S$ for every isle S' we are looking for. Collapse in G^{UN} the set $S \cup \{w\}$ into a single vertex s and add a super-source vertex t adjacent to all vertices of Q. Let G' be the resultingg (undirected) graph. Find a minimum $s-t$ vertex cut Z in G', or conclude that such a minimum cut is of size larger than ℓ; this can be done using $\mathcal{O}(\ell)$ rounds of the Ford-Fulkerson algorithm, taking total time $\mathcal{O}(\ell n)$. Moreover, within this time we can find the minimum cut *closest to* t, that is, the unique one with inclusion-wise maximal set of vertices remaining in the connected component with the vertex s (cf. [18]).

If such a cut is found, let S' be the subset of vertices of G corresponding to the connected component of $G' \setminus Z$ containing the vertex s. Clearly, $N_{G^{\text{UN}}}(S') = Z$, and S' is a (v, Q, ℓ)-isle containing S and w. Otherwise, we conclude that no (v, Q, ℓ)-isle containing both S and w exists, since for every such isle S', the set $N_{G^{\text{UN}}}(S')$ is an $s-t$ cut in G' of size at most ℓ.

The computation for fixed S and w takes $\mathcal{O}(\ell n)$ time. Since S is an (v, Q, ℓ)-isle, there are at most ℓ vertices w to try. Due to the fact that we always take the $s-t$ cut closest to t, the size of the set $N_{G^{\text{UN}}}(S)$ strictly grows at every iteration (possibly except the first one, when $S = \{v\}$). Consequently, they are at most $\ell + 1$ iterations of the procedure, and the running time bound follows. ◀

We pick an arbitrary vertex v^{out} on C_1 and an arbitrary vertex v^{in} on C_r, and use Lemma 10 for both these vertices, the set $Q := X$, and threshold $\ell := \lfloor r/(4\Delta + 2) \rfloor$; recall that $|X| = \Omega(\Delta^2/\alpha)$ by the assumptions of Theorem 3 thus we may assume $\ell \geq 2\Delta$. Let S^{out} and S^{in} be the two isles obtained. Since $\ell < r$, and every cycle C_i is connected with r vertex-disjoint paths to X, no cycle C_i is contained in either S^{out} or S^{in}. Since an isle is connected, we obtain the following.

▶ **Lemma 11.** *The isle S^{out} does not contain any vertex that is enclosed by $C_{\ell+1}$, and the isle S^{in} does not contain any vertex that is not strictly enclosed by $C_{r-\ell}$.*

Proof. The proofs for S^{in} and S^{out} are symmetrical, so we just focus here on the case of S^{out}. Assume to the contrary that S^{out} contains a vertex enclosed by $C_{\ell+1}$. Since $v^{\text{out}} \in S^{\text{out}}$ and by the connectity of S^{out}, S^{out} contains a vertex from every cycle C_i, $1 \leq i \leq \ell + 1$. Since $|N_{G^{\text{UN}}}(S^{\text{out}})| \leq \ell$, for some $1 \leq i \leq \ell + 1$ we have that $V(C_i)$ is completely contained in S^{out}. However, recall that there are $r > \ell$ vertex-disjoint paths in G^{UN} connecting C_i with X. This contradicts the facts that $S^{\text{out}} \cap X = \emptyset$ and $|N_{G^{\text{UN}}}(S^{\text{out}})| \leq \ell$. ◀

By Lemma 11, the isles S^{out} and S^{in} are somewhat local in the graph: they do not go too deep into the set of cycles $C_1, C_2, \ldots, C_r$. On the other hand, recall that they are *inclusion-wise maximal* isles; by the next lemma, this ensures that they are connected by a large number of vertex-disjoint undirected paths to the set X. Let $W^{\text{out}} = N^2_{G^{\text{UN}}}[S^{\text{out}}]$ and $W^{\text{in}} = N^2_{G^{\text{UN}}}[S^{\text{in}}]$.

▶ **Lemma 12.** *In G^{UN}, there are $\ell + 1$ node-disjoint undirected paths connecting W^{out} and X and $\ell + 1$ node-disjoint undirected paths connecting W^{in} and X.*

Proof. By symmetry, we can focus on the case of W^{out}. The intuition is as follows: if there does not exist a sufficient amount of desired node-disjoint paths, then the corresponding cut would allow us to construct a strictly larger isle, a contradiction to the maximality of S^{out}. In some sense, $N_{G^{\text{UN}}}(S^{\text{out}})$ is the "last bottleneck" of size at most ℓ between v^{out} and X, and, after passing it, we should have more than ℓ paths between X and $N^2_{G^{\text{UN}}}[S^{\text{out}}] = W^{\text{out}}$.

Formally, assume the contrary of the lemma statement; by Menger's theorem, there exist vertex sets $A, B \subseteq V(G^{\text{UN}})$ such that $A \cup B = V(G^{\text{UN}})$, $|A \cap B| \leq \ell$, $W^{\text{out}} \subseteq A$, $X \subseteq B$, and no edge of G^{UN} has one endpoint in $A \setminus B$ and the second endpoint in $B \setminus A$.

Recall that $S^{\text{out}} \cap X = \emptyset$ by the definition of an isle, while $N^2_{G^{\text{UN}}}[S^{\text{out}}] = W^{\text{out}} \subseteq A$. Hence we may assume that $(N_{G^{\text{UN}}}[S^{\text{out}}] \setminus X) \subseteq (A \setminus B)$, as removing all vertices of $N_{G^{\text{UN}}}[S^{\text{out}}] \setminus X$ from B would not invalidate any of the properties of the pair (A, B). Recall also that $G^{\text{UN}}[S^{\text{out}}]$ is connected; let S_A be the vertex set of the connected component of $G^{\text{UN}} \setminus (A \cap B)$ containing S^{out}. Clearly, $S_A \subseteq A \setminus B$, so $S_A \cap X = \emptyset$. Furthermore, $N_{G^{\text{UN}}}(S_A) \subseteq A \cap B$, so $|N_{G^{\text{UN}}}(S_A)| \leq \ell$. As $S^{\text{out}} \subseteq S_A$, by the maximality of S^{out}, we infer that $S_A = S^{\text{out}}$. Since $N_{G^{\text{UN}}}[S^{\text{out}}] \setminus X \subseteq S_A$, we infer that $N_{G^{\text{UN}}}(S^{\text{out}}) \subseteq X$. However, this is a contradiction, as G^{UN} is connected and $S^{\text{out}} \subsetneq V(G^{\text{UN}}) \setminus X$. ◀

Finding directed concentric cycles. We now use the undirected cycles $C_1, C_2, \ldots, C_r$ to find a large number of node-disjoint *directed* concentric cycles separating S^{in} and S^{out}. Recall that $\ell := \lfloor r/(4\Delta + 2) \rfloor$.

▶ **Lemma 13.** *One can in polynomial time find $\lceil \ell/2 \rceil$ node-disjoint directed concentric cycles, all going in the same direction (all clockwise or all counter-clockwise), such that all vertices of S^{in} are strictly enclosed by the innermost cycle, and all vertices of S^{out} are not enclosed by the outermost cycle, or vice versa, with the roles of S^{in} and S^{out} swapped.*

Proof. Denote $G' = G \setminus (N_{G^{\text{UN}}}[S^{\text{out}}] \cup N_{G^{\text{UN}}}[S^{\text{in}}])$. Let f^{out} and f^{in} be the faces of G' that contain S^{out} and S^{in}, respectively; by Lemma 11, the cycle $C_{\lceil r/2 \rceil}$ remains in G' and $f^{\text{out}} \neq f^{\text{in}}$. Furthermore, the vertices of $N^2_{G^{\text{UN}}}(S^{\text{out}})$ lie on the face f^{out} of G', and the vertices of $N^2_{G^{\text{UN}}}(S^{\text{in}})$ lie on the face f^{in}. We apply Lemma 7 twice to the graph G' and the requirement of ℓ cycles, once for the pair of faces $(f^{\text{out}}, f^{\text{in}})$ and once for the pair $(f^{\text{in}}, f^{\text{out}})$. If at least one of the applications returns a family of cycles, then we are done, as every cycle encloses either S^{in} or S^{out}. Thus, we are left with the case when both the applications return a curve; let us denote these curves γ_1 and γ_2, respectively.

Before we proceed to the formal calculations leading to a contradiction, let us give some intuition. The curves γ_1 and γ_2 are very skewed in terms of the directions of edges crossing it: only edges in one direction are allowed, while in the second direction only ℓ vertices are allowed, and every vertex is of maximum in-degree Δ. The locality of isles S^{out} and S^{in} (Lemma 11) implies that γ_1 and γ_2 cross most of the cycles C_i; consequently, they need to cross much more than $\ell\Delta$ arcs in one direction. However, the graph G' is very close to an Eulerian one, as we have a bound of ℓ on the size of the boundary of S^{out} and S^{in}. This leads to a contradiction with Lemma 6 for a closed curve being essentially a concatenation of γ_1 and γ_2.

Formally, let us first modify the curve γ_1 to obtain a face-edge curve γ_1' as follows: whenever γ_1 crosses a vertex v, we move it slightly to avoid v, at the cost of intersecting some of the arcs incident to v. Since the maximum in-degree and out-degree of G (and thus G') is at most Δ, we have that $\text{cross}^{L \to R}(\gamma_1') \leq \Delta(\ell - 1)$. Similarly, we obtain a curve γ_2' with $\text{cross}^{L \to R}(\gamma_2') \leq \Delta(\ell - 1)$. Since γ_1' starts in f^{out} and ends in f^{in}, while γ_2' starts in f^{in} and ends in f^{out}, we can concatenate these curves (without introducing any new intersection with G') and obtain a closed face-edge curve γ'. This curve γ' visits both f^{out} and f^{in}, and satisfies

$$\text{cross}^{L \to R}(\gamma') \leq 2\Delta(\ell - 1). \tag{1}$$

By Lemma 11, the undirected cycles $C_{\ell+2}, C_{\ell+3}, \ldots, C_{r-\ell-1}$ remain in G', and both γ_1' and γ_2' need to cross at least one edge of each of these cycles. Consequently,

$$\text{cross}^{L \to R}(\gamma') + \text{cross}^{R \to L}(\gamma') \geq 2(r - 2\ell - 2). \tag{2}$$

By merging (1) and (2), and by the choice of ℓ, we obtain that:

$$-\operatorname{imb}(\gamma') = \operatorname{cross}^{R\to L}(\gamma') - \operatorname{cross}^{L\to R}(\gamma') \geq 2(r - 2\ell - 2 - 2\Delta(\ell-1)) > 4\Delta\ell. \tag{3}$$

On the other hand, note that every vertex of G' with a non-zero imbalance is a former neighbor of a vertex of $N_{G^{\text{UN}}}(S^{\text{out}} \cup S^{\text{in}})$. As there are at most 2ℓ vertices in $N_{G^{\text{UN}}}(S^{\text{out}} \cup S^{\text{in}})$, and every such vertex has in-degree and out-degree bounded by Δ, we have

$$\sum_{v \in V(G')} |\operatorname{imb}_{G'}(v)| \leq \sum_{u \in N_{G^{\text{UN}}}(S^{\text{out}} \cup S^{\text{in}})} \operatorname{indeg}_G(u) + \operatorname{outdeg}_G(u) \leq 4\Delta\ell. \tag{4}$$

Equations (3) and (4) stand in contradiction with Lemma 6. ◀

Finishing the crossbar construction. Let $C'_1, \ldots, C'_{\lceil \ell/2 \rceil}$ be the concentric directed cycles found by Lemma 13; by symmetry, assume they all enclose S^{in}. Let $q := \lceil \ell/4 \rceil$. We consider two cases, depending on whether at least half of the vertices of X are enclosed by C'_q or not. The two cases correspond to the two symmetric outcomes of Theorem 3. In what follows, we describe only the first case, when at least half of the vertices of X are enclosed by C'_q, and we use cycles $C'_1, C'_2, \ldots, C'_q$ and vertex-disjoint paths from X to W^{out}; the second case is completely symmetric, but uses cycles $C'_{q+1}, C'_{q+2}, \ldots, C'_\ell$ and paths from X to W^{in}.

Consider the set of $\ell + 1$ paths in G^{UN} connecting W^{out} and X, whose existence is promised by Lemma 12, and let X^{out} be the set of the endpoints of the paths. The vertices in X^{out} may not be enclosed by C'_q. Our goal is to find a different set of vertices that are enclosed by C'_q such that they have disjoint paths to W^{out}; we use well-linkedness of X for this purpose. As $\ell + 1 \leq |X|/2$ and the set X is α node-well-linked, for every set $X' \subseteq X$ of $\ell + 1$ vertices enclosed by C'_q, there exist $\alpha(\ell + 1)$ node-disjoint paths connecting X^{out} and X'. By combining these paths with the paths connecting W^{out} and X^{out}, we obtain a flow that sends $\alpha(\ell + 1)/2$ amount of flow in G^{UN} with unit node capacities from X' to W^{out}, with $1/2$ originating in every vertex in X'. Consequently, there exists a set $Y \subseteq X$ of size at least $\alpha(\ell + 1)/2$, whose vertices are all enclosed by C'_q, and such that there exist $|Y|$ node-disjoint paths in G^{UN} connecting Y and W^{out}.

By Lemma 5, there exist at least $(\alpha\ell - 2)/(2(\Delta + 1))$ node-disjoint directed paths from Y to W^{out} (we let $Y^+ \subset Y$ denote the end points of these paths) and the same amount of node-disjoint directed paths from W^{out} to Y (we let $Y^- \subset Y$ denote the end points of these paths). Recall that $\ell = \Theta(\alpha|X|/\Delta)$, thus $(\alpha\ell - 2)/(2(\Delta + 1)) = \Theta(\alpha^2|X|/\Delta^2)$. As no vertex of W^{out} is strictly enclosed by C'_1, these paths, together with the cycles $C'_1, C'_2, \ldots, C'_q$, form the desired structure. This concludes the proof of Theorem 3.

4 Concluding Remarks

Our main technical contribution in this paper is to show that a planar directed graph has a constant congestion routing structure of size $\Omega(h/\text{polylog}(h))$, where $h = \text{dtw}(G)$. This structural result was motivated by the algorithmic problem of routing symmetric demands in directed graphs. Recent results, in the undirected graph setting, have demonstrated effectively the inherent synergy between approximation algorithms for routing problems and structural results in graph theory related to treewidth. The work in [7] and here are steps towards extending this synergy to directed graphs. The directed graph setting is significantly more challenging, however, and progress in this direction could yield several new benefits. We raise some open problems below.

- Does a planar directed graph with treewidth h have a constant congestion crossbar of size $\Omega(h)$. This would strengthen our result. In particular, is there a cylindrical grid minor of size $\Omega(h)$?
- The techniques in this paper could likely be extended to directed graphs that can be embedded on a bounded genus surface, and more generally to directed graphs whose undirected support graph is from a proper minor-closed family. The ideas of well-linked decomposition and degree-reduction do not rely on planarity. Moreover, there is a linear relationship between treewidth and the size of a grid-minor in undirected graphs from a proper minor-closed family [19].
- Does a general directed graph with treewidth h have a constant congestion crossbar of size $\Omega(h/\text{polylog}(h))$? Is there a cylindrical grid minor of size $\Omega(h^\delta)$ for some fixed $\delta > 0$?

References

1 M. Andrews. Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *Proc. of IEEE FOCS*, pages 277–286, 2010.

2 M. Andrews, J. Chuzhoy, V. Guruswami, S. Khanna, K. Talwar, and L. Zhang. Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. *Combinatorica*, 30(5):485–520, 2010.

3 C. Chekuri and J. Chuzhoy. Large-treewidth graph decompositions and applications. In *Proc. of ACM STOC*, pages 291–300, 2013.

4 C. Chekuri and J. Chuzhoy. Polynomial bounds for the grid-minor theorem. In *Proc. of ACM STOC*, pages 60–69, 2014.

5 C. Chekuri and J. Chuzhoy. Degree-3 treewidth sparsifiers. In *Proc. of ACM-SIAM SODA*, pages 242–255, 2015.

6 C. Chekuri and A. Ene. Poly-logarithmic approximation for maximum node disjoint paths with constant congestion. In *Proc. of ACM-SIAM SODA*, pages 326–341, 2013.

7 C. Chekuri and A. Ene. The all-or-nothing flow problem in directed graphs with symmetric demand pairs. *Mathematical Programming*, pages 1–24, 2014.

8 C. Chekuri, S. Khanna, and F.B. Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *Proc. of ACM STOC*, pages 183–192, 2005.

9 C. Chekuri, S. Khanna, and F.B. Shepherd. An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(7):137–146, 2006.

10 Chandra Chekuri, Sreeram Kannan, Adnan Raja, and Pramod Viswanath. Multicommodity flows and cuts in polymatroidal networks. *SIAM Journal on Computing*, 44(4):912–943, 2015. Preliminary version in Proc. of ITCS, 2012.

11 Chandra Chekuri, Sanjeev Khanna, and F Bruce Shepherd. The all-or-nothing multicommodity flow problem. *SIAM Journal on Computing*, 42(4):1467–1493, 2013. Preliminary version in Proc. of ACM STOC, 2004.

12 J. Chen, Y. Liu, S. Lu, B. O'Sullivan, and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5), 2008.

13 J. Chuzhoy. Routing in undirected graphs with constant congestion. In *Proc. of ACM STOC*, pages 855–874, 2012.

14 J. Chuzhoy. Excluded grid theorem: Improved and simplified. In *Proc. of ACM STOC*, pages 645–654, 2015.

15 J. Chuzhoy, V. Guruswami, S. Khanna, and K. Talwar. Hardness of routing with congestion in directed graphs. In *Proc. of ACM STOC*, pages 165–178, 2007.

16 J. Chuzhoy, D.H.K. Kim, and S. Li. Improved approximation for node-disjoint paths in planar graphs. In *Proc. of ACM STOC*, 2016. To appear.

17 J. Chuzhoy and S. Li. A polylogarithimic approximation algorithm for edge-disjoint paths with congestion 2. In *Proc. of IEEE FOCS*, 2012.

18 M. Cygan, F. Fomin, L. Kowalik, D. Loksthanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2014. In print.

19 E. D. Demaine and M. Hajiaghayi. Linearity of grid minors in treewidth with applications through bidimensionality. *Combinatorica*, 28(1):19–36, 2008.

20 P. Erdos and L. Pósa. On independent circuits contained in a graph. *Canadian Journal of Mathematics*, 17:347–352, 1965.

21 G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. *Journal of the ACM*, 47(4):585–616, 2000.

22 Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

23 T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *Journal of Combinatorial Theory, Series B*, 82(1):138–154, 2001.

24 T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Excluding a grid minor in digraphs. Manuscript, `http://arxiv.org/abs/1510.00473`, 2001.

25 S. Kamath, S. Kannan, and P. Viswanath. Network capacity under traffic symmetry: Wireline and wireless networks. *IEEE Transactions on Information Theory*, 60(9):5457–5469, 2014.

26 S. Kannan and P. Viswanath. Capacity of multiple unicast in wireless networks: A polymatroidal approach. *IEEE Transactions on Information Theory*, 60(10):6303–6328, 2014.

27 K. Kawarabayashi and S. Kreutzer. An excluded grid theorem for digraphs with forbidden minors. In *Proc. of ACM-SIAM SODA*, pages 72–81, 2014.

28 K. Kawarabayashi and S. Kreutzer. The directed grid theorem. In *Proc. of ACM STOC*, 2015.

29 R. Khandekar, S. Rao, and U. Vazirani. Graph partitioning using single commodity flows. *Journal of the ACM*, 56(4):19, 2009.

30 P. N. Klein, S.A. Plotkin, S. Rao, and E. Tardos. Approximation algorithms for Steiner and directed multicuts. *Journal of Algorithms*, 22(2):241–269, 1997.

31 S. G. Kolliopoulos and C. Stein. Approximating disjoint-path problems using packing integer programs. *Mathematical Programming*, 99(1):63–87, 2004.

32 A. Louis. Cut-matching games on directed graphs. *CoRR*, abs/1010.1047, 2010.

33 S. Rao and S. Zhou. Edge disjoint paths in moderately connected graphs. *SIAM Journal on Computing*, 39(5):1856–1887, 2010.

34 B. Reed. Introducing directed tree width. *Electronic Notes in Discrete Mathematics*, 3:222–229, 1999.

35 B. Reed, N. Robertson, P. D. Seymour, and R. Thomas. Packing directed circuits. *Combinatorica*, 16(4):535–554, 1996.

36 N. Robertson, P. D. Seymour, and R. Thomas. Quickly excluding a planar graph. *Journal of Combinatorial Theory, Series B*, 62(2):323–348, 1994.

37 P. D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

Quasi-4-Connected Components

Martin Grohe

RWTH Aachen University, Aachen, Germany
grohe@informatik.rwth-aachen.de

Abstract

We introduce a new decomposition of a graphs into *quasi-4-connected components*, where we call a graph *quasi-4-connected* if it is 3-connected and it only has separations of order 3 that separate a single vertex from the rest of the graph. Moreover, we give a cubic time algorithm computing the decomposition of a given graph.

Our decomposition into quasi-4-connected components refines the well-known decompositions of graphs into biconnected and triconnected components. We relate our decomposition to Robertson and Seymour's theory of tangles by establishing a correspondence between the quasi-4-connected components of a graph and its tangles of order 4.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases decompositions, connectivity, tangles

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.8

1 Introduction

Decompositions of graphs into their connected, biconnected and triconnected components are fundamental in structural graph theory, and they also belong to the basic toolbox of algorithmic graph theory. The existence of such decompositions goes back to work of MacLane [12] from the 1930s (also see Tutte [21]). In the 1970s, Hopcroft and Tarjan [10, 20] showed that the decompositions can be computed in linear time.

In modern terms, the decompositions into biconnected and triconnected components are best described as tree decompositions. To state the decomposition theorems and also our main result, a few technical definitions are unavoidable. Recall that a *tree decomposition* of a graph G is a pair (T, β), where T is a tree and β a mapping that associates a set $\beta(t) \subseteq V(G)$, called the *bag* at t, with every node t of the tree T (subject to certain conditions; see Section 2). The *adhesion* of the decomposition is the maximum of the sizes $|\beta(t) \cap \beta(u)|$ for tree edges tu, which intuitively is the order of the separations of the decomposition. Now the decomposition into biconnected components can be phrased as follows: every graph G has a tree decomposition (T, β) of adhesion at most 1 such that for all tree nodes t the induced subgraph $G[\beta(t)]$ is either 2-connected or a complete graph of order at most 2. The decomposition into triconnected components is more complicated, mainly because the triconnected components of a graph are no longer induced subgraphs, but just topological subgraphs. We say that the *torso* of a set $X \subseteq V(G)$ of vertices of a graph G is the graph $G[\![X]\!]$ obtained from the induced subgraph $G[X]$ by adding edges vw for all distinct $v, w \in X$ such that there is a connected component C of $G \setminus X$ with $v, w \in N(C)$, the neighbourhood of C in G. For example, the torso of the set $X = \{x_1, \dots, x_4\}$ in the graph G shown in Figure 1(a) is the complete graph on X. Now the decomposition into triconnected components can be phrased as follows: every graph G has a tree decomposition (T, β) of adhesion at most 2 such that for all tree nodes t the torso $G[\![\beta(t)]\!]$ is a topological subgraph of G that is either 3-connected or a complete graph of order at most 3.

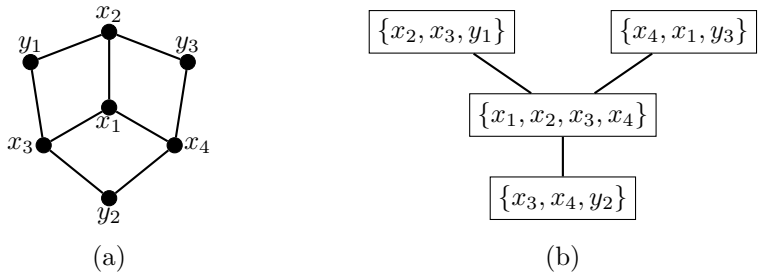


Figure 1 A graph and its decomposition into triconnected components.

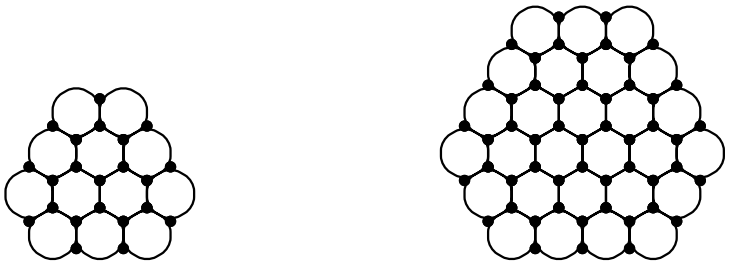


Figure 2 Hexagonal grids of radius 2 and 3.

How about decompositions into 4-connected components, or k-connected components for $k \geq 4$? At least in the clean form of the above decomposition theorems, they simply do not exist. Consider, for example, a hexagonal grid (see Figure 2). Even though the grid is not 4-connected, and it does not even have a nontrivial 4-connected subgraph, there is no good way of decomposing it in a tree-like fashion by separations of order 3. In fact, the only separations of the grid of order 3 are those splitting off a single vertex. If we ignore such separations, we may view the whole grid as one highly connected region. Let us call a graph G *quasi-4-connected* if it is 3-connected and for all separations (Y, S, Z) of order 3 (that is, $|S| = 3$ and Y, S, Z form a partition of $V(G)$ and there are no edges between Y and Z), either $|Y| \leq 1$ or $|Z| \leq 1$. Surprisingly, with this mild relaxation of 4-connectivity we get a nice decomposition theorem along the lines of the decompositions into biconnected and triconnected components.

▶ **Theorem 1** (Decomposition Theorem). *Every graph G has a tree decomposition (T, β) of adhesion at most 3 such that for all tree nodes t the torso $G[\![\beta(t)]\!]$ is a minor of G that is either quasi-4-connected or a complete graph of order at most 4.*

Furthermore, this decomposition can be computed in cubic time.

The decomposition is not unique, but the isomorphism types of the *quasi-4-connected components* into which we decompose are.

There have been earlier generalisations of the decomposition of graphs into triconnected components. The most prominent of these are Robertson and Seymour's tangles [17], which play an important role in the structure theory for graphs with excluded minors [16]. Intuitively, a tangle of order k describes a "k-connected region" in a graph by "pointing to it", that is, by assigning a direction to each separation of order less than k in such a way that "most" of the region described by the tangle is on the side the separation is directed towards. It is known that the tangles of orders 1, 2, 3 are in one-to-one correspondence to the connected, biconnected and triconnected components of a graph [17, 8]. We establish a

similar correspondence between the tangles of order 4 and the quasi-4-connected components. This is our second main theorem, which I think is interesting in its own right, but which is also essential for the proof of Theorem 1. We defer the precise technical statement of this Correspondence Theorem to Section 4 (Theorem 4).

This paper grew out of my work on descriptive complexity theory for graph classes with excluded minors [6, 5], and this may also serve as an illustration of potential applications of our Decomposition Theorem. Separations of order 3 play a special, but somewhat annoying role in the main structure theorems for graph classes with excluded minors such as the "Flat Grid Theorem" of [18] and the structure theorem of [19], and the theorems simplify for quasi-4-connected graphs. In [5] I exploited some of the main ideas underlying our Decomposition Theorem to obtain such simplifications in the context of logical definability, and I believe the Decomposition Theorem proved here may turn out to be similarly useful in an algorithmic context.[1]

Due to space limitations, I had to omit many proof details, examples, and remarks from this conference version of the paper. They can be found in the full version [7] (available on arXiv).

1.1 Related work

It was shown in [17, 1] that for every k, every graph admits a canonical decomposition into its tangles of order k. Related to this is the decomposition into so-called $(k-1)$-blocks due to [3]. An important difference between these results and ours, or rather an additional feature of our decomposition, is that the pieces of our decomposition are quasi-4-connected graphs in their own right and can be dealt with separately (for example in an algorithmic context), whereas tangles of order 4 or 3-blocks are only defined relative to the surrounding graph.

In [15, 14], a notion of k-edge connected component is considered. It is similar to the $(k-1)$-blocks, but with respect to edge connectivity.

On the algorithmic side, it was shown in [9] that the decomposition of a graph into its tangles of order k can be computed in time $n^{O(k)}$. I believe that our techniques can be used to improve this to cubic time for $k = 4$.

There is a different line of work on "k-connected components" that, as far as I can see, is unrelated to ours. There, k-connected components are simply defined as maximal k-connected subgraphs (see, for example, [13]). This leads to completely different decompositions. For example, a graph of maximum degree 3 will only have trivial 4-connected components in this framework. However, what I see as the crucial difference between our form of decomposition and this line of work is that we get *tree* decompositions. This is important for typical dynamic-programming or divide-and-conquer algorithms on the decomposition.

2 Preliminaries

We assume basic knowledge of graph theory and refer the reader to [4] for background. Our notation is standard, let us just review the most important and frequently used notations.

[1] Let me clarify the relation of this work to Chapter 10 of the forthcoming monograph [5]. The basic ideas are the same, and actually my original motivation for the present paper was to make these ideas accessible to readers not interested in logic. However, when I started to work on this paper I noticed the connection to tangles, and it is this connection that provides the right framework and also makes the decomposition much simpler. On the other hand, the main goal of [5] is to obtain a decomposition that is definable in fixed-point logic with counting, and the decomposition we obtain here is not. So, except for some of the basic lemmas underlying the proof of the Correspondence Theorem, the results are incomparable.

All graphs considered in this paper are finite and simple. The vertex set and edge set of a graph G are denoted by $V(G)$ and $E(G)$, respectively. The *order* of G is $|G| := |V(G)|$. For a set $W \subseteq V(G)$, we denote the induced subgraph of G with vertex set W by $G[W]$ and the induced subgraph with vertex set $V(G) \setminus W$ by $G \setminus W$. For a vertex v, we denote the set of neighbours of v in G by $N^G(v)$. In this and similar notations, we omit the index ^{G} if G is clear from the context. For a set $W \subseteq V(G)$, we define $N(W) := \left(\bigcup_{v \in W} N(v)\right) \setminus W$, and for a subgraph $H \subseteq G$ we let $N^G(H) := N^G(V(H))$.

A *tree decomposition* of a graph G is a pair (T, β), where T is a tree and $\beta : V(T) \to 2^{V(G)}$ such that for all $v \in V(G)$ the set $\{t \in V(T) \mid v \in \beta(t)\}$ is connected in T and for all $vw \in E(G)$ there is a $t \in V(T)$ such that $v, w \in \beta(t)$.

A *minor* of G is a graph obtained from G by deleting vertices and edges and contracting edges. A *model* of H in G consists of a family $(M_w)_{w \in V(H)}$ of mutually disjoint connected subsets of $V(G)$ and a family $(e_f)_{f \in E(H)}$ of edges of G such that for every edge $f = ww'$ of H the edge e_f has one endvertex in M_w and one endvertex in $M_{w'}$. Then H is a *minor* of G if and only if there is a model of H in G. We call the sets M_w, for $w \in V(H)$, the *branch sets* of the model $\mathcal{M}$. When reasoning about a model, it is often enough to know the branch sets.

A *faithful model* of H in G is a model $\big((M_w)_{w \in V(H)}, (e_f)_{f \in E(H)}\big)$ such that $w \in M_w$ for all $w \in V(H)$. We say that H is a *faithful minor* of G if $V(H) \subseteq V(G)$ and there is a faithful model of H in G.

Separations of a graph G are usually defined as pairs of subgraphs. However, in this paper it will be more convenient to view them as partitions of the vertex set. We say that a *separation* of G is a triple (Y, S, Z) of (possibly empty) mutually disjoint subsets of $V(G)$ such that $Y \cup S \cup Z = V(G)$ and there is no edge $vw \in E(G)$ such that $v \in Y$ and $w \in Z$. The order of the separation (Y, S, Z) is $|S|$, and the separation is *proper* if both Y and Z are nonempty. The set of all separations of G is denoted by $\mathrm{Sep}(G)$, and the subset of all separations of order less than k (at most k, exactly k) by $\mathrm{Sep}_{<k}(G)$ (resp. $\mathrm{Sep}_{\leq k}(G)$, $\mathrm{Sep}_{=k}(G)$).

A set $S \subseteq V(G)$ is a *separator* of G of *order* $k := |S|$, or a *k-separator*, if there are two vertices $v, w \in V(G) \setminus S$ such that there is a path from v to w in G, but no path from v to w in $G \setminus S$. Note that if G is connected then S is a separator if and only if there is a proper separation (Y, S, Z) of G.

A graph G is *k-connected* if $|G| > k$ and G has no proper $(k-1)$-separation.

A subset $X \subseteq V(G)$ of the vertex set of a graph G is *k-inseparable* if $|X| > k$ and there is no separation (Y, S, Z) of G of order at most k such that $X \cap Y \neq \emptyset$ and $X \cap Z \neq \emptyset$.

3 Tangles

Let G be a graph. Deviating from Robertson and Seymour's [17] original definition, we define tangles as families of separations of the vertex set (as we defined them in Section 2) rather than separations viewed as pairs of graphs or partitions of the edge set. (We show that the two notions are equivalent in the full version [7].) A *G-tangle* of order k is a family $\mathcal{T} \subseteq \mathrm{Sep}_{<k}(G)$ of separations of G of order less than k satisfying the following conditions.

(T.1) For all separations $(Y, S, Z) \in \mathrm{Sep}_{<k}(G)$ either $(Y, S, Z) \in \mathcal{T}$ or $(Z, S, Y) \in \mathcal{T}$.

(T.2) If $(Y_1, S_1, Z_1), (Y_2, S_2, Z_2), (Y_3, S_3, Z_3) \in \mathcal{T}$ then either $Z_1 \cap Z_2 \cap Z_3 \neq \emptyset$ or there is an edge $e \in E(G)$ that has an endvertex in each Z_i.

(T.3) If $(Y, S, Z) \in \mathcal{T}$ then $Z \neq \emptyset$.

For background on tangles and examples, I refer the reader to [17, 8].

Let G be a graph. We define a partial order $\preceq$ on $\mathrm{Sep}(G)$ by letting

$$(Y,S,Z) \preceq (Y',S',Z') \;:\Longleftrightarrow\; S \cup Z \subset S' \cup Z' \text{ or } \big(S \cup Z = S' \cup Z' \text{ and } S \subseteq S'\big). \tag{1}$$

For a G-tangle $\mathcal{T}$, we let $\mathcal{T}_{\min}$ be the set of minimal elements of $\mathcal{T}$ with respect to the partial order $\preceq$. The minimal elements of a tangle will play an important role later. It can be shown that if $(Y,S,Z) \in \mathcal{T}_{\min}$ then Z is connected in G and $S = N(Z)$.

It is shown in [17, 8] that the tangles of order at most 3 are in one-to-one correspondence to the connected, biconnected, and triconnected components of a graph. The following characterisation of the triconnected components motivates our definition of quasi-4-connected regions in the next section.

▶ **Proposition 2.** *Let G be a graph and $R \subseteq V(G)$. The the following are equivalent.*

1. *R is an inclusionwise maximal subset of G such that $G[\![R]\!]$ is 3-connected and a topological subgraph of G.*
2. *$G[\![R]\!]$ is 3-connected and a topological subgraph of G, and for every connected component C of $G \setminus R$ we have $|N(C)| \leq 2$.*

We call sets R satisfying the conditions of this proposition the *triconnected regions* of a graph and the graphs $G[\![R]\!]$ the *triconnected components*.

We can "lift" a tangle from a minor of a graph to the original graph. Let G be a graph, H a minor of G, and $\mathcal{M}$ a model of H in G, say, with branch sets $(M_w)_{w \in V(H)}$. For a separation $(Y,S,Z) \in \mathrm{Sep}(G)$, the *$\mathcal{M}$-projection* of (Y,S,Z) to H is the triple $\pi_{\mathcal{M}}(Y,S,Z) = (Y',S',Z')$ of subsets of $V(H)$ defined by $Y' := \big\{w \in V(H) \;\big|\; V(M_w) \subseteq Y\big\}$, $S' := \big\{w \in V(H) \;\big|\; V(M_w) \cap S \neq \emptyset\big\}$, $Z' := \big\{w \in V(H) \;\big|\; V(M_w) \subseteq Z\big\}$. It is easy to see that (Y',S',Z') is a separation of H of order $|S'| \leq |S|$.

▶ **Lemma 3** ([17])**.** *Let G be a graph, H a minor of G, and $\mathcal{M}$ a model of H in G. Let $\mathcal{T}'$ be an H-tangle of order k. Then the set $\mathcal{T}$ of all separations $(Y,S,Z) \in \mathrm{Sep}_{<k}(G)$ such that $\pi_{\mathcal{M}}(Y,S,Z) \in \mathcal{T}'$ is a G-tangle of order k.*

We call $\mathcal{T}$ be the *lifting* of $\mathcal{T}'$ to G with respect to the model $\mathcal{M}$. Clearly, the lifting may depend on the model. This is even the case if we only consider faithful minors and models.

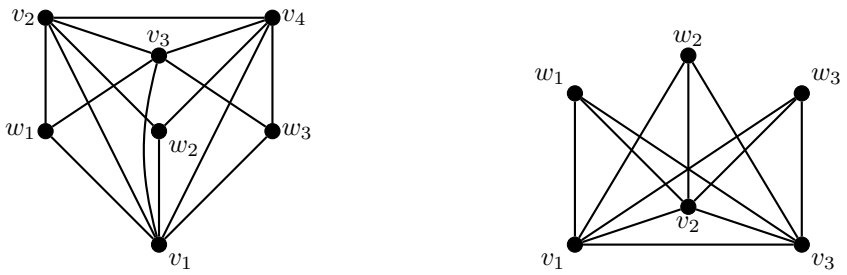
4 Tangles of Order 4

Let us now look at tangles of order 4. We restrict our attention to 3-connected graphs. This is natural; in the full version of the paper we also give a formal justification that we can do this without loss of generality. *For the rest of this section, we assume that G is a 3-connected graph.*

The main result of this section is a correspondence between tangles of order 4 and what we will call *quasi-4-connected regions* of a graph. This correspondence holds for all but a small number of *exceptional* regions, which we shall completely characterise. We first state the theorem; the necessary definitions follow.

▶ **Theorem 4** (Correspondence Theorem)**.** *With every non-exceptional quasi-4-connected region R of G we can associate a G-tangle $\mathcal{T}_R$ of order 4 and with every G-tangle $\mathcal{T}$ of order 4 a non-exceptional quasi-4-connected region $R_{\mathcal{T}}$ such that $\mathcal{T} = \mathcal{T}_{R_{\mathcal{T}}}$.*

We shall call the torsos $G[\![R_{\mathcal{T}}]\!]$ for the G-tangles of order 4 the *quasi-4-connected components* of G.



(a) the graph TH_{+3} (b) the graph TR_{+3}

Figure 3 The exceptional quasi-4-connected graphs.

In general, the mapping $R \mapsto \mathcal{T}_R$ is not injective; the mapping $\mathcal{T} \mapsto R_\mathcal{T}$ is (otherwise the theorem could not hold). The mapping $R \mapsto \mathcal{T}_R$ is *canonical* (or *isomorphism invariant*). This means that for any two graphs G, G' and regions R, R', if f is an isomorphism from G to G' that maps R to R' then f also maps $\mathcal{T}_R$ to $\mathcal{T}_{R'}$. The mapping $\mathcal{T} \mapsto R_\mathcal{T}$ is not canonical. However, the mapping from $\mathcal{T}$ to the quasi-4-connected component $G[\![R_\mathcal{T}]\!]$, viewed as an abstract graph, is.

We can only give a very high-level outline of the proof of the Correspondence Theorem.

4.1 Quasi-4-Connected Graphs and Regions

Recall from the introduction that a graph G is *quasi-4-connected* if G is 3-connected and for all separations $(Y, S, Z) \in \mathrm{Sep}_{=3}(G)$, either $|Y| \leq 1$ or $|Z| \leq 1$. A quasi-4-connected graph G is *exceptional* if it is isomorphic to a subgraph of one of the graphs TH_{+3} or TR_{+3} shown in Figure 3.

▶ **Theorem 5.** *Let G be a quasi-4-connected graph. Then G has a tangle of order 4 if and only if it is not exceptional. Furthermore, if G has a tangle of order 4, it has exactly one such tangle, which consists of all separations $(Y, S, Z) \in \mathrm{Sep}_{<4}(G)$ such that $|Y| < |Z|$.*

A *quasi-4-connected region* of G is a subset $R \subseteq V(G)$ satisfying the following conditions.

(Q.1) $G[\![R]\!]$ is a faithful minor of G.

(Q.2) $G[\![R]\!]$ is quasi-4-connected.

(Q.3) For every connected component C of $G \setminus R$ it holds that $N(C) = 3$.

While conditions (Q.1) and (Q.2) are, to some extent, natural, condition (Q.3) may seem less so. It is a (weak) maximality condition: if $R' \supset R$ such that $G[\![R']\!]$ is quasi-4-connected, then $R' \setminus R$ contains at most one vertex of every connected component C of $G \setminus R$ (unless $|R| = 4$); otherwise $N(C)$ would be separator of $G[\![R']\!]$ witnessing that it is not quasi-4-connected. Conditions (Q.1)–(Q.3) are motivated by the characterisation of the triconnected components given in Proposition 2(2). The reason for choosing these conditions instead of adding some maximality condition is simply that they work best in combination with tangles and for the Decomposition Theorem; it is condition (Q.3) which guarantees that our decomposition will have adhesion 3.

Let R be a quasi-4-connected region of G. If $G[\![R]\!]$ is a non-exceptional quasi-4-connected graph, then it has a unique tangle of order 4, and using Lemma 3, we can lift this tangle to a

G-tangle of order 4. It can be proved that the lifted tangle does not depend on the model of $G[\![R]\!]$ in G, as long as it is faithful. In this case we let $\mathcal{T}_R$ (of the Correspondence Theorem) be this lifted tangle.

However, sometimes we can even associate a tangle with a quasi-4-connected region R if $G[\![R]\!]$ is exceptional. A *non-exceptional extension* of R is a graph $\widehat{H}$ satisfying the following conditions.

(X.1) $\widehat{H}$ is a faithful minor of G.

(X.2) $\widehat{H}$ is non-exceptional quasi-4-connected.

(X.3) $R \subseteq V(\widehat{H})$, and for each connected component C of $G \setminus R$ we have $|V(\widehat{H}) \cap V(C)| \leq 1$.

(X.4) Subject to (X.1)–(X.3), $V(\widehat{H})$ is inclusionwise minimal.

We call the region R *non-exceptional* if it has a non-exceptional extension. Note that if $G[\![R]\!]$ is a non-exceptional quasi-4-connected graph, then $G[\![R]\!]$ is a non-exceptional extension of R.

If $G[\![R]\!]$ is exceptional and $\widehat{H}$ is a non-exceptional extension of R, then there is a unique $\widehat{H}$-tangle $\widehat{\mathcal{T}}$ of order 4. Let $\mathcal{I}$ be a faithful image of $\widehat{H}$ in G. Using Lemma 3, we can lift this tangle to a G-tangle $\mathcal{T}(\widehat{H}, \mathcal{I})$ of order 4. It turns out that this tangle neither depends on the choice of $\widehat{H}$ nor on the choice of $\mathcal{I}$. We let $\mathcal{T}_R := \mathcal{T}(\widehat{H}, \mathcal{I})$.

4.2 The Region of a Tangle

The goal of this section is to define the mapping $\mathcal{T} \mapsto R_{\mathcal{T}}$ from G-tangles of order 4 to quasi-4-connected regions. This is much more difficult than defining the mapping $R \mapsto \mathcal{T}_R$; technically it is clearly the most difficult part of the paper. We can only give the basic idea here. *We fix a G-tangle $\mathcal{T}$ of order 4 for the rest of the section.*

We call two separations $(Y_1, S_1, Z_1), (Y_2, S_2, Z_2) \in \mathrm{Sep}(G)$ *orthogonal* if $(Y_1 \cup S_1) \cap (Y_2 \cup S_2) \subseteq S_1 \cap S_2$. It is not hard to show that the minimal separations of a tangle of order 3 in a graph are mutually orthogonal. The minimal separations of a tangle of order 4 are not necessarily orthogonal, but the next lemma shows that they can only "cross" in a very restricted way.

▶ **Lemma 6** (Crossing Lemma). *Let $(Y_1, S_1, Z_1), (Y_2, S_2, Z_2) \in \mathcal{T}_{\min}$ be distinct. Then either (Y_1, S_1, Z_1) and (Y_2, S_2, Z_2) are orthogonal or $Y_1 \cap Y_2 = \emptyset$ and $S_1 \cap S_2 = \emptyset$ and there is an edge $s_1 s_2 \in E(G)$ such that for $i = 1, 2$ we have $S_i \cap Y_{3-i} = \{s_i\}$.*

In the latter case, we call the edge $s_1 s_2$ the crossedge *of (Y_1, S_1, Z_1) and (Y_2, S_2, Z_2).*

We call a proper separation $(Y, S, Z) \in \mathrm{Sep}_{=3}(G)$ *degenerate* if $|Y| = 1$ and S is an independent set of G. It can be shown that if (Y, S, Z) is non-degenerate then $G[\![Z]\!]$ is a faithful minor of G. We call a crossedge e of separations $(Y_1, S_1, Z_1), (Y_2, S_2, Z_2) \in \mathcal{T}_{\min}$ *non-degenerate* if the two separations are non-degenerate. The key to our proof is the following lemma (which is actually easy to prove).

▶ **Lemma 7** (Crossedge Independence Lemma[2]). *The set of non-degenerate crossedges is a matching of G.*

Let $e^1, \ldots, e^m$ be the non-degenerate crossedges of $\mathcal{T}$, and suppose that $e^i = s_1^i s_2^i$. We contract all these edges to their endvertex s_1^i. The order of the contractions is irrelevant

[2] Actually, this is only a corollary to what we call the "Crossedge Independence Lemma" in the full version [7].

because the edges form a matching. Up to isomorphism, it is also irrelevant whether we contract e^i to s_1^i or s_2^i. Let $G^{(m)}$ be the resulting graph. We show that $G^{(m)}$ is still 3-connected and has a tangle $\mathcal{T}^{(m)}$ of order 4 such that $\mathcal{T}$ is the lifting of $\mathcal{T}^{(m)}$ to G. Furthermore, $\mathcal{T}^{(m)}$ has no non-degenerate crossedges. Hence the non-degenerate separations in $\mathcal{T}^{(m)}_{\min}$ are mutually orthogonal. We let

$$R_{\mathcal{T}} := V(G^{(m)}) \setminus \bigcup_{\substack{(Y,S,Z)\in\mathcal{T}^{(m)}_{\min}\\ \text{non-degenerate}}} Y. \tag{2}$$

We show that $R_{\mathcal{T}}$ is a non-exceptional quasi-4-connected region of G and that $\mathcal{T} = \mathcal{T}_{R_{\mathcal{T}}}$.

5 Decomposition into Quasi-4-Connected Components

With the Correspondence Theorem at hand, it is now relatively easy to prove the Decomposition Theorem 1.

▶ **Theorem 8.** *Let G be a 3-connected graph. Then G has a tree decomposition (T, β) of adhesion at most 3 such that for all $t \in V(T)$, the torso $G[\![\beta(t)]\!]$ is either a complete graph K_3 or K_4 or a quasi-4-connected component of G.*

Furthermore, such a decomposition can be computed in time $O(n^2(n+m))$.

Here, and throughout this section, we denote the numbers of vertices and edges of the input graph G of our algorithms by n and m, respectively.

The Decomposition Theorem 1 follows by combining the decomposition of Theorem 8 with the standard decomposition of a graph into its triconnected components.

The proof of Theorem 8 requires some preparation. For the rest of this section, we assume that G is a 3-connected graph. Let $(Y, S, Z) \in \mathrm{Sep}_{=3}(G)$ be non-degenerate. A *split vertex* of (Y, S, Z) is a vertex $z \in Z$ such that for every connected component C of $G \setminus (S \cup \{z\})$ it holds that $|N(C)| = 3$.

▶ **Lemma 9.** *Let $(Y_0, S_0, Z_0) \in \mathrm{Sep}_{=3}(G)$ be a non-degenerate proper separation such that Z_0 is connected and (Y_0, S_0, Z_0) has no split vertex. Then the set $\mathcal{T}(Y_0, S_0, Z_0)$ of all separations $(Y, S, Z) \in \mathrm{Sep}_{<4}(G)$ such that either $Z_0 \subseteq Z$ or $|Z \cap S_0| > |Y \cap S_0|$ is a G-tangle of order 4.*

Proof. Let $\mathcal{T} := \mathcal{T}(Y_0, S_0, Z_0)$. To see that $\mathcal{T}$ satisfies (T.1), let $(Y, S, Z) \in \mathrm{Sep}_{<4}(G)$. If $S \subseteq Y_0 \cup S_0$, then the connected set Z_0 is either a subset of Z or of Y, and thus either $(Y, S, Z) \in \mathcal{T}$ or $(Z, S, Y) \in \mathcal{T}$. Suppose next that $|S \cap Z_0| = 1$. Let z be the unique vertex in $S \cap Z_0$. Then z is not a split vertex of (Y_0, S_0, Z_0), and hence there is a connected component C of $G \setminus (S_0 \cup \{z\})$ such that $N(C) = S_0 \cup \{z\}$. Then $V(C) \subseteq Z_0$, because $z \in Z_0$, and thus $V(C) \cap S = \emptyset$. It follows that either $V(C) \subseteq Y$ or $V(C) \subseteq Z$. Without loss of generality we may assume that $V(C) \subseteq Z$. As $S_0 \subseteq N(C)$, this implies $S_0 \setminus S \subseteq Z$. As $S_0 \setminus S \neq \emptyset$, it follows that $(Y, S, Z) \in \mathcal{T}$. Finally, suppose that $|S \cap Z_0| \geq 2$. If $S \cap S_0 = \emptyset$, then either $|Z \cap S_0| \geq 2$ or $|Y \cap S_0| \geq 2$, and thus either $(Y, S, Z) \in \mathcal{T}$ or $(Z, S, Y) \in \mathcal{T}$. If $|S \cap S_0| = 1$, then $S \cap Y_0 = \emptyset$, and as G is 3-connected and $Y_0 \neq \emptyset$, the vertices in $S_0 \setminus S$ belong to the same connected component of $G \setminus S$. Hence either both are in Z or both are in Y, and again it follows that either $(Y, S, Z) \in \mathcal{T}$ or $(Z, S, Y) \in \mathcal{T}$.

Observe next that $|V(G)| \geq 6$, because $|Y_0| \geq 1$ and $|S_0| = 3$ and $|Z_0| \geq 2$ (otherwise the unique vertex in Z_0 would be a split vertex).

▶ **Claim 10.** *For all $(Y, S, Z) \in \mathcal{T}$ we have $|S \cup Z| \geq 4$.*

Proof. It follows from the definition of $\mathcal{T}$ that $Z \neq \emptyset$. If $Y = \emptyset$, then $|S \cup Z| = |V(G)| \geq 6$. Otherwise, (Y, S, Z) is a proper separation and thus $|S| = 3$, which implies $|S \cup Z| \geq 4$. ◀

The claim implies that $\mathcal{T}$ satisfies (T.3).

To prove that $\mathcal{T}$ satisfies (T.2), let $(Y_i, S_i, Z_i) \in \mathcal{T}$ for $i = 1, 2, 3$. Suppose for contradiction $Z_1 \cap Z_2 \cap Z_3 = \emptyset$ and that there is no edge that has an endvertex in each Z_i.

▶ **Claim 11.** *For distinct $i, j, k \in [3]$ and $x \in V(G)$, if $x \in Z_i \cap Z_j$ then $x \in Y_k$.*

Proof. We have $x \notin Z_k$, because $Z_i \cap Z_j \cap Z_k = \emptyset$. Suppose that $x \in S_k$, and let $z \in N(x) \cap Z_k$. Such a z exists, because $Z_k \neq \emptyset$ and $N(Z_k) \subseteq S_k$, and as $|S_k| \leq 3$ and G is 3-connected, this implies $N(Z_k) = S_k$. But the edge xz has an endvertex in every Z_i, which contradicts our assumption that no such edge exists. ◀

Case 1: There is an $i \in [3]$ such that $S_i \subseteq Y_0 \cup S_0$. Without loss of generality, we may assume that $i = 1$ and $(Y_1, S_1, Z_1) = (Y_0, S_0, Z_0)$. We may further assume that $S_i \not\subseteq Y_0 \cup S_0$ for $i = 2, 3$. Then $|Z_i \cap S_0| > |Y_i \cap S_0|$.

By Claim 11 we have $Z_2 \cap Z_3 \cap S_0 = Z_2 \cap Z_3 \cap S_1 = \emptyset$. Thus for some $i \in \{2, 3\}$ $|Z_i \cap S_0| < 2$. Without loss of generality we assume $|Z_2 \cap S_0| < 2$. Then $|Y_2 \cap S_0| = \emptyset$ and thus $|S_2 \cap S_0| = 2$. Since $S_2 \not\subseteq Y_0 \cup S_0$, we have $|S_2 \cap Z_0| = 1$. As the vertex in $S_2 \cap Z_0$ is not a split vertex, there is a connected component C of $G \setminus (S_0 \cup S_2)$ such that $N(C) = S_0 \cup S_2$. Then $V(C) \subseteq Z_0 \cap Z_2 = Z_1 \cap Z_2$. Now let $v \in Z_3 \cap S_0$, and let $w \in V(C)$ be adjacent to v. Then the edge vw has an endvertex in each Z_i.

Case 2: $|S_i \cap Z_0| \neq \emptyset$ for all $i \in [3]$. Then $|Z_i \cap S_0| > |Y_i \cap S_0|$. If $|Z_i \cap Z_j \cap S_0| = \emptyset$ for all $i \neq j$, then $|Z_i \cap S_0| = 1$ and thus $|Y_i \cap S_0| = 0$ for all i. Thus $|S_i \cap S_0| = 2$ and $|S_i \cap Y_0| = \emptyset$, because $S_i \not\subseteq S_0 \cup Y_0$. But this implies $Y_0 \subseteq Z_1 \cap Z_2 \cap Z_3$, which is a contradiction.

Hence without loss of generality we may assume that $Z_1 \cap Z_2 \cap S_0 \neq \emptyset$. Let $s \in Z_1 \cap Z_2 \cap S_0$. Then by Claim 11, $s \in Y_3$. Then $|Y_3 \cap S_0| \geq 1$, and this implies $|Z_3 \cap S_0| \geq 2$. Let $s', s'' \in Z_3 \cap S_0$. Then $S_0 = \{s, s', s''\}$.

If $|S_3 \cap Z_0| \leq 1$, there is a connected component C of $G \setminus (S_0 \cup S_3)$ such that $N(C) = S_0 \cup S_3$. But then there is a path from $s \in Y_3$ to $s' \in Z_3$ in $G \setminus S_3$, which is impossible. Hence $|S_3 \cap Z_0| \geq 2$.

Thus $|S_3 \cap Y_0| \leq 1$. Since G is a 3-connected and $Y_0 \neq \emptyset$, there is a path from s to $\{s', s''\}$ with all internal vertices in Y_0. Hence $|Y_0 \cap S_3| = 1$, and the unique vertex $y \in Y_0 \cap S_3$ separates $s \in Y_3$ from $\{s', s''\} \subseteq Z_3$ in the graph $G[Y_0 \cup S_0]$. Then $ss', ss'' \notin E(G)$. Furthermore, $sy \in E(G)$ and y is the only neighbour of s in $Y_0 \cup S_0$, because otherwise $\{y, s\}$ would be separator of G. By Claim 11 and because $y \in S_3$, we have $y \notin Z_1 \cap Z_2$. Say, $y \notin Z_2$. Then $y \in S_2$, because y is adjacent to $s \in Z_2$. As $S_2 \not\subseteq Y_0 \cup S_0$, it now follows that s' and s'' are not both in S_2. As $|Z_2 \cap S_0| > |Y_2 \cap S_2|$, one of these vertices, say, s' is in Z_2.

By Claim 11, $s' \in Z_2 \cap Z_3$ implies $s' \in Y_1$. Arguing as above with (Y_1, S_1, Z_1) instead of (Y_3, S_3, Z_3), we see that $Z_1 \cap S_0 = \{s, s''\}$ and $|S_1 \cap Z_0| = 2$ and $|S_1 \cap Y_0| = 1$, and the unique vertex $y' \in S_1 \cap Y_0$ separates s' from s, s'' in G. Furthermore, $s's, s's'' \notin E(G)$, and $s'y' \in E(G)$ and y' is the only neighbour of s' in $Y_0 \cup S_0$.

Now we have $s'' \in Z_1 \cap Z_3$, and again by the same argument we see that $s'' \in Y_2$ and $Z_2 \cap S_0 = \{s, s'\}$ and $|S_2 \cap Z_0| = 2$ and $|S_2 \cap Y_0| = 1$ and the unique vertex $y'' \in S_1 \cap Y_0$ separates s' from s, s'' in G. Furthermore, $s''s, s''s' \notin E(G)$, and $s''y'' \in E(G)$ and y'' is the only neighbour of s'' in $Y_0 \cup S_0$.

Let us rename the vertices s, s', s'' to s_{12}, s_{23}, s_{13} and the vertices y, y', y'' to y_{12}, y_{23}, y_{13}. Then for distinct i, j, k we have $s_{ij} \in S_0 \cap Z_i \cap Z_j \cap Y_k$ and $S_k \cap Y_0 = \{y_{ij}\}$ and $N(s_{ij}) \cap$

$(Y_0 \cup S_0) = \{y_{ij}\}$. Note that this implies that $S_0 = \{s_{12}, s_{13}, s_{23}\}$ is an independent set. Moreover, $Y_0 \setminus \{y_{ij}\} \subseteq Z_k$, because all $y \in Y_0 \setminus \{y_{ij}\}$ are reachable in $G[Y_0 \cup S_0] \setminus \{y_{ij}\}$ by a path from $\{s_{ik}, s_{jk}\} \subseteq Z_k$.

As the separation (Y_0, S_0, Z_0) is non-degenerate and S_0 is an independent set, we have $|Y_0| > 1$. Since $N(S_0) = \{y_{12}, y_{23}, y_{13}\}$ and $N(y_{ij}) \cap S_0 = \{s_{ij}\}$ and G is 3-connected, it is easy to see that this implies that the vertices y_{ij} are mutually distinct. Now let $e = vw$ be an arbitrary edge of $G[Y_0]$. Such an edge exists, and it has an endvertex in each Z_k. Again, this is a contradiction. ◀

Let $W, X \subseteq V(G)$. Then a (W, X)-*separation* is a separation (Y, S, Z) such that $W \subseteq Y \cup S$ and $X \subseteq Z \cup S$. It is *proper* if $W \cap Y \neq \emptyset$ and $X \cap Z \neq \emptyset$. A (proper) (W, X)-separation (Y, S, Z) is *minimum* if its order is minimal, that is, there is no (proper) (W, X)-separation (Y', S', Z') such that $|S'| < |S|$. It is *leftmost minimum* if it is minimum and, subject to this condition, Y is inclusionwise minimal. It can be shown by a standard submodularity argument that there always is a unique leftmost minimum (W, X)-separation. There is not necessarily a unique leftmost minimum proper (W, X)-separation, but the number of such separations is (polynomially) bounded in terms of k.

▶ **Lemma 12.** *Let $k \geq 1$. Then there is a linear time algorithm that, given a graph G and sets $W, X \subseteq V(G)$, decides if there is a proper (W, X)-separation of order at most k, and if there is computes the set of all leftmost minimum proper (W, X)-separations.*

Let us say that a separation $(Y_0, S_0, Z_0) \in \mathrm{Sep}_{=3}(G)$ *defines* a tangle if (Y_0, S_0, Z_0) is non-degenerate and Z_0 is connected in G and (Y_0, S_0, Z_0) has no split vertex. Then the tangle *defined by* (Y_0, S_0, Z_0) is $\mathcal{T}(Y_0, S_0, Z_0)$ (of Lemma 9).

▶ **Lemma 13.** *There is an algorithm that, given a 3-connected graph G and a separation (Y_0, S_0, Z_0) of G of order 3 defining the tangle $\mathcal{T} = \mathcal{T}(Y_0, S_0, Z_0)$, computes the set of all non-degenerate separations in $\mathcal{T}_{\min}$ and the set of all non-degenerate crossedges of $\mathcal{T}$ in time $O(n(n+m))$.*

Proof. We show how to compute the set $\mathcal{T}_{\min}$; then we can easily filter out the non-degenerate separations.

Let $x \in Z_0$. Observe that if (Z, S, Y) is a proper $(S_0, \{x\})$-separation of order at most 3, then $(Y, S, Z) \in \mathcal{T}$. This follows immediately from the definition of $\mathcal{T}$. It implies the following equivalence for every separation (Y, S, Z) of G of order 3.

1. $(Y, S, Z) \in \mathcal{T}_{\min}$ and (Y, S, Z) does not cross (Y_0, S_0, Z_0).
2. There is an $x \in Z_0$ such that (Z, S, Y) is a leftmost minimum proper $(S_0, \{x\})$-separation.

We can use this equivalence to compute the set of all $(Y, S, Z) \in \mathcal{T}_{\min}$ such that (Y, S, Z) does not cross (Y_0, S_0, Z_0) (repeatedly applying the algorithm of Lemma 12 to all $x \in Z_0$). Note that the equivalence also gives us a linear bound on the number of such (Y, S, Z).

It remains to deal with the $(Y, S, Z) \in \mathcal{T}_{\min}$ crossing (Y_0, S_0, Z_0). For each $s \in S_0$ that has a unique neighbour $y \in Y_0 \cup S_0$, the edge sy may be a crossedge. This gives us at most three potential crossedges, and we deal with them separately. So let $s \in S$ and $y \in Y_0$ such that $N(s) \cap (Y_0 \cap S_0) = \{y\}$. Then for every separation $(Y, S, Z) \in \mathrm{Sep}_{=3}(G)$ the following are equivalent.

3. $y \in S$ and $\big(Z \cap (S_0 \cup Z_0), S \cap (S_0 \cup Z_0), Y \cap (S_0 \cup Z_0)\big)$ is a leftmost minimum proper $(S \setminus \{s\}, \{s\})$-separation in the graph $G[S_0 \cup Z_0]$.
4. $(Y, S, Z) \in \mathcal{T}_{\min}$ and (Y, S, Z) crosses (Y_0, S_0, Z_0) with crossedge ys.

To see this, note that (3) implies that $|S \cap Z_0| = 2$, because (Y_0, S_0, Z_0) has no split vertex. The equivalence between (3) and (4) allows us to compute the remaining separations in $\mathcal{T}_{\min}$.

As we have an overall linear bound on the number of separations in $\mathcal{T}_{\min}$, we can easily compute the set of non-degenerate crossedges. ◀

Let us a call a 3-separator S of G *degenerate* if there is a connected component C of $G \setminus S$ such that the separation $(G \setminus (S \cup V(C)), S, V(C))$ is degenerate. It is easy to see that this is the case if and only if S is an independent set and $G \setminus S$ has exactly two connected components, one of which has order 1.

▶ **Lemma 14.** *There is an a algorithm that, given a 3-connected graph G, decides if G has a non-degenerate 3-separator and computes one if there is in time $O(n^2(n+m))$.*

Proof. We first test if there is an $S \subseteq V(G)$ such that $|S| = 3$ and all connected components of $G \setminus S$ have order 1. In this case, S is a non-degenerate 3-separator if $|G| \geq 6$ or if $|G| \geq 5$ and S is not an independent set.

In the following, we assume that for every $S \subseteq V(G)$ such that $|S| = 3$ there is at least one connected components C of $G \setminus S$ such that $|C| \geq 2$. Now suppose that S is a non-degenerate 3-separator of G. Let Y be the vertex set of a connected component of G of size $|Y| \geq 2$, and let $Z := V(G) \setminus (S \cup Y)$. Let $y \in Y$ and $z \in Z$.

Then there is a leftmost minimum proper $(\{y\}, \{z\})$-separation (Y', S', Z') with $Y' \cup S' \subseteq Y \cup S$, because (Y, S, Z) is a minimum proper $(\{y\}, \{z\})$-separation. The separator S' is non-degenerate unless $Y' = \{y\}$ and S' is an independent set. Then $S' = N(y)$. However, in this case there is a leftmost minimum proper $(S', \{z\})$ separation (Y'', S'', Z'') such that S'' is non-degenerate. To see this, let $y' \in N(y) \cap Y$. Then there is a proper leftmost minimum $(S', \{z\})$ separation (Y'', S'', Z'') with $y, y' \in Y''$ and $Y'' \cup S'' \subseteq Y \cup S$, because (Y, S, Z) is a minimum proper $(S', \{z\})$ separation with $y, y' \in Y$. The set S'' is a non-degenerate 3-separator.

Thus we can find a non-degenerate 3-separator as follows. For all pairs y, z of distinct vertices, we compute all leftmost minimum proper $(\{y\}, \{z\})$-separations (Y', S', Z') and check if there is one such that S' is a non-degenerate 3-separator. If y has degree 3 and $S' := N(y)$ is an independent set, we also compute all leftmost minimum proper $(S', \{z\})$ separations (Y'', S'', Z'') and check if S'' is a non-degenerate 3-separator. ◀

Proof of Theorem 8. If G has no non-degenerate 3-separator, then G is quasi-4-connected, and we return the trivial tree decomposition with a one-node tree. In the following, we assume that G has at least one non-degenerate 3-separator.

We view the tree T in the tree decomposition as directed with all edges pointing away from the root, and we denote the descendant order in the tree by $\trianglelefteq$. With each (directed) edge $e = (s, t)$ of the tree we associate a separation $\mathrm{sep}(s, t) = (Y, S, Z)$ of order 3 such that Z is connected in G and $S = \beta(t) \cap \beta(s)$ and $S \cup Z = \bigcup_{u \geq t} \beta(u)$.

We build the tree decomposition iteratively starting from the root r of the tree. We pick an arbitrary non-degenerate 3-separator S_r of G and let $\beta(r) := S_r$. For every connected component C of $G \setminus S_r$ we create a child t of r, and we let $\mathrm{sep}(r, t) := (V(G) \setminus (S_r \cup V(C)), S_r, V(C))$.

At every step of the construction, we pick a leaf t of the current tree such that $\beta(t)$ is not yet defined. Let s be the parent of t and $\mathrm{sep}(s, t) = (Y_0, S_0, Z_0)$.

Case 1: $|Z_0| \leq 1$. Then $|S_0 \cup Z_0| \leq 4$, and we let $\beta(t) := S_0 \cup Z_0$. The node t will remain a leaf of the final tree.

Case 2: $|Z_0| > 1$ and (Y_0, S_0, Z_0) has a split vertex $z_0 \in Z_0$. Then we let $\beta(t) := S_0 \cup \{z_0\}$. For every connected component C of $G \setminus (S_0 \cup \{z_0\})$ with $V(C) \subseteq Z_0$ we create a child u of t and let $\mathrm{sep}(t,u) := (V(G) \setminus (N(C) \cup V(C)), N(C), V(C))$.

Case 3: $|Z_0| > 1$ and (Y_0, S_0, Z_0) has no split vertex. Let $\mathcal{T} = \mathcal{T}(Y_0, S_0, Z_0)$ be the G-tangle of Lemma 9. Note that $(Y_0, S_0, Z_0) \in \mathcal{T}_{\min}$. Let $R_{\mathcal{T}}$ be the quasi-4-connected region associated with $\mathcal{T}$. When contracting the non-degenerate crossedges that involve S_0, we make sure that we contract them to their endvertices in S_0. Then $S_0 \subseteq R_{\mathcal{T}}$ and $Y_0 \cap R_{\mathcal{T}} = \emptyset$. We let $\beta(t) := R_{\mathcal{T}}$.

For every connected component C of $G \setminus R_{\mathcal{T}}$ we create a child u of t and let $\mathrm{sep}(t,u) := (V(G) \setminus (N(C) \cup V(C)), N(C), V(C))$.

The completes the description of our construction. We need to describe a time $O(n^2(n+m))$-algorithm implementing it. By Lemma 14, we can compute a non-degenerate 3-separator S_r (for the root r) within this time if there is one.

Now we show that we can handle every step of the construction in time $O(n(n+m))$. So let t be a leaf of the current tree, s its parent, and $(Y_0, S_0, Z_0) := \mathrm{sep}(s,t)$. Case 1 is easy. For Case 2, we need to compute all connected components of $G \setminus (S_0 \cup \{z\})$ for all $z \in Z_0$, which we can do in time $O(n(n+m))$. For Case 3, we need to compute $\mathcal{T}_{\min}$ and $R_{\mathcal{T}}$ for the tangle $\mathcal{T} = \mathcal{T}(Y_0, S_0, Z_0)$, and Lemma 13 allows us to do this. ◀

Note that the results of Section 4, in particular the Correspondence Theorem 4, are used in Case 3 of the proof of Theorem 8 (and this is the only place in the proof where they are used).

▶ Remark. Let (T, β) be tree decomposition of G into quasi-4-connected components. The G-tangles of order 4 are associated with all nodes t such that either $|\beta(t)| \geq 5$ or $|\beta(t)| = 4$ and for each subset $S \subseteq \beta(t)$ of size $|S| = 3$ there is a neighbour u of t such that $\beta(u) \cap \beta(t) = S$. In the second case, the neighbours of t allow us to find a non-exceptional extension of the quasi-4-connected region $\beta(t)$.

6 Conclusions

Relaxing 4-connectedness, we introduce the notion of quasi-4-connectedness of graphs and prove that every graph has a decomposition into quasi-4-connected components. We show that the quasi-4-connected components correspond to the tangles of order 4, putting our result in the context of recent work on tangles and decompositions [1, 2, 3, 9, 11, 17]. Furthermore, we prove that our decomposition can be computed in cubic time. Although we do not explore this in the present paper, I believe that the decomposition may turn out to be a useful algorithmic tool, just like the decomposition into 3-connected components (though maybe not quite as broadly applicable).

The most obvious question is whether our result has a generalisation to "quasi-k-connected components", whatever they may be, for $k \geq 5$. I am skeptical, because we exploit many special properties of separators of order 3 here, most importantly the limited way in which they can cross. However, our decomposition is not a straightforward generalisation of the decomposition into 3-connected components either, and it may well be that new ideas lead to perfectly nice decompositions of higher order.

Finally, in particular when thinking of applications, it would be desirable to have a decomposition algorithm working in quadratic or even in linear time. I see no fundamental obstructions to the existence of such an algorithm.

References

1 J. Carmesin, R. Diestel, M. Hamann, and F. Hundertmark. Canonical tree-decompositions of finite graphs I. Existence and algorithms. *ArXiv*, arXiv:1305.4668v3 [math.CO], 2013.

2 J. Carmesin, R. Diestel, M. Hamann, and F. Hundertmark. Canonical tree-decompositions of finite graphs II. Essential parts. *ArXiv*, arXiv:1305.4909v2 [math.CO], 2013.

3 J. Carmesin, R. Diestel, F. Hundertmark, and M. Stein. Connectivity and tree structure in finite graphs. *Combinatorica*, 34(1):11–46, 2014.

4 R. Diestel. *Graph Theory*. Springer-Verlag, 3rd edition, 2005.

5 M. Grohe. Descriptive complexity, canonisation, and definable graph structure theory. Manuscript available at `http://www.lii.rwth-aachen.de/de/mitarbeiter/13-mitarbeiter/professoren/39-book-descriptive-complexity.html`.

6 M. Grohe. Fixed-point definability and polynomial time on graphs with excluded minors. *Journal of the ACM*, 59(5), 2012.

7 M. Grohe. Quasi-4-connected components. *ArXiv*, arXiv:1602.04505 [cs.DM], 2016. Full version of this paper.

8 M. Grohe. Tangles and connectivity in graphs. In A.-H. Dediu, J. Janoušek, C. Martín-Vide, and Bianca Truthe, editors, *Proceedings of the 10th International Conference on Language and Automata Theory and Applications*, volume 9618 of *Lecture Notes in Computer Science*, pages 24–41. Springer Verlag, 2016.

9 M. Grohe and P. Schweitzer. Computing with tangles. In *Proceedings of the 47th ACM Symposium on Theory of Computing*, pages 683–692, 2015.

10 J. E. Hopcroft and R. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(2):135–158, 1973.

11 F. Hundertmark. Profiles. An algebraic approach to combinatorial connectivity. *ArXiv*, arXiv:1110.6207v1 [math.CO], 2011.

12 S. MacLane. A structural characterization of planar combinatorial graphs. *Duke Mathematical Journal*, 3(3):460–472, 1937.

13 S. Makino. An algorithm for finding all the k-components of a digraph. *International Journal of Computer Mathematics*, 24(3-4):213–221, 1988.

14 H. Nagamochi and T. Ibaraki. *Algorithmic aspects of graph connectivity*. Cambridge University Press, 2008.

15 H. Nagamochi and T. Watanabe. Computing k-edge-connected components of a multigraph. *EICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E76-A(4):513–517, 1993.

16 N. Robertson and P.D. Seymour. Graph minors I–XXIII. *Journal of Combinatorial Theory, Series B* 1982–2012.

17 N. Robertson and P.D. Seymour. Graph minors X. Obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52:153–190, 1991.

18 N. Robertson and P.D. Seymour. Graph minors XIII. The disjoint paths problem. *Journal of Combinatorial Theory, Series B*, 63:65–110, 1995.

19 N. Robertson and P.D. Seymour. Graph minors XVI. Excluding a non-planar graph. *Journal of Combinatorial Theory, Series B*, 77:1–27, 1999.

20 R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

21 W.T. Tutte. *Graph Theory*. Addison-Wesley, 1984.

Subexponential Time Algorithms for Embedding H-Minor Free Graphs

Hans L. Bodlaender[1], Jesper Nederlof*[2], and Tom C. van der Zanden[3]

1 **Department of Computer Science, Utrecht University, Utrecht, The Netherlands; and Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands**
H.L.Bodlaender@uu.nl

2 **Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands**
J.Nederlof@tue.nl

3 **Department of Computer Science, Utrecht University, Utrecht, The Netherlands**
T.C.vanderZanden@uu.nl

Abstract

We establish the complexity of several graph embedding problems: Subgraph Isomorphism, Graph Minor, Induced Subgraph and Induced Minor, when restricted to H-minor free graphs. In each of these problems, we are given a pattern graph P and a host graph G, and want to determine whether P is a subgraph (minor, induced subgraph or induced minor) of G. We show that, for any fixed graph H and $\epsilon > 0$, if P is H-Minor Free and G has treewidth tw, (induced) subgraph can be solved $2^{O(k^\epsilon tw + k/\log k)} n^{O(1)}$ time and (induced) minor can be solved in $2^{O(k^\epsilon tw + tw \log tw + k/\log k)} n^{O(1)}$ time, where $k = |V(P)|$.

We also show that this is optimal, in the sense that the existence of an algorithm for one of these problems running in $2^{o(n/\log n)}$ time would contradict the Exponential Time Hypothesis. This solves an open problem on the complexity of Subgraph Isomorphism for planar graphs.

The key algorithmic insight is that dynamic programming approaches can be sped up by identifying isomorphic connected components in the pattern graph. This technique seems widely applicable, and it appears that there is a relatively unexplored class of problems that share a similar upper and lower bound.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases subgraph isomorphism, graph minors, subexponential time

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.9

1 Introduction

We study several problems related to recognizing a pattern graph P as substructure of a host graph G: Subgraph Isomorphism, Induced Subgraph, Graph Minor and Induced Minor. We consider the case in which P excludes a specific minor H, $\epsilon > 0$ is a constant and give algorithms parameterized by the treewidth tw of G and the number of vertices k of

* This work was done while the second author was visiting the Simons Institute for the Theory of Computing.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 9; pp. 9:1–9:14

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

P. Specifically, we show that for any $\epsilon > 0$ and graph H, if P is H-minor free and G has treewidth tw, (induced) subgraph can be solved $2^{O(k^\epsilon tw + k/\log k)} n^{O(1)}$ time and (induced) minor can be solved in $2^{O(k^\epsilon tw + tw \log tw + k/\log k)} n^{O(1)}$ time.

The $k/\log k$ dependence in the exponent is optimal: we present lower bounds based on the Exponential Time Hypothesis, showing that all these problems can not be solved in time $2^{o(n/\log n)}$, even when P is a tree and G is connected and series-parallel (and thus planar). Our lower bound answers a question of Marx [24] negatively: assuming the ETH, there is no $2^{O(\sqrt{k})} n^{O(1)}$ algorithm for subgraph isomorphism on planar graphs. This result is surprising, since for many problems on planar graphs a square root does appear in the exponent.

As an important special case of our result, we show that subgraph isomorphism can be solved in time $2^{O(k^\epsilon \sqrt{n} + k/\log k)}$ on H-Minor Free graphs (which includes planar, bounded-treewidth and bounded-genus graphs). Our result can be combined with a recent result of Fomin et al. [17] to show that subgraph isomorphism can be solved in $2^{O(k/\log k)} n^{O(1)}$ if P is connected and G is apex-minor free, which our lower bound shows is optimal (under ETH).

Subgraph isomorphism has received considerable attention in the literature. Results include polynomially solvable cases, such as recognizing a fixed pattern in planar graphs [13, 15], biconnected outerplanar graphs [22], graphs of log-bounded fragmentation [20] and graphs of bounded genus [10] and certain subclasses of graphs of bounded treewidth [26], exact algorithms [30], lower bounds [19, 11, 28] and results on parameterized complexity [25].

For a pattern graph P of treewidth t, subgraph isomorphism is solvable in $2^{O(k)} n^{O(t)}$ time using the color-coding technique [3]. If the host graph is planar, subgraph isomorphism can be solved in $2^{O(k)} n$ time [13]. In general graphs, subgraph isomorphism can be solved in $2^{O(n \log n)}$ time and, assuming the ETH, this is tight [16].

Graph minor problems are also of interest, especially in the light of Robertson and Seymour's seminal work on graph minor theory (see e.g. [29]) and the recent development of bidimensionality theory [12]. Many graph properties can be tested by checking for the inclusion of some minor. Testing whether a graph G contains a fixed minor P can be done in $O(n^3)$ time [27], this was recently improved to $O(n^2)$ [21]. However, the dependence on n is superexponential in a strong sense. Testing whether a graph P is a minor of a planar graph G can be done in $2^{O(k)} n^{O(1)}$ time [1], which is only single-exponential. Our lower bound shows this running time can not be improved to $2^{o(k/\log k)}$, assuming ETH. Our algorithms are subexponential in k, but (in contrast to [1, 21]) superpolynomial in n. This is to our knowledge the first subexponential minor testing algorithm (for a non-trivial class of graphs).

Our algorithms are based on dynamic programming on tree decompositions. In particular, we use dynamic programming on the host graph and store correspondences to vertices in the pattern graph. The key algorithmic insight is that this correspondence may or may not use certain connected components of the pattern graph (that remain after removing some separator vertices). Instead of storing for each component whether it is used or not, we identify isomorphic connected components and store only the number of times each is used.

In [20], the authors give an algorithm for subgraph isomorphism, which runs in polynomial time for a host graph of bounded treewidth and a pattern graph of log-bounded fragmentation (i.e. removing a separator decomposes the graph into at most logarithmically many connected components). This is achieved using a similar dynamic programming technique, which (in general) uses time exponential in the number of connected components that remain after removing a separator. By assuming the number of connected components (fragmentation) is logarithmic, the authors obtain a polynomial time algorithm. In contrast, we consider a graph class where fragmentation is unbounded, but the number of non-isomorphic connected components is small. This leads to subexponential algorithms.

This paper builds on techniques due to Bodlaender, Nederlof and van Rooij [7, 9]. They give a $2^{O(n/\log n)}$-time algorithm for finding tree decompositions with few bags and a matching lower bound (based on the Exponential Time Hypothesis), and a $2^{O(n/\log n)}$ algorithm for determining whether a given k-colored graph is a subgraph of a properly colored interval graph. These earlier papers, coupled with our results, suggest that this technique may have many more applications, and that there exists a larger class of problems sharing this upper and lower bound.

2 Preliminaries

Graphs. Given a graph G, we let $V(G)$ denote its vertex set and $E(G)$ its edge set. Given $X \subseteq V(G)$, let $G[X]$ denote the subgraph of G induced by X and use shorthand $E(X) = E(G[X])$. Let $Nb(v)$ denote the neighbourhood of v, that is, the vertices adjacent to v. For a set of vertices S, let $Nb(S) = \cup_{v \in S} Nb(v) \setminus S$. Let $CC(G)$ denote the set of the connected components of G. Given $X \subseteq V(G)$, we write as shorthand $CC(X) = CC(G[X])$.

Functions. Given a function $f : A \to B$, we let $f^{-1}(b) = \{a \in A \mid f(a) = b\}$. Depending on the context, we may also let $f^{-1}(b)$ denote (if it exists) the unique $a \in A$ so that $f(a) = b$. We say $g : A \to B$ is a *restriction* of $f : A' \to B'$ if $A \subseteq A'$ and $B \subseteq B'$ and for all $a \in A$, $g(a) = f(a)$. We say g is an *extension* of f if f is a restriction of g.

Isomorphism. We say a graph P is isomorphic to a graph G if there is a bijection $f : V(P) \to V(G)$ so that $(u, v) \in E(P) \iff (f(u), f(v)) \in E(G)$. We say a graph P is a subgraph of G if we can obtain a graph isomorphic to P by deleting edges and or vertices from G, and we say P is an induced subgraph if we can obtain it by deleting only vertices.

Contractions, minors. We say a graph G' is obtained from G by contracting edge (u, v), if G' is obtained from G by replacing vertices u, v with a new vertex w which is made adjacent to all vertices in $Nb(u) \cup Nb(v)$. A graph G' is a minor of G if a graph isomorphic to G' can be obtained from G by contractions and deleting vertices and/or edges. G' is an induced minor if we can obtain it by contractions and deleting vertices (but not edges).

Tree decompositions. A tree decomposition of a graph G is a rooted tree T with for every vertex $i \in V(T)$ a bag $X_i \subseteq V(G)$, such that $\cup_{i \in V(T)} X_i = V(G)$, for all $(u, v) \in E(G)$ there is an $i \in V(T)$ so that $\{u, v\} \subseteq X_i$ and for all $v \in V(G)$, $T[\{i \in V(T) \mid v \in X_i\}]$ is connected. The *width* of a tree decomposition is $\max_{i \in V(T)} |X_i| - 1$ and the *treewidth* of a graph G is the minimum width over all tree decompositions of G. For a node $t \in T$, we let $G[t]$ denote the subgraph of G induced by the vertices contained in the bags of the subtree of T rooted at t.

To simplify our algorithms, we assume that a tree decomposition is given in *nice* form, where each node is of one of four types:

- **Leaf**: A leaf node is a leaf $i \in T$, and $|X_i| = 1$.
- **Introduce**: An introduce node is a node $i \in T$ that has exactly one child $j \in T$, and X_i differs from X_j only by the inclusion of one additional vertex.
- **Forget**: An introduce node is a node $i \in T$ that has exactly one child $j \in T$, and X_i differs from X_j only by the removal of one vertex.
- **Join**: A join node is a node $i \in T$ with exactly two children $j, k \in T$, so that $X_i = X_j = X_k$.

A tree decomposition can be converted to a nice tree decomposition of the same width and of linear size in linear time [5].

3 Algorithmic Results

We begin by describing our algorithm for subgraph isomorphism, which is based on dynamic programming on a tree decomposition T of the host graph G. The algorithm is somewhat similar to that of Hajiaghayi et al. [20] for subgraph isomorphism on log-bounded fragmentation graphs, and we use similar notions of (extensible) partial solutions and characteristic of a partial solution (Section 3.1). Our main contribution is the canonization technique (Section 3.2) and its analysis (Section 3.3), which gives the subexponential running time.

3.1 An algorithm for Subgraph Isomorphism

▶ **Definition 1** ((Extensible) Partial Solution)**.** For a given node $t \in T$ of the tree decomposition of G, a *partial solution (relative to t)* is a triple (G', P', ϕ) where G' is a subgraph of $G[t]$, P' is an induced subgraph of P and $\phi : V(G') \to V(P')$ is an isomorphism from G' to P'.

Say that a partial solution (G', P', ϕ) relative to t is *extensible* if there exists an extension of ϕ, $\psi : V(G'') \to V(P)$ which is an isomorphism from a subgraph G'' of G to P where $V(G'') \cap V(G[t]) = V(G')$.

To facilitate dynamic programming, at node t of the tree decomposition we only consider partial solutions (G', P', ϕ) which *might* be extensible (i.e. we attempt to rule out non-extensible solutions). Note that in a partial solution we have already decided on how the vertices in $G[t]$ are used, and the extension only makes decisions about vertices not in $G[t]$. Instead of dealing with partial solutions directly, our algorithm works with characteristics of partial solutions:

▶ **Definition 2** (Characteristic of a Partial Solution)**.** The characteristic (f, S) of a partial solution (G', P', ϕ) relative to a node $t \in T$ is a function $f : X_t \to V(P) \cup \{\square\}$, together with a subset $S \subseteq V(P) \setminus f(X_t)$, so that:

- for all $v \in V(G') \cap X_t$, $f(v) = \phi(v)$ and $f(v) = \square$ otherwise,
- f is injective, except that it may map multiple elements to $\square$,
- $S = V(P') \setminus \phi(X_t)$.

The following easy observation justifies restricting our attention to characteristics of partial solutions:

▶ **Lemma 3** (Equivalent to Lemma 10, [20])**.** *If two partial solutions have the same characteristic, either both are extensible or neither is extensible.*

▶ **Lemma 4.** *If (f, S) is the characteristic of an extensible partial solution (G', P', ϕ) relative to a node $t \in T$, then S is a union of connected components of $P[V(P) \setminus f(X_t)]$.*

Proof (due to Hajiaghayi et al. [20]). Suppose there exist adjacent vertices $v_1, v_2 \in V(P) \setminus \phi(X_t)$ and $v_1 \in V(P')$, $v_2 \notin V(P')$. Then it is never possible to find a preimage u for v_2 in an extension of (G', P', ϕ): we require that $u \notin V(G[t])$, but all vertices adjacent to $\phi^{-1}(v_1)$ are contained in $V(G[t])$. ◀

The latter fact will be key to achieving the subexponential running time. The requirement that S is a union of connected components also appears in the definition of 'good pair' in Bodlaender et al. [7]. We show how to compute the characteristics of partial solutions in

Procedure 1 Leaf case: computes the partial solution characteristics for a leaf bag $t \in T$, with $X_t = \{v\}$.

let $R = \emptyset$
for each $u \in V(P) \cup \{\square\}$ **do**
    let $f : X_t \to V(P) \cup \{\square\}$ be the function so that $f(v) = u$
    let $R = R \cup \{(f, \emptyset)\}$
end for
filter R
return R

Procedure 2 Introduce case: introduces a vertex v into a bag X_t.

let R be the set of partial solution characteristics for t
let $R' = \emptyset$
for each $(f, S) \in R$ and each $u \in V(P) \setminus (f(X_t) \cup S) \cup \{\square\}$ **do**
    if $u = \square$ **or** for all $w \in Nb(u) \cap f(X_t), (v, f^{-1}(w)) \in E(G)$ **then**
        let $f' : X_t \cup \{v\} \to V(P) \cup \{\square\}$ be the extension of f so that $f(v) = u$
        let $R' = R' \cup \{(f', S)\}$
    end if
end for
filter R'
return R'

Procedure 3 Forget case: forgets a vertex v from a bag X_t.

let R be the set of partial solution characteristics for t
let $R' = \emptyset$
for each $(f, S) \in R$ **do**
    let f' be the restriction of f to $X_t \setminus \{v\}$
    if $f(v) = \square$ **or** $f(v)$ is not adjacent to any vertex of $V(P) \setminus (f(X_t) \cup S)$ **then**
        let $R' = R' \cup \{(f', S \cup \{f(v)\} \setminus \{\square\})\}$
    end if
end for
filter R'
return R'

Procedure 4 Join case: combines the partial solution characteristics for two bags $X_s = X_t$.

let R be the set of partial solution characteristics for s
let T be the set of partial solution characteristics for t
let $R' = \emptyset$
for each $(f, S) \in R$ and each $(g, Q) \in T$ **do**
    if $f = g$ **and** $S \cap Q = \emptyset$ **then**
        let $R' = R' \cup \{(f, S \cup Q)\}$
    end if
end for
filter R'
return R'

a bottom-up fashion, so that we can tell whether G has a subgraph isomorphic to P by examining the characteristics of the root bag. We proceed by giving pseudocode for the leaf, introduce, forget and join cases and argue for their correctness.

The correctness of Procedure 1 is self-evident, as we simply enumerate all the possibilities for f (which means guessing a vertex in P to map v to). We will give details of the *filter* procedure in the next section, for now it suffices to treat the pseudocode as if this call were not present.

Procedure 2 extends existing partial solutions by choosing a vertex to map v to. To ensure we obtain valid characteristics of partial solutions, we check that for any edge incident to v in $P[S \cup f(X_t)]$ there is a corresponding edge in G. Because S is a union of connected components of $G[V(P) \setminus f(X_t)]$, $f(v)$ can not be adjacent to any vertex in S, and thus it suffices to check adjacency only against vertices in $f(X_t)$. Then S remains a union of connected components since the removal of a vertex can only further disconnect S. Note that u is chosen so that f remains injective.

Procedure 3 discards any solutions that would result in S not remaining the union of connected components that we require after forgetting a vertex (note that this means we keep only partial solutions were we have already chosen preimages for all of the neighbours of the image of the vertex being forgotten).

Finally, consider Procedure 4. Because (as a basic property of nice tree decompositions) $V(H[i]) \cap V(H[j]) = X_i$, we obtain an injective function if and only if $S \cap R = \emptyset$. We can therefore merge two partial solutions if they map the vertices of $X_t = X_s$ in the same way and $S \cap R = \emptyset$. Note that we do indeed create all possible partial solutions in this way: given a partial solution, we can split it into partial solutions for the left and right subtrees since (as there are no edges between the internal vertices of the left and right subtrees) a connected component of S must be covered entirely by either the left or right subtree.

These procedures, applied bottom-up on the tree decomposition, give an algorithm that decides subgraph isomorphism. Note that if one exists, a solution can be reconstructed from the characteristics.

3.2 Reducing the number of partial solutions using isomorphism tests

In this section, we show how adapt the algorithm from the previous section to achieve the claimed running time bound. This involves a careful analysis of the number of characteristics, and using isomorphism tests to reduce this number. Currently, if the connected components of S are small (e.g., $O(1)$ vertices each) then their number is large (e.g., $\Omega(n)$ components) and thus in the worst case we have $2^{\Omega(n)}$ partial solutions. However, if there are many small connected components many will necessarily be isomorphic to each other (since there are only few isomorphism classes of small connected components) and we can thus reduce the number of characteristics by identifying isomorphic connected components:

▶ **Definition 5** (Partial Solution Characteristic Isomorphism)**.** Given a bag $t \in T$, two characteristics of partial solutions $(f, S), (g, R)$ for t are *isomorphic* if:

- $f = g$,
- there is a bijection $h : CC(S) \to CC(R)$,
- for all connected components $c \in CC(S)$, c and $h(c)$ are isomorphic when all vertices $v \in c$ vertices are labelled with $Nb(v) \cap f(X_t)$ (i.e. the set of vertices of $f(X_t)$ to which v is adjacent).

Clearly, the algorithm given in the previous section remains correct even if after each procedure we remove duplicate isomorphic characteristics. To this end, we modify the join

Procedure 5 Connected Component Canonization: Computes a canonical representation for a union of connected components $S \subseteq V(P) \setminus f(X_t)$

1: let S' be the union of the large connected components of S
2: let $Q = \emptyset$
3: **for** each small connected component s of S **do**
4: compute the canonical representation r of s when each $v \in V(s)$ is labelled with $Nb(v) \cap f(X_t)$
5: let $Q = Q \cup \{r\}$
6: **end for**
7: Sort S' and Q lexicographically
8: **return** (S', Q)

Procedure 7 Filtering Procedure: Filters a set of partial solution characteristics R to remove duplicates

1: compute the canonical representation C_S for every $(f, S) \in R$ using Procedure 5
2: **sort** R first by f, then by C_S in lexicographical order
3: loop over R, removing all but one of each group of isomorphic partial solutions
4: **return** R

case (Procedure 4): the disjointness check $S \cap Q = \emptyset$ should be replaced with a check that if $P[V(P) \setminus f(X_t)]$ contains $N_P(y)$ connected components of isomorphism class y, and $P[S]$ (resp. $P[Q]$) contains $N_S(y)$ (resp. $N_Q(y)$) connected components of isomorphism class y, then $N_S(y) + N_Q(y) \leq N_P(y)$. Similarly, the statement $S \cup Q$ needs to be changed to, if the union is not disjoint, replace connected components that occur more than once with other connected components of the same isomorphism class (so as to make the union disjoint while preserving the total number of components of the same type).

Call a connected component *small* if it has at most $c \log k$ vertices, and large otherwise. We let $c > 0$ be a constant that depends only on $|V(H)|$ and ϵ. We do not state our choice of c explicitly, but in our analysis we will assume it is "small enough".

For a small connected component s, we label each of its vertices by the subset of vertices of $f(X_t)$ to which it is adjacent. We then compute a canonical representation of this labeled component, for example by considering all permutations of its vertices, and choosing the permutation that results in the lexicographically smallest representation. Note that since we only canonize the small connected components using such a trivial canonization algorithm does not affect the running time of our algorithm, as $(c \log k)!$ is only slightly superpolynomial.

Procedure 5 computes a canonical representation of a partial solution. It requires that we have some predefined ordering of the vertices of G. The canonization procedure 5 allows us to define the *filter* procedure (Procedure 6).

Traditionally, a canonization is a function that maps non-isomorphic graphs to distinct strings, and isomorphic graphs to identical strings. We use this term slightly more loosely, as our canonization procedure 5 may map isomorphic graphs to distinct strings since it only canonizes the small connected components. Thus, Procedure 6 may not remove all duplicate isomorphic partial solutions. However, we will show that it removes enough of them.

3.3 Bounding the number of non-isomorphic partial solutions

In this section, we analyse the number of non-isomorphic partial solutions, and show that the algorithm given in the previous section indeed achieves the stated time bound. In the following, let $\epsilon > 0$ and let G be a graph of treewidth at most tw. Furthermore, suppose that P is H-minor free for some fixed graph H.

Recall that a partial solution for a node $t \in T$ of the tree decomposition consists of $f : X_t \to V(P) \cup \{\square\}$ and a subset $S \subseteq V(P) \setminus f(X_t)$, which is a union of connected components of the subgraph induced by $S \subseteq V(P) \setminus f(X_t)$. The number of choices for f is at most $(k+1)^{|X_t|} = 2^{O(tw \log k)}$. We now proceed to bound the number of cases for S.

We distinguish between connected components of $V(P) \setminus f(X_t)$ of which there are "few", and connected components of $V(P) \setminus f(X_t)$ of which there can be "many", but few non-isomorphic ones. For some constant c, we say a component is *small* if it has at most $c \log k$ vertices, and *large* otherwise. The large connected components are amongst the few, since there are at most $k/(c \log k)$ components with at least $c \log k$ vertices. For each of these components, we store explicitly whether or not it is contained in S. They contribute a factor of $2^{O(k/\log k)}$ to the number of cases. For the small connected components, we will show a partition into the "few" (which we treat similarly to the large connected components), and into the "many, but few non-isomorphic" (for which we store, for each isomorphism class, the number of components from that isomorphism class contained in S).

▶ **Claim.** *For a given node t and function $f : X_t \to V(P)$, there are at most $O(k^{\epsilon/2}tw)$ isomorphism classes of small connected components.*

Proof. For a (small) connected component $x \in CC(V(P) \setminus f(X_t))$, its isomorphism class is determined by the isomorphism class of x itself, and the adjacency of vertices $v \in x$ to vertices in $f(X_t)$. Since $|x| \leq c \log k$ and P is H-minor free, there exists a constant $C_H > 1$ so that there are at most $2^{C_H \cdot c \log k}$ cases for the isomorphism class of x itself (see [4]).

What remains is to bound the number of cases for adjacency of x to X_t. In this specific case, $Nb(x)$ denotes the set of vertices of X_t to which x is incident, that is, $v \in Nb(x)$ if and only if $v \in X_t$ and there exists a vertex $u \in x$ so that $(u, v) \in E(P)$. Using the following lemma, we further divide the small connected components into two cases: the components with a large neighbourhood, and the components with a small neighbourhood.

▶ **Lemma 6** (Gajarskỳ et al., special case of Lemma 3 of [18]). *Let H be a fixed graph. Then there exists a constant d (depending on H), so that if $G = (A, B, E)$ is H-minor free and bipartite, there are at most*

- *$O(|A|)$ vertices in B with degree greater than d,*
- *$O(|A|)$ subsets $A' \subseteq A$ such that $A' = Nb(u)$ for some $u \in B$.*

Taking $A = X_t$, deleting the edges between vertices in X_t and contracting every connected component $x \in CC(V(P) \setminus f(X_t))$ to a single vertex in B, the lemma states that there are at most $O(tw)$ components with $|Nb(x)| > d$ and that the components with $|Nb(x)| \leq d$ have at most $O(tw)$ distinct neighbourhoods in X_t.

For the connected components $x \in CC(V(P) \setminus f(X_t))$ with $|Nb(x)| \leq d$, we have $2^{C_H \cdot c \log k}$ cases for the isomorphism class of x, $O(tw)$ cases for $Nb(x)$ and for every vertex of x, at most 2^d cases for incidence to $Nb(x)$. We thus have at most $2^{C_h \cdot c \log k} \cdot O(tw) \cdot (2^d)^{c \log k}$ isomorphism classes for $x \in CC(S)$ with $Nb(x) \leq d$. For sufficiently small $c > 0$, the asymptotic complexity is $O(k^{\epsilon/2}tw)$. ◀

Since of each component there can be most k occurrences in S, the total number of cases for storing the multiplicity of each class of components is $(k+1)^{O(k^{\epsilon/2}tw)} = 2^{O(k^{\epsilon/2}tw \log k)} = 2^{O(k^{\epsilon}tw)}$.

We now have all the results we need to finish the analysis. Storing the multiplicities of the small connected components gives $2^{O(k^{\epsilon}tw)}$ cases, while storing the subset of large connected components explicitly contributes $2^{O(k/\log k)}$ cases. A partial solution is further characterized by f, for which there are only $2^{O(tw \log k)}$ cases. For a given node t of the tree decomposition, there are thus at most $2^{O(k^{\epsilon}tw + k/\log k)}$ partial solutions.

Finally, we can compute a 5-approximate tree-decomposition of G in time exponential in tw [6], perform dynamic programming as described in Procedures 1-6 to obtain:

▶ **Theorem 7.** *For any graph H and $\epsilon > 0$,* Subgraph Isomorphism *can be solved in time $2^{O(k^{\epsilon}tw+k/\log k)}n^{O(1)}$ if the host graph has treewidth tw and the pattern graph is H-minor free.*

3.4 Adaptation to other problems

In this section, we discuss how our algorithm for Subgraph Isomorphism can be adapted to Induced Subgraph and (Induced) Minor. We begin by describing the graph minor case, then give a brief note on how to adapt both algorithms for the induced case. Some details are omitted from this extended abstract.

Note that P is a minor of G if and only if we have a function $f : V(G) \to V(P) \cup \{\square\}$, such that

- for all $v \in V(P)$, $f^{-1}(v)$ is non-empty, and induces a connected subgraph of G,
- for all $(v, w) \in E(P)$, there are $x \in f^{-1}(v)$ and $y \in f^{-1}(w)$ with $(x, y) \in E(G)$.

Vertices that are deleted are mapped to $\square$, otherwise $f(v)$ gives the vertex that v is contracted to. Call such a function a *solution* for the Graph Minor problem.

If we restrict such solutions to a subgraph $G[t]$, we obtain the notion of partial solution:

▶ **Definition 8** (Partial Solution (Graph Minor))**.** Given a node $t \in T$ of the tree decomposition of G, a *partial solution for the* Graph Minor *problem relative to a node t* is a function $f : V(G[t]) \to V(P) \cup \{\square\}$, such that

1. For each $v \in V(P)$, at least one of the following three cases holds:
   a. each connected component of $G[t][f^{-1}(v)]$ contains at least one vertex from X_t,
   b. $G[t][f^{-1}(v)]$ has one connected component,
   c. $f^{-1}(v)$ is empty.
2. For all $(v, w) \in E(P)$, at least one of the following cases holds:
   a. Some vertex of $f^{-1}(v)$ is adjacent to some vertex of $f^{-1}(w)$ in $G[t]$,
   b. $f^{-1}(v) \cap X_t \neq \emptyset$ and $f^{-1}(w) \cap X_t \neq \emptyset$,
   c. $f^{-1}(v) \cap X_t \neq \emptyset$ and $f^{-1}(w) = \emptyset$,
   d. $f^{-1}(w) \cap X_t \neq \emptyset$ and $f^{-1}(v) = \emptyset$.
   e. $f^{-1}(w) = \emptyset$ and $f^{-1}(v) = \emptyset$.

Intuitively, in 1) we require that the preimage of v can still be made connected (a), is already connected (b) or has not been assigned yet (c), and in 2) we require that the edge (v, w) is already covered (a), can still be covered (b,c,d,e).

As before, our dynamic programming algorithm uses characteristics of partial solutions:

▶ **Definition 9** (Characteristic of a partial solution (Graph Minor))**.** Given a node $t \in T$ of the tree decomposition of G and a *partial solution for the* Graph Minor *problem relative to a node t* $f : V(G[t]) \to V(P) \cup \{\square\}$, the *characteristic* of f is a tuple $(f', S, \sim, F)$ such that:

1. f' is the restriction of f to X_t,
2. $S \subseteq V(P)$ with $S = f(V(G[t])) \setminus (f(X_t) \cup \{\square\})$,
3. $\sim$ is an equivalence relation on X_t, with $v \sim w$, if and only if $f(v) = f(w)$ and there exists a path from v to w in $G[t]$ such that for all vertices x on this path $f(x) = f(v)$.
4. $F \subseteq E(P[f(X_t)])$ such that for every $(v, w) \in E(P[f(X_t)])$, it holds that $(v, w) \in F$ if and only if each of the following holds:
   a. $f^{-1}(v) \cap X_t \neq \emptyset$,
   b. $f^{-1}(w) \cap X_t \neq \emptyset$,
   c. There are $x \in f^{-1}(v)$ and $y \in f^{-1}(w)$ with $(x, y) \in E(G[t])$.

As before, it is easy to see the equivalence between the existence of a minor, solution, and partial solution with certain characteristic.

Compared to our approach for subgraph isomorphism, we no longer require f to be injective – $f^{-1}(v)$ corresponds to the vertices that are contracted to form v. We require that $f^{-1}(v)$ eventually becomes connected. This can either be achieved inside the bag, be achieved below the bag (in the tree decomposition), or above the bag. The relation $\sim$ tracks which components are already connected by vertices below the bag. Similarly, edges inside $P[f(X_t)]$ might not have corresponding edges inside $G[X_t]$, but might instead correspond to edges below or above this bag. The set F stores which edges correspond to edges below the current bag.

This lemma is the counterpart of Lemma 3 and shows that we can apply dynamic programming:

▶ **Lemma 10.** *If partial solutions for the* Graph Minor *problem f and g have the same characteristic and are both relative to t, then f can be extended to a solution if and only if g can be extended to a solution.*

The following lemma shows that we can apply our technique of reducing the number of partial solution characteristics by using isomorphisms:

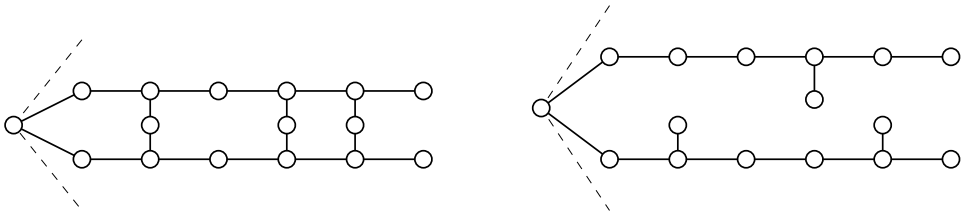
▶ **Lemma 11.** *A partial solution for* Graph Minor *f with characteristic $(f', S, \sim, F)$ can be extended to a solution only if S is a union of connected components of $G[V(P) \setminus f(X_t)]$.*

The analysis of the number of cases of f' and S remains unchanged. There are at most $(tw)^{tw} = 2^{O(tw \log tw)}$ cases for $\sim$, and since P is sparse, at most $2^{O(tw)}$ cases for F.

For the induced cases, only a small modification is needed: it suffices to check in the introduce case that all neighbours (in X_t) of the vertex being introduced are mapped to vertices that are adjacent to the image of the introduced vertex and discard the partial solution otherwise. We thus obtain the following theorem:

▶ **Theorem 12.** *For any graph H and $\epsilon > 0$, if the host graph has treewidth tw and the pattern graph is H-minor free,* Subgraph Isomorphism *and* Induced Subgraph *can be solved in time $2^{O(k^\epsilon tw + k/\log k)} n^{O(1)}$ and* Graph Minor *and* Induced Minor *can be solved in time $2^{O(k^\epsilon tw + tw \log tw + k/\log k)} n^{O(1)}$.*

As a direct corollary, we have that Subgraph Isomorphism, Graph Minor, Induced Subgraph and Induced Minor can be solved in $2^{O(n^{0.5+\epsilon} + k/\log k)}$ time if the host graph is H-minor free for some fixed graph H, as H-minor free graphs have treewidth $O(\sqrt{n})$ [2]. Important special cases include planar graphs, graphs of bounded genus, and graphs of bounded treewidth.



(a) Host graph **(b)** Guest graph

Figure 1 Construction used in the proof of Theorem 14. Host graph (a), representing a string $101001 \in A$ and pattern graph (b), representing strings $000100 \in B$ and $010010 \in C$. The dashed lines represent additional paths attached to vertices u and u'.

4 Hardness Results

Both (Induced) Subgraph and (Induced) Minor are known to be *NP*-complete, even if P is a tree and G is series-parallel (and thus planar), connected and all but one vertex of P and G have degree at most 3, by reduction from Three-Dimensional Matching [26].

We obtain our $2^{o(n/\log n)}$ lower bound by a reduction very similar to the reduction from Three-Dimensional Matching in [26]. We instead reduce from the String 3-Groups problem [7]. In the following, given a string s, we let s_i denote the i^{th} character of s.

String 3-Groups

Instance: Sets $A, B, C \subseteq \{0,1\}^{6\lceil \log n \rceil + 1}$, $|A| = |B| = |C| = n$

Question: Do there exist n triples, so that for each chosen triple $(a, b, c) \in A \times B \times C$ and for all i, $a_i + b_i + c_i \leq 1$ and each element of A, B, C occurs in exactly one triple?

▶ **Theorem 13** ([7], [8]). *Assuming the Exponential Time Hypothesis, there is no algorithm solving* String 3-Groups *in* $2^{o(n)}$ *time.*

▶ **Theorem 14.** *Assuming the Exponential Time Hypothesis, there is no algorithm solving* Subgraph Isomorphism *in* $2^{o(n/\log n)}$ *time, even when the pattern graph is a tree and the host graph is connected and series-parallel and in both the host graph and pattern graph, all vertices but one have maximum degree 3.*

Proof. Let A, B, C be an instance of String 3-Groups. We modify the instance by prepending a 0 to each string in A and B, and a 1 to each string in C. Let $m = 6\lceil \log n \rceil + 2$.

To construct the host graph G, we take a vertex u. For each $a \in A$ we take a path $p_1, \ldots, p_m$ and a path $q_1, \ldots, q_m$. We add edges (p_1, u) and (q_1, u). Whenever $a_i = 0$, we create a vertex r_i and edges $(p_i, r_i), (r_i, q_i)$.

To construct the pattern graph P, we take a vertex u'. For each $b \in B$ (resp. each $c \in C$) we take a path $s_1, \ldots, s_m$. We add an edge (s_1, u'). Whenever $b_i = 1$ (resp. $c_i = 1$), we create a vertex t_i and an edge (s_i, t_i).

A solution to the String 3-Groups problem and this instance of Subgraph Isomorphism correspond as follows: u is mapped to u'. If (a, b, c) is a triple in a solution to String 3-Groups, then the path $s_1, \ldots, s_m$ corresponding to b can be mapped to the path $p_1, \ldots, p_m$ corresponding to a, while the path $s_1, \ldots, s_m$ corresponding to c is mapped to the path $q_1, \ldots, q_m$. Clearly such a mapping is possible if and only if for each i, at most one of a_i, b_i or c_i is 1, since the vertex v_i only exists if $a_i = 0$ and can be used at most once (by either the vertex t_i corresponding to b_i or the vertex t_i corresponding to c_i).

In the reverse direction, note that any subgraph isomorphism must map u to u' by virtue of their degrees. Clearly, the paths in H must correspond one-to-one with paths in G. The correspondence of the paths immediately gives us a partition into triples. The construction enforces that in each such triple, in each position at most one of the strings has a 1, as required. Note that since we modified the instance so that each string $c \in C$ starts with a 1, no triple will contain more than one string from c and consequently, each triple contains exactly one string from each of a, b, c.

Since the graph created in the reduction has $O(n \log n)$ vertices, and assuming the Exponential Time Hypothesis there is no $2^{o(n)}$ algorithm for String 3-Groups, there is no $2^{o(n/\log n)}$ time algorithm for Subgraph Isomorphism, even for the graph classes as claimed in the theorem. ◀

The proof of Theorem 14 can be adapted to Induced Subgraph by subdividing each edge (p_i, r_i) once. Furthermore, the proof also works for (Induced) Graph Minor, since performing a contraction in H is never beneficial.

▶ **Theorem 15.** *Assuming the Exponential Time Hypothesis, there is no algorithm solving* Subgraph Isomorphism, Graph Minor, Induced Subgraph *and* Induced Minor *in* $2^{o(n/\log n)}$ *time, even when the pattern graph is a tree and the host graph is connected and series-parallel and in both the host graph and pattern graph, all vertices but one have maximum degree 3.*

5 Conclusion

We have presented algorithms for (Induced) Subgraph and (Induced) Minor that, by taking advantage of isomorphic structures in the pattern graph, run in subexponential time on H-minor free graphs. These algorithms are essentially optimal since we have shown that the existence of $2^{o(n/\log n)}$ time algorithms would contradict the Exponential Time Hypothesis. We have thus settled the (traditional) complexity of these problems on (general) H-Minor Free graphs.

Our result applies to a wide range of graphs: we require P to be H-Minor Free and G to have truly sublinear treewidth. Some restriction on G is indeed necessary, since if G is an arbitrary graph then Hamiltonian path is a special case (in which P is a path) and a $2^{o(n)}$ algorithm would contradict the ETH [23].

An interesting open question is whether the parameterized complexity can still be improved. Perhaps the dependence of the running time on the treewidth of G can be removed: does there exist an algorithm for subgraph isomorphism on planar graphs running in $2^{O(k/\log k)}n^{O(1)}$? Our result, combined with one due to Fomin et al. [17] implies that the answer to this question is yes if P is connected and G is apex-minor free, but the problem remains open otherwise.

We note that our lower bound proof also works for Immersion [14]. However, our algorithmic technique does not seem to work for immersion. Does the immersion problem also have a $2^{O(n/\log n)}$ algorithm, or is a stronger lower bound possible?

Lemma 6 holds for a more general class of graphs, and we believe it may be possible to extend our result to patterns from a graph class with expansion $O(1)$ or perhaps expansion $O(\sqrt{r})$. We note that for different graph classes, a tradeoff between the size of the small connected components and the factor $k/\log k$ in the exponent is possible: it might be possible to obtain a $2^{O(n/\log\log n)}$-time algorithm for less restrictive graph classes.

Together with the Minimum Size Tree/Path Decomposition problem [7], these problems are amongst the first for which a $2^{\Theta(n/\log n)}$ upper and lower bound is known. Our work shows that the techniques from [7] can be adapted to other problems, and we suspect there may be many more problems for which identifying isomorphic components can speed up dynamic programming algorithms.

References

1. Isolde Adler, Frederic Dorn, Fedor V. Fomin, Ignasi Sau, and Dimitrios M. Thilikos. Fast minor testing in planar graphs. *Algorithmica*, 64(1):69–84, 2012.
2. Noga Alon, Paul Seymour, and Robin Thomas. A separator theorem for nonplanar graphs. *Journal of the American Mathematical Society*, 3(4):801–808, 1990.
3. Noga Alon, Raphy Yuster, and Uri Zwick. Color-coding: A New Method for Finding Simple Paths, Cycles and Other Small Subgraphs Within Large Graphs. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC'94, pages 326–335, New York, NY, USA, 1994. ACM. doi:10.1145/195058.195179.
4. Omid Amini, Fedor V. Fomin, and Saket Saurabh. Counting subgraphs via homomorphisms. *SIAM Journal on Discrete Mathematics*, 26(2):695–717, 2012.
5. Hans L. Bodlaender. Treewidth: Algorithmic techniques and results. In Igor Prívara and Peter Ružička, editors, *Mathematical Foundations of Computer Science 1997*, volume 1295 of *Lecture Notes in Computer Science*, pages 19–36. Springer Berlin Heidelberg, 1997. doi:10.1007/BFb0029946.
6. Hans L. Bodlaender, Pal Gronas Drange, Markus S. Dregi, Fedor V. Fomin, Daniel Lokshtanov, and Michal Pilipczuk. An $O(c^k n)$ 5-approximation algorithm for treewidth. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 499–508. IEEE, 2013.
7. Hans L. Bodlaender and Jesper Nederlof. Subexponential Time Algorithms for Finding Small Tree and Path Decompositions. In *Algorithms–ESA 2015*, pages 179–190. Springer, 2015.
8. Hans L. Bodlaender and Jesper Nederlof. Subexponential time algorithms for finding small tree and path decompositions. *arXiv preprint arXiv:1601.02415*, 2016.
9. Hans L. Bodlaender and Johan M.M. Van Rooij. Exact algorithms for intervalizing colored graphs. In *Theory and Practice of Algorithms in (Computer) Systems*, pages 45–56. Springer, 2011.
10. Paul Bonsma. Surface split decompositions and subgraph isomorphism in graphs on surfaces. *arXiv preprint arXiv:1109.4554*, 2011.
11. Marek Cygan, Jakub Pachocki, and Arkadiusz Socała. The hardness of Subgraph Isomorphism. *arXiv preprint arXiv:1504.02876*, 2015.
12. Erik D. Demaine and MohammadTaghi Hajiaghayi. Bidimensionality: new connections between FPT algorithms and PTASs. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 590–601. Society for Industrial and Applied Mathematics, 2005.
13. Frederic Dorn. Planar subgraph isomorphism revisited. *arXiv preprint arXiv:0909.4692*, 2009.
14. Rodney G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.
15. David Eppstein. Subgraph Isomorphism in Planar Graphs and Related Problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'95, pages 632–640, Philadelphia, PA, USA, 1995. Society for Industrial and Applied Mathematics.

16 Fedor V. Fomin, Alexander Golovnev, Alexander S. Kulikov, and Ivan Mihajlin. Tight Bounds for Subgraph Isomorphism and Graph Homomorphism. *arXiv preprint arXiv:1507.03738*, 2015.

17 Fedor V. Fomin, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. Subexponential parameterized algorithms for planar and apex-minor-free graphs via low treewidth pattern covering. *arXiv preprint arXiv:1604.05999*, 2016.

18 Jakub Gajarskỳ, Petr Hliněnỳ, Jan Obdržálek, Sebastian Ordyniak, Felix Reidl, Peter Rossmanith, Fernando Sanchez Villaamil, and Somnath Sikdar. Kernelization using structural parameters on sparse graph classes. In *Algorithms–ESA 2013*, pages 529–540. Springer, 2013.

19 Arvind Gupta and Naomi Nishimura. The complexity of subgraph isomorphism for classes of partial k-trees. *Theoretical Computer Science*, 164(1):287–298, 1996.

20 MohammadTaghi Hajiaghayi and Naomi Nishimura. Subgraph isomorphism, log-bounded fragmentation, and graphs of (locally) bounded treewidth. *Journal of Computer and System Sciences*, 73(5):755–768, 2007.

21 Ken-ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed. The disjoint paths problem in quadratic time. *Journal of Combinatorial Theory, Series B*, 102(2):424–435, 2012. doi:10.1016/j.jctb.2011.07.004.

22 Andrzej Lingas. Subgraph isomorphism for biconnected outerplanar graphs in cubic time. *Theoretical Computer Science*, 63(3):295–302, 1989. doi:10.1016/0304-3975(89)90011-X.

23 Daniel Lokshtanov, Dániel Marx, Saket Saurabh, et al. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, (105):41–72, 2011.

24 Dániel Marx. What is next? Future directions in parameterized complexity. In *The Multivariate Algorithmic Revolution and Beyond*, pages 469–496. Springer, 2012.

25 Dániel Marx and Michał Pilipczuk. Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism (but were afraid to ask). *arXiv preprint arXiv:1307.2187*, 2013.

26 Jiří Matoušek and Robin Thomas. On the complexity of finding iso-and other morphisms for partial k-trees. *Discrete Mathematics*, 108(1):343–364, 1992.

27 Neil Robertson and Paul D. Seymour. Graph minors. XIII. The Disjoint Paths Problem. *Journal of Combinatorial Theory, Series B*, 63(1):65–110, 1995. doi:10.1006/jctb.1995.1006.

28 Maciej M. Sysło. The subgraph isomorphism problem for outerplanar graphs. *Theoretical Computer Science*, 17(1):91–97, 1982. doi:10.1016/0304-3975(82)90133-5.

29 Dimitrios M. Thilikos. The Multivariate Algorithmic Revolution and Beyond. chapter Graph Minors and Parameterized Algorithm Design, pages 228–256. Springer-Verlag, Berlin, Heidelberg, 2012.

30 Julian R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.

Relating Graph Thickness to Planar Layers and Bend Complexity*

Stephane Durocher[†1] **and Debajyoti Mondal**[2]

1 **Department of Computer Science, University of Manitoba, Winnipeg, Canada**
durocher@cs.umanitoba.ca

2 **Department of Computer Science, University of Manitoba, Winnipeg, Canada**
jyoti@cs.umanitoba.ca

Abstract

The thickness of a graph $G = (V, E)$ with n vertices is the minimum number of planar subgraphs of G whose union is G. A polyline drawing of G in $\mathbb{R}^2$ is a drawing Γ of G, where each vertex is mapped to a point and each edge is mapped to a polygonal chain. Bend and layer complexities are two important aesthetics of such a drawing. The bend complexity of Γ is the maximum number of bends per edge in Γ, and the layer complexity of Γ is the minimum integer r such that the set of polygonal chains in Γ can be partitioned into r disjoint sets, where each set corresponds to a planar polyline drawing. Let G be a graph of thickness t. By Fáry's theorem, if $t = 1$, then G can be drawn on a single layer with bend complexity 0. A few extensions to higher thickness are known, e.g., if $t = 2$ (resp., $t > 2$), then G can be drawn on t layers with bend complexity 2 (resp., $3n + O(1)$).

In this paper we present an elegant extension of Fáry's theorem to draw graphs of thickness $t > 2$. We first prove that thickness-t graphs can be drawn on t layers with $2.25n + O(1)$ bends per edge. We then develop another technique to draw thickness-t graphs on t layers with reduced bend complexity for small values of t, e.g., for $t \in \{3, 4\}$, the bend complexity decreases to $O(\sqrt{n})$. Previously, the bend complexity was not known to be sublinear for $t > 2$. Finally, we show that graphs with linear arboricity k can be drawn on k layers with bend complexity $\frac{3(k-1)n}{(4k-2)}$.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling, G.2.2 Graph Theory

Keywords and phrases Graph Drawing, Thickness, Geometric Thickness, Layers, Bends

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.10

1 Introduction

A polyline drawing of a graph $G = (V, E)$ in $\mathbb{R}^2$ maps each vertex of G to a distinct point, and each edge of G to a polygonal chain. Many problems in VLSI layout and software visualization are tackled using algorithms that produce polyline drawings. For a variety of practical purposes, these algorithms often seek to produce drawings that optimize several drawing aesthetics, e.g., minimizing the number of bends, minimizing the number of crossings, etc. In this paper we examine two such parameters: *bend complexity* and *layer complexity*.

The *thickness* of a graph G is the minimum number $\theta(G)$ such that G can be decomposed into $\theta(G)$ planar subgraphs. Let Γ be a polyline drawing of G. Then the *bend complexity*

* A full version of the paper is available online at [11].

† Work of the author is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 10; pp. 10:1–10:13

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

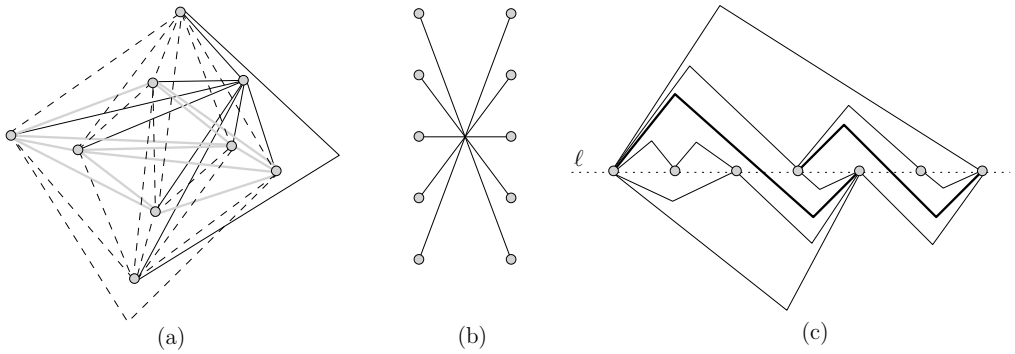


Figure 1 (a) A polyline drawing of K_9. (b) A drawing of a matching of size 5. (c) A monotone topological book embedding of some graph. The edges that crosses the spine ℓ are shown in bold.

of Γ is the minimum integer b such that each edge in Γ has at most b bends. A set of edges $E' \subseteq E$ is called a *crossing-free edge set in* Γ, if the corresponding polygonal chains correspond to a *planar polyline drawing*, i.e., no two polylines that correspond to a pair of edges in E' intersect, except possibly at their common endpoints. The *layer complexity* of Γ is the minimum integer t such that the edges of Γ can be partitioned into t crossing-free edge sets. Figure 1(a) illustrates a polyline drawing of K_9 on 3 layers with bend complexity 1. At first glance the layer complexity of Γ may appear to be related to the thickness of G. However, the layer complexity is a property of the drawing Γ, while thickness is a graph property. The layer complexity of Γ can be arbitrarily large even when G is planar, e.g., consider the case when G is a matching and Γ is a straight-line drawing, where each edge crosses all the other edges; see Figure 1(b).

The layer complexity of a thickness-t graph G is at least t, and every n-vertex thickness-t graph admits a drawing on t layers with bend complexity $O(n)$ [20]. The problem of drawing thickness-t graphs on t planar layers is closely related to the *simultaneous embedding* problem, where given a set of planar graphs $G_1, \ldots, G_t$ on a common set of vertices, the task is to compute their planar drawings $D_1, \ldots, D_t$ such that each vertex is mapped to the same point in the plane in each of these drawings. Figure 1(a) can be considered to be a simultaneous embedding of three given planar graphs.

1.1 Related Work

Graphs with low thickness admit polyline drawings on few layers with low bend complexity. If $\theta(G) = 1$, then by Fáry's theorem [16], G admits a drawing on a single layer with bend complexity 0. Every pair of planar graphs can be simultaneously embedded using two bends per edge [15, 17]. Therefore, if $\theta(G) = 2$, then G admits a drawing on two layers with bend complexity 2. The best known lower bound on the bend complexity of such drawings is one [10]. Duncan et al. [9] showed that graphs with maximum degree four can be drawn on two layers with bend complexity 0. Wood [21] showed how to construct drawings on $O(\sqrt{m})$ layers with bend complexity 1, where m is the number of edges in G.

Given an n-vertex planar graph G and a point location for each vertex in $\mathbb{R}^2$, Pach and Wenger [20] showed that G admits a planar polyline drawing with the given vertex locations, where each edge has at most $120n$ bends. They also showed that $\Omega(n)$ bends are sometimes necessary. Badent et al. [1] and Gordon [18] independently improved the bend complexity to $3n + O(1)$. Consequently, for $\theta(G) \geq 3$, these constructions can be used to draw G on $\theta(G)$ layers with at most $3n + O(1)$ bends per edge.

A rich body of literature [3, 4, 12, 13] examines *geometric thickness*, i.e., the maximum number of planar layers necessary to achieve 0 bend complexity. Dujmović and Wood [7] proved that $\lceil k/2 \rceil$ layers suffice for graphs of treewidth k. Duncan [8] proved that $O(\log n)$ layers suffice for graphs with arboricity two or outerthickness two, and $O(\sqrt{n})$ layers suffice for thickness-2 graphs. Dillencourt et al. [6] proved that complete graphs with n vertices require at least $\lceil (n/5.646) + 0.342 \rceil$ and at most $\lceil n/4 \rceil$ layers.

1.2 Our Results

The goal of this paper is to extend our understanding of the interplay between the layer complexity and bend complexity in polyline drawings.

We first show that every n-vertex thickness-t graph admits a polyline drawing on t layers with bend complexity $2.25n + O(1)$, improving the $3n + O(1)$ upper bound derived from [1, 18]. We then give another drawing algorithm to draw thickness-t graphs on t layers, which improves the bend complexity for smaller values of t, e.g., for graphs with $t \in \{3, 4\}$, it reduces the bend complexity to $O(\sqrt{n})$. No such sublinear upper bound on the bend complexity was previously known for $t > 2$. Finally, we show that every n-vertex graph with linear arboricity $k \geq 2$ admits a polyline drawing on k layers with bend complexity $\frac{3(k-1)n}{(4k-2)}$, where the *linear arboricity* of a graph G is the minimum number of linear forests (i.e., each connected component is a path) whose union is G.

The rest of the paper is organized as follows. We start with some preliminary definitions and results (Section 2). In the subsequent section (Section 3) we present two constructions to draw thickness-t graphs on t layers. Section 4 presents the results on drawing graphs of bounded arboricity. Finally, Section 5 concludes the paper pointing out the limitations of our results and suggesting directions for future research.

2 Technical Details

In this section we describe some preliminary definitions, and review some known results.

Let $G = (V, E)$ be a planar graph. A *monotone topological book embedding* of G is a planar drawing Γ of G that satisfies the following properties.

P_1: The vertices of G lie along a horizontal line ℓ in Γ. We refer to ℓ as the *spine* of Γ.

P_2: Each edge $(u, v) \in E$ is an x-monotone polyline in Γ, where (u, v) either lies on one side of ℓ, or crosses ℓ at most once.

P_3: Let (u, v) be an edge that crosses ℓ at point d, where u appears before v on ℓ. Let $u, \ldots, d, \ldots, v$ be the corresponding polyline. Then the polyline $u, \ldots, d$ lies above ℓ, and the polyline $d, \ldots, v$ lies below ℓ.

Figure 1(c) illustrates a monotone topological book embedding of a planar graph. Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two graphs on a common set of vertices. A *simultaneous embedding* Γ of G_1 and G_2 consists of their planar drawings D_1 and D_2, where each vertex is mapped to the same point in the plane in both D_1 and D_2. Erten and Kobourov [15] showed that every pair of planar graphs admit a simultaneous embedding with at most three bends per edge. Giacomo and Liotta [17] observed that by using monotone topological book embeddings Erten and Kobourov's [15] construction can achieve a drawing with two bends per edge. Here we briefly recall this drawing algorithm. Without loss of generality assume that both G_1 and G_2 are triangulations. Let π_i, where $1 \leq i \leq 2$, be a vertex ordering that corresponds to a monotone topological book embedding of G_i. Let P_i be the corresponding *spinal path*, i.e., a path that corresponds to π_i. Note that some of the edges of P_i may not

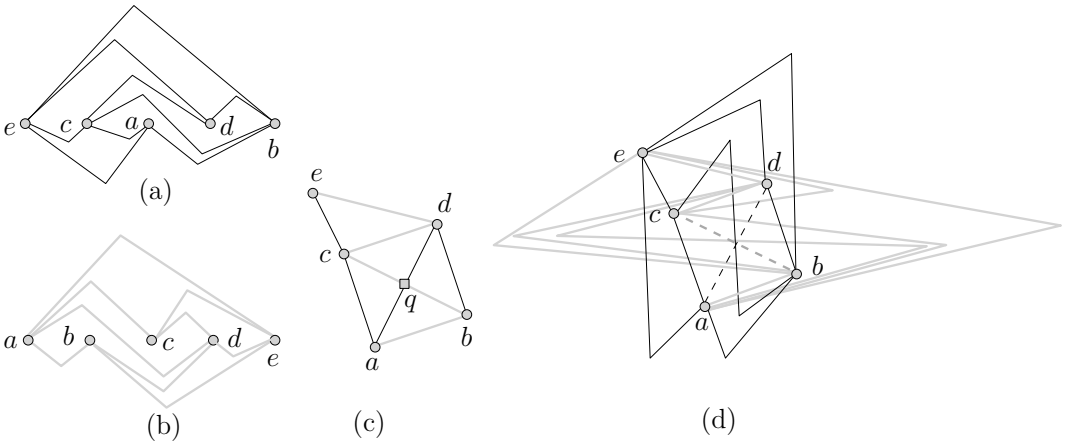


Figure 2 (a)–(b) Monotone topological book embeddings of G_1 and G_2. (c)–(d) Simultaneous embedding of G_1 and G_2, where the deleted edges are shown in dashed lines.

exist in G_i, e.g., edges (a, d) and (b, c) in Figures 2(a) and (b), respectively, and these edges of P_i create edge crossings in G_i. Add a dummy vertex at each such edge crossing. Let $\delta_i(v)$ be the position of vertex v in π_i. Then P_1 and P_2 can be drawn simultaneously on an $O(n) \times O(n)$ grid [5] by placing each vertex at the grid point $(\delta_1(v), \delta_2(v))$; see Figure 2(c). The mapping between the dummy vertices of P_1 and P_2 can be arbitrary, here we map the dummy vertex on (a, d) to the dummy vertex on (b, c). Finally, the edges of G_i that do not belong to P_i are drawn. Let e be such an edge in G_i. If e does not cross the spine, then it is drawn using one bend on one side of P_i according to the book embedding of G_i. Otherwise, let q be a dummy vertex on the edge $e = (u, v)$, which corresponds to the intersection point of e and the spine. The edges (u, q) and (v, q) are drawn on opposite sides of P_i such that the polyline from u to v do not create any bend at q. Since each of (u, q) and (v, q) contains only one bend, e contains only two bends. Finally, the edges of P_i that do not belong to G_i are removed from the drawing; see Figure 2(d).

Let Γ be a planar polyline drawing of a path $P = \{v_1, v_2, \ldots, v_n\}$. We call Γ an *uphill* drawing if for any point q on Γ, the upward ray from q does not intersect the path $v_1, \ldots, q$. Note that q may be a vertex location or an interior point of some edge in Γ. Let a and b be two points in $\mathbb{R}^2$. Then a and b are *r-visible* to each other if and only if their exists a polygonal chain of length r with end points a, b that does not intersect Γ at any point except possibly at a, b. A point p *lies between two other points* v, w, if either the inequality $x(v) < x(p) < x(w)$ or $x(w) < x(p) < x(v)$ holds.

A set of points is *monotone* if the polyline connecting them from left to right is monotone with respect to y-axis. Let S be a set of n points in general position. By the Erdös-Szekeres theorem [14], S can be partitioned into $O(\sqrt{n})$ disjoint monotone subsets, and such a partition can be computed in $O(n^{1.5})$ time [2].

3 Drawing Thickness-t Graphs on t Layers

In this section we give two separate construction techniques to draw thickness-t graphs on t layers. We first present a construction achieving $2.25n + O(1)$ upper bound (Section 3.1), which is simple and intuitive. Although the technique is simple, the idea of the construction will be used frequently in the rest of the paper. Therefore, we explained the construction in reasonable details.

Later, we present a second construction (Section 3.2), which is more involved, and relies on a deep understanding of the geometry of point sets. In this case, the upper bound on the bend complexity will depend on some generalization of Erdös-Szekeres theorem [14], e.g., partitioning a point set into monotone subsequences in higher dimensions (Section 3.2.3).

3.1 A Simple Construction with Bend Complexity $2.25n + O(1)$

Let $G_1, \ldots, G_t$ be the planar subgraphs of the input graph G, and let S be an ordered set of n points on a semicircular arc. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of vertices of G. We show that each G_i, where $1 \leq i \leq t$, admits a polyline drawing with bend complexity $2.25n + O(1)$ such that vertex v_j is mapped to the jth point of S. To draw G_i, we will use the vertex ordering of its monotone topological book embedding. The following lemma will be useful to draw the spinal path P_i of G_i.

▶ **Lemma 1.** *Let $S = \{p_0, p_1, \ldots, p_{n+1}\}$ be a set of points lying on an x-monotone semicircular arc (e.g., see Figure 3(a)), and let $P = \{v_1, v_2, \ldots, v_n\}$ be a path of n vertices. Assume that p_0 and p_{n+1} are the leftmost and rightmost points of S, respectively, and the points $p_1, \ldots, p_n$ are equally spaced between them in some arbitrary order. Then P admits an uphill drawing Γ with the vertex v_i assigned to p_i, where $1 \leq i \leq n$, and every point p_i satisfies the following properties:*

(A) *Both the points p_0 and p_{n+1} are $(3n/4)$-visible to p_i.*

(B) *One can draw an x-monotone polygonal chain from p_0 to p_{n+1} with $3n/4$ bends that intersects Γ only at p_i.*

Proof. We prove the lemma by constructing such a drawing Γ for P. The construction assigns a polyline for each edge of P. The resulting drawing may contain edge overlaps, and the bend complexity could be as large as $n - 2$. Later we remove these degeneracies and reduce the bend complexity to obtain Γ.

Drawings of Edges: For each point $p_i \in S$, where $1 \leq i \leq n$, we create an *anchor point* p'_i at $(x(p_i), y(p_i) + \epsilon)$, where $\epsilon > 0$. We choose ϵ small enough such that for any j, where $1 \leq i \neq j \leq n$, all the points of S between p_i and p_j lie above (p'_i, p'_j). Figure 3(a) illustrates this property for the anchor point p'_1.

We first draw the edge (v_1, v_2) using a straight line segment. For each j from 2 to $n - 1$, we now draw the edges (v_j, v_{j+1}) one after another. Assume without loss of generality that $x(p_j) < x(p_{j+1})$. We call a point $p \in S$ between p_j and p_{j+1} a *visited point* if the corresponding vertex v appears in $v_1, \ldots, v_j$, i.e., v has already been placed at p. We draw an x-monotone polygonal chain L that starts at v_j, connects the anchors of the intermediate visited points from left to right, and ends at v_{j+1}. Figure 3(b) illustrates such a construction.

Since the number of bends on L is equal to the number of visited points of S between p_j and p_{j+1}, each edge contains at most α bends, where α is the number of points of S between p_j and p_{j+1}.

Removing Degeneracies: The drawing D_n of the path P constructed above contains edge overlaps, e.g., see the edges (v_3, v_4) and (v_4, v_5) in Figure 3(c). To remove the degeneracies, for each i, we spread the corresponding bend points between p_i and p'_i, in the order they appear on the path, see Figure 3(d). Consequently, we obtain a planar drawing of P. Let the resulting drawing be D'_n. Since each edge (p_j, p_{j+1}) is drawn as an x-monotone polyline above the path $p_1, \ldots, p_j$, D'_n satisfies the uphill property. Note that D'_n may have bend

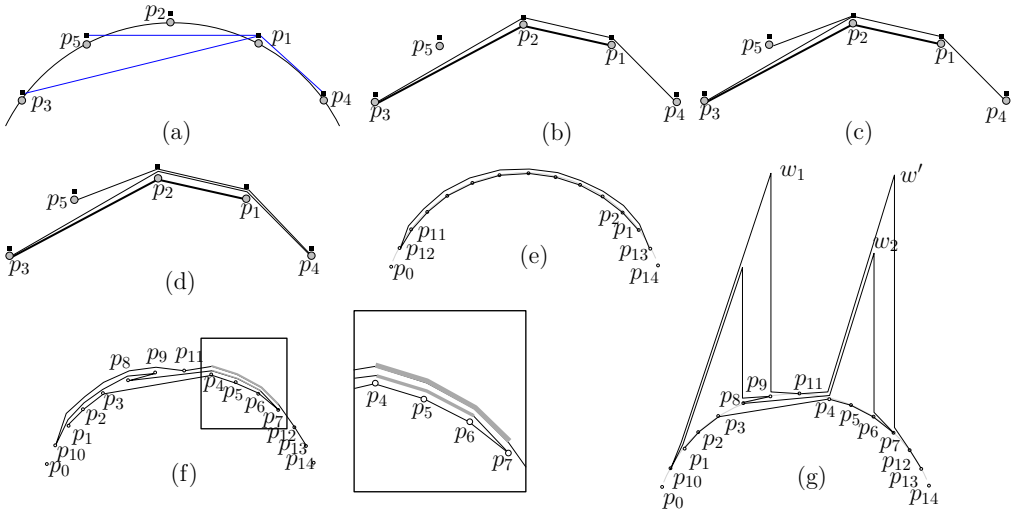


Figure 3 Illustration for the proof of Lemma 1. Anchor points are shown in black squares. For a larger view of this figure, see Appendix A.

complexity $n-2$, e.g., see Figure 3(e). We now show how to reduce the bend complexity and satisfy Properties A–B.

Reducing Bend Complexity: A pair of points in S are *consecutive* if they do not contain any other point of S in between. Let e be any edge of P. Let C_e be the corresponding polygonal chain in D'_n. A pair of bend points on C_e are called *consecutive bends* if their corresponding points in S are also consecutive. A *bend-interval* of C_e is a maximal sequence of consecutive bends in C_e. Note that we can partition the bends on e into disjoint sets of bend-intervals.

For any bend-interval s, let $l(s)$ and $r(s)$ be the x-coordinates of the left and right endpoints of s, respectively. Let s_1 and s_2 be two bend-intervals lying on two distinct edges e_1 and e_2 in D'_n, respectively, where e_2 appears after e_1 in P. We claim that the intervals $[l(s_1), r(s_1)]$ and $[l(s_2), r(s_2)]$ are either disjoint, or $[l(s_1), r(s_1)] \subseteq [l(s_2), r(s_2)]$. We refer to this property as the *balanced parenthesis property of the bend-intervals*. To verify this property assume that for some s_1, s_2, we have $[l(s_1), r(s_1)] \cap [l(s_2), r(s_2)] \neq \phi$. Since e_2 appears after e_1, and since s_2 is a maximal sequence of consecutive bends, the inequalities $l(s_2) \leq l(s_1)$ and $r(s_2) \geq r(s_1)$ hold, i.e., $[l(s_1), r(s_1)] \subseteq [l(s_2), r(s_2)]$. We say that s_1 *is nested by* s_2. Figure 3(f) illustrates such a scenario, where s_1, s_2 are shown in thin and thick gray lines, respectively.

We now consider the edges of P in reverse order, i.e., for each j from n to 2, we modify the drawing of $e = (v_j, v_{j-1})$. For each bend-interval $s = (b_1, b_2 \ldots, b_r)$ of C_e, if s has three or more bends, then we delete the bends $b_2, \ldots, b_{r-1}$, and join b_1 and b_r using a new bend point w. To create w, we consider the two cases of the balanced parenthesis property.

If s is not nested by any other bend-interval in D'_n, then we place w high enough above b_r such that the chain b_1, w, b_r does not introduce any edge crossing, e.g., see the point $w_1(= w)$ in Figure 3(g). On the other hand, if s is nested by some other bend-interval, then let s' be such a bend-interval immediately above s. Since $s' = (b'_1, b'_2, \ldots, b'_r)$ is already processed, it must have been replaced by some chain b'_1, w', b'_r. Therefore, we can find a location for b inside $\angle b'_1 w' b'_r$ such that the chain b_1, w, b_r does not introduce any edge crossing, e.g., see the points w' and $w_2(= w)$ in Figure 3(g). Let the resulting drawing of P be Γ.

We now show that the above modification reduces the bend complexity to $3n/4$. Let e be an edge of P that contains α points from S between its endpoints. Let C_e be the corresponding polygonal chain in D'_n. Recall that any bend-interval of length ℓ in C_e contributes to $\min\{\ell, 3\}$ bends on e in Γ. Therefore, if there are at most $\alpha/4$ bend-intervals on C_e, then e can have at most $3\alpha/4$ bends in Γ. Otherwise, if there are more than $\alpha/4$ bend-intervals, then there are at least $\alpha/4$ points[1] of S that do not contribute to bends on C_e. Therefore, in both cases, C_e can have at most $3\alpha/4$ bends in Γ.

Satisfying Properties A–B: Let p_i be any point of $S \setminus \{p_0, p_{n+1}\}$. We first show that p_0 is $(3n/4)$-visible to p_i. Let D_i, where $1 \leq i \leq n$, be the drawing of the path $v_1, v_2, \ldots, v_i$. Observe that one can insert an edge (p_0, p_i) using an x-monotone polyline L such that the bends on L correspond to the intermediate visited points. Now the drawing of the rest of the path $v_i, v_{i+1}, \ldots, v_n$ can be continued such that it does not cross L. Therefore, if the number of points of S between p_0 and p_i is α, then L has at most α bends. Finally, the process of reducing bend complexity improves the number of bends on L to $3\alpha/4$.

Similarly, we can observe that p_{n+1} is at most $3\alpha'/4$ visible to p_i, where α' is the number of points of S between p_i and p_{n+1}. Since the edges (p_0, p_i) and (p_i, p_{n+1}) are x-monotone, we can draw an x-monotone polygonal chain from p_0 to p_{n+1} with at most $3(\alpha + \alpha')/4 \leq (3n/4)$ bends that intersects Γ only at p_i. ◀

▶ **Theorem 2.** *Every n-vertex graph of t admits a drawing on t layers with bend complexity $2.25n + O(1)$.*

Proof. Let $G_1, \ldots, G_t$ be the planar subgraphs of the input graph G, and let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of vertices of G. Let $S = \{p_0, p_1, \ldots, p_{n+1}\}$ be a set of $n + 2$ points lying on a semicircular arc as defined in Lemma 1. Let P_i be spinal path of the monotone topological book embedding of G_i, where $1 \leq i \leq t$. We first compute an uphill drawing Γ_i of the path P_i. We then draw the edges of G_i that do not belong to P_i. Let $e = (u, v)$ be such an edge, and without loss of generality assume that u appears to the left of v on the spine.

If e lies above (resp., below) the spine, then we draw two x-monotone polygonal chains; one from u to p_0 (resp., p_{n+1}), and the other from v to p_0 (resp., p_{n+1}). By Lemma 1, these polygonal chains do not intersect Γ_i except at u and v, and each contains at most $3n/4$ bends. Hence e contains at most $1.5n$ bends in total.

If e crosses the spine, then it crosses some edge (w, w') of P_i. Draw the edges (u, w) and (w, v) using the polylines $u, \ldots, p_0, \ldots, w$ and $w, \ldots, p_{n+1}, \ldots, v$, respectively. The polylines $u, \ldots, p_0$ and $p_{n+1}, \ldots, v$ are x-monotone, and have at most $3n/4$ bends each. The polyline $C = (p_0, \ldots, w \ldots, p_{n+1})$ is also x-monotone and has at most $3n/4$ bends. Hence the number of bends is $2.25n$ in total. It is straightforward to avoid the degeneracy at w, by adding a constant number of bends on C.

Note that we still have some edge overlaps at p_0 and p_{n+1}. It is straightforward to remove these degeneracies by adding only a constant number of more bends per edge. ◀

3.2 A Construction for Small Values of t

In this section we give another construction to draw thickness-t graphs on t layers. We first show that every thickness-t graph, where $t \in \{3, 4\}$, can be drawn on t layers with bend complexity $O(\sqrt{n})$, and then show how to extend the technique for larger values of t.

[1] Every pair of consecutive bend-intervals contain such a point in between.

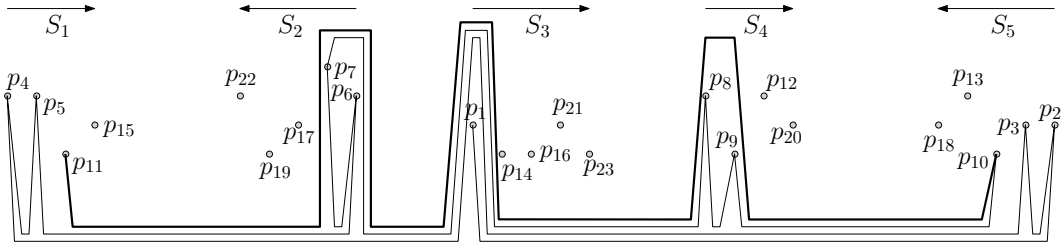


Figure 4 Illustration for the proof of Lemma 3. The edge (p_{10}, p_{11}) is shown in bold. Passing through each intermediate set requires at most 4 bends.

3.2.1 Construction when $t = 3$

Let S be an ordered set of n points, where the ordering is by increasing x-coordinate. A (k, n)-*group* $S_{k,n}$ is a partition of S into k disjoint ordered subsets $\{S_1, \ldots, S_k\}$, each containing contiguous points from S. Label the points of S using a permutation of $p_1, p_2, \ldots, p_n$ such that for each set $S' \in S_{k,n}$, the indices of the points in S' are either increasing or decreasing. If the indices are increasing (resp., decreasing), then we refer S' as a rightward (resp., leftward) set. We will refer to such a labelling as a *smart labelling* of $S_{k,n}$. Figure 4 illustrates a $(5, 23)$-group and a smart labelling of the underlying point set $S_{5,23}$.

Note that for any i, where $1 \leq i \leq n$, deletion of the points $p_1, \ldots, p_i$ removes the points of the rightward (resp., leftward) sets from their left (resp., right). The *necklace* of $S_{k,n}$ is a path obtained from a smart labelling of $S_{k,n}$ by connecting the points p_i, p_{i+1}, where $1 \leq i \leq n-1$. The following lemma constructs an uphill drawing of the necklace using $O(k)$ bends per edge.

▶ **Lemma 3.** *Let S be a set of n points ordered by increasing x-coordinate, and let $S_{k,n} = \{S_1, \ldots, S_k\}$ be a (k, n)-group of S. Label $S_{k,n}$ with a smart labelling. Then the necklace of $S_{k,n}$ admits an uphill drawing with $O(k)$ bends per edge.*

Proof. We construct this uphill drawing incrementally in a similar way as in the proof of Lemma 1. Let D_j, where $1 \leq j \leq n$, be the drawing of the path $p_1, \ldots, p_j$. At each step of the construction, we maintain the invariant that D_j is an uphill drawing.

We first assign v_1 to p_1. Then for each i from 1 to $n-1$, we draw the edge (p_i, p_{i+1}) using an x-monotone polyline L that lies above D_i and below the points $p_{j'}$, where $j' > i+1$. Figure 4 illustrates such a drawing of (p_i, p_{i+1}).

The crux of the construction is that one can draw such a polyline L using at most $O(k)$ bends. Assume that p_i and p_{i+1} belong to the sets $S_l \in S_{k,n}$ and $S_r \in S_{k,n}$, respectively. If S_l and S_r are identical, then p_i and p_{i+1} are consecutive, and hence it suffices to use at most $O(1)$ bends to draw L. On the other hand, if S_l and S_r are distinct, then there can be at most $k-2$ sets of $S_{k,n}$ between them. Let S_m be such a set. While passing through S_m, we need to keep the points that already belong to the path, below L, and the rest of the points above L. By the property of smart labelling, the points that belong to D_i are consecutive in S_m, and lie to the left or right side of S_m depending on whether S_m is rightward or leftward. Therefore, we need only $O(1)$ bends to pass through S_m. Since there are at most $k-2$ sets between S_l and S_r, $O(k)$ bends suffice to construct L. ◀

We are now ready to describe the main construction. Let G be an n-vertex thickness-3 graph, and let G_1, G_2, G_3 be the planar subgraphs of G. Let P_i be the spinal path of the monotone topological book embedding of G_i, where $1 \leq i \leq 3$. We first create a set of n points and assign them to the vertices of G. Later we route the edges of G.

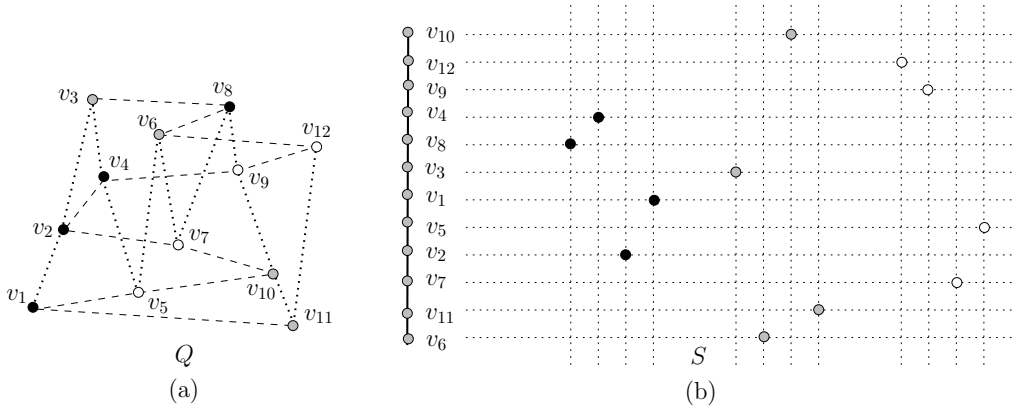


Figure 5 Creating vertex locations for drawing thickness-3 graphs, where P_1, P_2 and P_3 are shown in dotted, dashed and thick solid lines, respectively.

Creating Vertex Locations: Assume without loss of generality that $P_1 = (v_1, \ldots, v_n)$. For each i from 1 to n, we place a point at (i, j) in the plane, where j is the position of v_j in P_2. Let the resulting point set be Q. Recall that Q can be partitioned into disjoint monotone subsets $Q_1, \ldots, Q_k$, where $k \in O(\sqrt{n})$ [2]. Figure 5(a) illustrates such a partition in black, gray and white.

The sets $Q_1, \ldots, Q_k$ are ordered by the x-coordinate, and the indices of the labels of the points at each set is in increasing order. Therefore, if we place the points of the ith set between the lines $x = 2(i-1)n$ and $x = (2i-1)n$, then the resulting point set Q' would be a (k, n)-group, labelled by a smart labelling. Finally, we adjust the y-coordinates of the points according to the position of the corresponding vertices in P_3. Let the resulting point set be S. Figure 5(b) illustrates the vertex locations, where $P_1 = (v_1, v_2, \ldots, v_n)$, $P_2 = (v_{11}, v_1, \ldots, v_3)$, and $P_3 = (v_6, v_{11}, \ldots, v_{10})$.

Edge Routing: It is straightforward to observe that the path P_1 is a necklace for the current labelling of the points of $S_{k,n}$. Therefore, by Lemma 3, we can construct an uphill drawing of P_1 on S. Observe that for every set $S' \in S_{k,n}$, the corresponding points are monotone in Q, i.e., the points of S' are ordered along the x-axis either in increasing or decreasing order of their y-coordinates in Q. Therefore, relabelling the points according to the increasing order of their y-coordinates in Q will produce another smart labelling of S, and the corresponding necklace would be the path P_2. Therefore, we can use Lemma 3 to construct an uphill drawing of P_2 on S. Since the height of the points of S are adjusted according to the vertex ordering on P_3, connecting the points of S from top to bottom with straight line segments yields a y-monotone drawing of P_3.

We now route the edges of G_i that do not belong to P_i, where $1 \leq i \leq 3$. Since P_3 is drawn as a y-monotone polygonal path, we can use the technique of Erten and Kobourov [15] to draw the remaining edges of G_3. To draw the edges of G_2, we insert two points p_0 and p_{n+1} to the left and right of all the points of S, respectively. Then the drawing of the remaining edges of G_1 and G_2 is similar to the edge routing described in the proof of Theorem 2. That is, if the edge $e = (u, v)$ lies above (resp., below) the spine, then we draw it using two x-monotone polygonal chains from p_0 (resp., p_{n+1}). Otherwise, if e crosses the spine, then we draw three x-monotone polygonal chains, one from u to p_0, another from p_0 to p_{n+1}, and the third one from v to p_{n+1}. Since $k \in O(\sqrt{n})$, the number of bends on e is $O(\sqrt{n})$. Finally, we remove the degeneracies, which increases the bends per edge by a small constant.

3.2.2 Construction when $t = 4$

We now show that the technique for drawing thickness-3 graphs can be generalized to draw thickness-4 graphs with the same bend complexity.

Let $G_1, \ldots, G_4$ be the planar subgraphs of G, and let $P_1, \ldots, P_4$ be the corresponding spinal paths. While constructing the vertex locations, we use a new y-coordinate assignment for the points of S. Instead of placing the points according to the vertex ordering on the path P_3, we create a particular order, by transposing the x- and y-axis, that would help to construct uphill drawings of P_3 and P_4 with bend complexity $O(\sqrt{n})$. That is, we first create a (k', n)-group $S'_{k',n}$ using P_3 and P_4, where $k' \in O(\sqrt{n})$, in a similar way that we created $S_{k,n}$ using P_1 and P_2. We then adjust the y-coordinates of the points of S according to the order these points appear in $S'_{k',n}$. Appendix B includes an example of such a construction.

The construction of G_1 and G_2 remains the same as described in the previous section. However, since P_3 and P_4 now admit uphill drawings on S with respect to y-axis, the drawings of G_3 and G_4 are now analogous to the construction of G_1 and G_2.

3.2.3 Construction when $t > 4$

De Bruijn [19] observed that the result of Erdös-Szekeres [14] can be generalized to higher dimensions. Given a sequence ρ of n tuples, each of size κ, one can find a subsequence of at least $n^{1/\lambda}$ tuples, where $\lambda = 2^\kappa$, such that they are monotone (i.e., increasing or decreasing) in every dimension. If we repeatedly extract such monotone sequences, then we obtain a partition of ρ into a set of monotone subsequences. We use this idea to extend our drawing algorithm to higher thickness.

Let $G_1, \ldots, G_t$ be the planar subgraphs of G, and let $P_1, \ldots, P_t$ be the corresponding spinal paths. Let $v_1, v_2, \ldots, v_n$ be the vertices of G. Construct a corresponding sequence $\rho = (\tau_1, \tau_2, \ldots, \tau_n)$ of n tuples, where each tuple is of size t, and the ith element of a tuple τ_j corresponds to the position of the corresponding vertex v_j in P_i, where $1 \leq i \leq t$ and $1 \leq j \leq n$. We now partition ρ into a set of monotone subsequences. Let $f(n, t)$ be the number of monotone subsequences in this partition.

For each of these monotone sequences, we create an ordered set of consecutive points along the x-axis, where the vertex v_j corresponds to the point p_j. It is now straightforward to observe that these sets correspond to a (k, n)-group $S_{k,n}$, where $k \leq f(n, t)$. Furthermore, since each group corresponds to a monotone sequence of tuples, for each P_i, the positions of the corresponding vertices are either increasing or decreasing. Hence, every path P_i corresponds to a necklace for some smart labelling of $S_{k,n}$. Therefore, by Lemma 3, we can construct an uphill drawing of P_i on S. We now add the remaining edges of G_i following the construction described in Section 3.2.1. Since $k \leq f(n, t)$, the number of bends is bounded by $O(f(n, t))$.

Observe that all the points in the above construction have the same y-coordinate. Therefore, we can improve the construction by distributing the load equally among the x-axis and y-axis as we did in Section 3.2.2. Specifically, we draw the graphs $G_1, \ldots, G_{\lceil t/2 \rceil}$ using the uphill drawings of their spinal paths with respect to the x-axis, and the remaining graphs using the uphill drawings of their spinal paths with respect to the y-axis. Consequently, the bend complexity decreases to $O(f(n, \lceil t/2 \rceil))$. We can improve this bound further by observing that we are free to choose any arbitrary vertex labelling for G while creating the initial sequence of tuples. Instead of using an arbitrary labelling, we could label the vertices according to their ordering on some spinal path, which would reduce the bend complexity to $O(f(n, \lceil (t-2)/2 \rceil))$. As shown in Sections 3.2.1 and 3.2.2, if $t \in \{3, 4\}$, then $f(n, \lceil (t-2)/2 \rceil) \in O(\sqrt{n})$.

▶ **Theorem 4.** *Every n-vertex graph G of thickness $t \geq 3$ admits a drawing on t layers with bend complexity $O(\beta)$, where β is the minimum integer such that the sequence of n tuples obtained from the spinal paths of G can be partitioned into β monotone subsequences. Furthermore, if $t \in \{3, 4\}$, then $\beta \in O(\sqrt{n})$.*

A careful analysis of the generalization of Erdös-Szekeres [14] theorem gives an $O(\sqrt{2}^t \cdot n^{1-(1/\gamma)})$ upper bound on the bend complexity, where $\gamma = 2^{\lceil (t-2)/2 \rceil}$. For the details, we refer to the full version of the paper [11].

4 Drawing Graphs of Linear Arboricity k

In this section we construct polyline drawings, where the layer number and bend complexities are functions of the linear arboricity of the input graphs. We show that the bandwidth of a graph can be bounded in terms of its linear arboricity and the number of vertices, and then the result follows from an application of Lemma 1.

The *bandwidth* of an n-vertex graph $G = (V, E)$ is the minimum integer b such that the vertices can be labelled using distinct integers from 1 to n satisfying the condition that for any edge $(u, v) \in E$, the absolute difference between the labels of u and v is at most b. The following lemma proves an upper bound on the bandwidth of graphs.

▶ **Lemma 5.** *Given an n-vertex graph $G = (V, E)$ with linear arboricity k, the bandwidth of G is at most $\frac{3(k-1)n}{(4k-2)}$.*

Proof. Without loss of generality assume that G is a union of k spanning paths $P_1, \ldots, P_k$. For any ordered sequence σ, let $\sigma(i)$ be the element at the ith position, and let $|\sigma|$ be the number of elements in σ. We now construct an ordered sequence $\sigma = \sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_k \circ \sigma_{k+1}$ of the vertices in V, as follows.

σ_1: We initially place the first x vertices of P_1 in the sequence, where the exact value of x is to be determined later.
σ_2: We then place the vertices that are neighbors of σ_1 in P_2, in order, i.e., we first place the neighbors of $\sigma_1(1)$, then the neighbors of $\sigma_1(2)$ that have not been placed yet, and so on.
σ_i: For each $i = 3, \ldots, k$, we place the vertices that are neighbors of σ_1 in P_i in order.
σ_{k+1}: We next place the remaining vertices of P_1 in order.

Figure 6(a) illustrates an example for three paths with $x = 2$. Observe that $|\sigma_1| \leq x$, and $|\sigma_t| \leq 2x$, where $1 < t \leq k$. We now compute an upper bound on the bandwidth of G using the vertex ordering of σ.

For any i, j, where $1 \leq i < j \leq k + 1$, let $\sigma_{i,j}$ be the sequence $\sigma_i \circ \ldots \circ \sigma_j$. The edges of P_1 that are in σ_1 have bandwidth 1, and those that are in $\sigma_1(x) \circ \sigma_{2,k+1}$ have bandwidth at most $(n - x)$, e.g., see Figure 6(b). Now let (v, w) be an edge of G that does not belong to P_1. We compute the bandwidth of (v, w) considering the following cases.

Case 1. If none of v and w belongs to σ_1, then the bandwidth of (v, w) is at most $(n - x)$.
Case 2. If both v and w belong to σ_1, then the bandwidth of (v, w) is at most x.
Case 3. If at most one of v and w belongs to σ_1, then without loss of generality assume that v belongs to σ_1. Since (v, w) does not belong to P_1, we may assume that w belongs to the path P_t, where $1 < t \leq k$. By the construction of σ, w belongs to $\sigma_{1,t}$, e.g., see Figure 6(b). Without loss of generality assume that w belongs to σ_r, where $1 < r \leq t$. Let v be the qth vertex in the sequence σ. Then the position of w cannot be more than $q + 2x \cdot (r - 2) + 2q$, where the term $2x \cdot (r - 2)$ corresponds to the length of $\sigma_2 \circ \ldots \circ \sigma_{r-1}$. Therefore, the bandwidth of the edge (v, w) is at most $2x \cdot (r - 2) + 2q \leq 2x(r - 1) \leq 2x(t - 1)$.

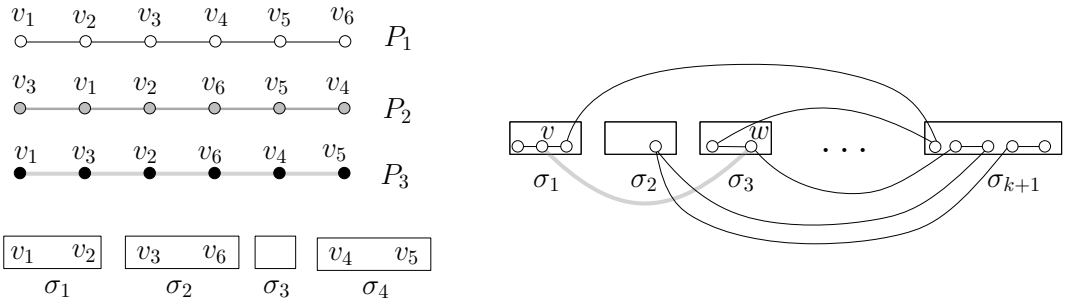


Figure 6 (a) Construction of σ. (b) A schematic representation of P_1 and (v, w), where (v, w) belongs to P_3.

Observe that the bandwidth of the edges of P_1 is upper bounded by $(n - x)$. The bandwidth of any edge that belongs to P_t, where $1 < t \leq k$ is at most $\max\{n - x, 2x(k - 1)\}$. Consequently, the bandwidth of G is at most $\max\{n - x, 2x(k - 1)\} \leq \frac{(2k-2)n}{(2k-1)}$, where $x = \frac{n}{(2k-1)}$. ◀

The following theorem is immediate from the proof of Lemmas 1 and 5.

▶ **Theorem 6.** *Every n-vertex graph with linear arboricity k can be drawn on k layers with at most $\frac{3(k-1)n}{(4k-2)} < 0.75n$ bends per edge.*

5 Conclusions

In this paper we have developed algorithms to draw graphs on few planar layers and with low bend complexity. Although our algorithms do not construct drawings with integral coordinates, it is straightforward to see that these drawings can also be constructed on polynomial-size integer grids, where all vertices and bends have integral coordinates. We leave the task of finding compact grid drawings achieving the same upper bounds as a direction for future research.

We believe our upper bounds on bend complexity to be nearly tight, but we require more evidence to support this intuition. The only related lower bound is that of Pach and Wenger [20], who showed that given a planar graph G and a unique location to place each vertex of G, $\Omega(n)$ bends are sometimes necessary to construct a planar polyline drawing of G with the given vertex locations. Therefore, a challenging research direction would be to prove tight lower bounds on the bend complexity while drawing thickness-t graphs on t layers.

Acknowledgement. We thank anonymous reviewers for many constructive comments and suggestions.

References

1 Melanie Badent, Emilio Di Giacomo, and Giuseppe Liotta. Drawing colored graphs on colored points. *Theoretical Computer Science*, 408(2-3):129–142, 2008.

2 Reuven Bar-Yehuda and Sergio Fogel. Partitioning a sequence into few monotone subsequences. *Acta Informatica*, 35(5):421–440, 1998.

3 János Barát, Jiří Matoušek, and David R. Wood. Bounded-degree graphs have arbitrarily large geometric thickness. *Electronic Journal of Combinatorics*, 13(R3), 2006.

4 Thomas Bläsius, Stephen G. Kobourov, and Ignaz Rutter. Simultaneous embedding of planar graphs. In Roberto Tamassia, editor, *Handbook of Graph Drawing and Visualization*, chapter 11, pages 349–380. CRC Press, August 2013.

5 Peter Braß, Eowyn Cenek, Christian A. Duncan, Alon Efrat, Cesim Erten, Dan Ismailescu, Stephen G. Kobourov, Anna Lubiw, and Joseph S. B. Mitchell. On simultaneous planar graph embeddings. *Computational Geometry*, 36(2):117–130, 2007.

6 Michael B. Dillencourt, David Eppstein, and Daniel S. Hirschberg. Geometric thickness of complete graphs. *Journal of Graph Algorithms and Applications*, 4(3):5–17, 2000.

7 Vida Dujmović and David R. Wood. Graph treewidth and geometric thickness parameters. *Discrete & Computational Geometry*, 37(4):641–670, 2007.

8 Christian A. Duncan. On graph thickness, geometric thickness, and separator theorems. *Computational Geometry*, 44(2):95–99, 2011.

9 Christian A. Duncan, David Eppstein, and Stephen G. Kobourov. The geometric thickness of low degree graphs. In *Proceedings of the 20th ACM Symposium on Computational Geometry (SoCG)*, pages 340–346. ACM, 2004.

10 Stephane Durocher, Ellen Gethner, and Debajyoti Mondal. Thickness and colorability of geometric graphs. *Computational Geometry: Theory and Applications*, 56:1–18, 2016.

11 Stephane Durocher and Debajyoti Mondal. Relating graph thickness to planar layers and bend complexity, 2016. URL: `http://arxiv.org/abs/1602.07816`.

12 Hikoe Enomoto and Miki Shimabara Miyauchi. Embedding graphs into a three page book with $O(m \log n)$ crossings of edges over the spine. *SIAM Journal on Discrete Mathematics*, 12(3):337–341, 1999.

13 David Eppstein. Separating thickness from geometric thickness. In János Pach, editor, *Towards a Theory of Geometric Graphs*. American Mathematical Society, 2004.

14 Paul Erdös and George Szekeres. A combinatorial theorem in geometry. *Compositio Math.*, 2:463–470, 1935.

15 Cesim Erten and Stephen G. Kobourov. Simultaneous embedding of planar graphs with few bends. *Journal of Graph Algorithms and Applications*, 9(3):347–364, 2005.

16 István Fáry. On straight-line representation of planar graphs. *Acta Sci. Math. (Szeged)*, 11:229–233, 1948.

17 Emilio Di Giacomo and Giuseppe Liotta. Simultaneous embedding of outerplanar graphs, paths, and cycles. *International Journal of Computational Geometry & Applications*, 17(2):139–160, 2007.

18 Taylor Gordon. Simultaneous embeddings with vertices mapping to pre-specified points. In *Proceedings of the 18th Annual International Conference on Computing and Combinatorics (COCOON)*, volume 7434 of *LNCS*, pages 299–310. Springer, 2012.

19 Joseph B. Kruskal. Monotonic subsequences. *Proceedings of the American Mathematical Society*, 4:264–274, 1953.

20 János Pach and Rephael Wenger. Embedding planar graphs at fixed vertex locations. *Graphs & Combinatorics*, 17(4):717–728, 2001.

21 David R. Wood. Geometric thickness in a grid. *Discrete Mathematics*, 273(1-3):221–234, 2003.

Optimal Approximate Matrix Product in Terms of Stable Rank*

Michael B. Cohen[†1], Jelani Nelson[‡2], and David P. Woodruff[§3]

1 **MIT, Cambridge, USA**
micohen@mit.edu

2 **Harvard University, Cambridge, USA**
minilek@seas.harvard.edu

3 **IBM Research Almaden, San Jose, USA**
dpwoodru@us.ibm.com

Abstract

We prove, using the subspace embedding guarantee in a black box way, that one can achieve the spectral norm guarantee for approximate matrix multiplication with a dimensionality-reducing map having $m = O(\tilde{r}/\epsilon^2)$ rows. Here $\tilde{r}$ is the maximum stable rank, i.e., the squared ratio of Frobenius and operator norms, of the two matrices being multiplied. This is a quantitative improvement over previous work of [Magen and Zouzias, SODA, 2011] and [Kyrillidis et al., arXiv, 2014] and is also optimal for any oblivious dimensionality-reducing map. Furthermore, due to the black box reliance on the subspace embedding property in our proofs, our theorem can be applied to a much more general class of sketching matrices than what was known before, in addition to achieving better bounds. For example, one can apply our theorem to efficient subspace embeddings such as the Subsampled Randomized Hadamard Transform or sparse subspace embeddings, or even with subspace embedding constructions that may be developed in the future.

Our main theorem, via connections with spectral error matrix multiplication proven in previous work, implies quantitative improvements for approximate least squares regression and low rank approximation, and implies faster low rank approximation for popular kernels in machine learning such as the gaussian and Sobolev kernels. Our main result has also already been applied to improve dimensionality reduction guarantees for k-means clustering, and also implies new results for nonparametric regression.

Lastly, we point out that the proof of the "BSS" deterministic row-sampling result of [Batson et al., SICOMP, 2012] can be modified to obtain deterministic row-sampling for approximate matrix product in terms of the stable rank of the matrices. The original "BSS" proof was in terms of the rank rather than the stable rank.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems

Keywords and phrases subspace embeddings, approximate matrix multiplication, stable rank, regression, low rank approximation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.11

* This is an extended abstract. The URL http://arxiv.org/abs/1507.02268 has our full version.

† MBC was supported by an Akamai Presidential Fellowship and NSF grant CCF-1111109.

‡ JN was supported by NSF grant IIS-1447471 and CAREER CCF-1350670, ONR grant N00014-14-1-0632 and Young Investigator N00014-15-1-2388, and a Google Faculty Research Award.

§ DPW was supported by XDATA program of the Defense Advanced Research Projects Agency (DARPA), administered through Air Force Research Laboratory FA8750-12-C-0323.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 11; pp. 11:1–11:14

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

1 Introduction

Much recent work has successfully utilized randomized dimensionality reduction techniques to speed up solutions to linear algebra problems, with applications in machine learning, statistics, optimization, and several other domains; see the recent monographs [19, 32, 42] for more details. In our work here, we give new spectral norm guarantees for approximate matrix multiplication (AMM). Aside from AMM being interesting in its own right, it has become a useful primitive in the literature for analyzing algorithms for other large-scale linear algebra problems as well. We show applications of our new guarantees to speeding up standard algorithms for generalized regression and low-rank approximation problems. We also describe applications of our results to k-means clustering (discovered in [11]) and nonparametric regression [43].

In AMM we are given A, B each with a large number of rows n, and the goal is to compute some matrix C such that $\|C - A^TB\|_X$ is "small", for some norm $\|\cdot\|_X$. Furthermore, we would like to compute C much faster than the usual time required to exactly compute A^TB.

Work on randomized methods for AMM began with [15], which focused on $\|\cdot\|_X = \|\cdot\|_F$, i.e., Frobenius norm. They showed by picking an appropriate sampling matrix $\Pi \in \mathbb{R}^{m\times n}$, $\|(\Pi A)^T(\Pi B) - A^TB\|_F \le \varepsilon\|A\|_F\|B\|_F$ with good probability if $m = \Omega(1/\varepsilon^2)$. By a *sampling matrix*, we mean the rows of Π are independent, and each row is all zero except for a 1 in a (non-uniformly) random location. If $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times p}$, note $(\Pi A)^T(\Pi B)$ can be computed in $O(mdp)$ time once ΠA and ΠB are formed, as opposed to the straightforward $O(ndp)$ time to compute A^TB.

Frobenius error was also later achieved in [38] via a different approach, with some later optimizations in [22]. This was not via sampling, but rather to use Π drawn from a distribution satisfying an "oblivious Johnson-Lindenstrauss (JL)" guarantee, i.e. a distribution $\mathcal{D}$ over $\mathbb{R}^{m\times n}$ satisfying the following condition for some $\varepsilon, \delta \in (0, 1/2)$: $\forall x \in \mathbb{R}^n$, $\mathbb{P}_{\Pi\sim\mathcal{D}}(|\|\Pi x\|_2^2 - \|x\|_2^2| > \varepsilon\|x\|_2^2) < \delta$. Such a matrix Π can be taken with $m = O(\varepsilon^{-2}\log(1/\delta))$ [21]. Furthermore, one can take Π to be a Fast JL transform [1] (or any of the follow-up improvements [2, 24, 36, 4, 20]) or a sparse JL transform [14, 22] to speed up the computation of ΠA and ΠB. One could also use the Thorup-Zhang sketch [40] combined with a certain technique of [28] (see [42, Theorem 2.10] for details) to efficiently boost success probability.

Other than Frobenius error, the main other error guarantee investigated in previous work is spectral error. That is, we would like $\|C - A^TB\|$ to be small, where $\|M\|$ denotes the largest singular value of M. If one is interested in applying A^TB to some set of input vectors then this type of error is the most meaningful, since $\|C - A^TB\|$ being small is equivalent to $\|Cx\| \approx \|A^TBx\|$ for any x. The first work along these lines was again by [15], who gave a procedure based on entry-wise sampling of the entries of A and B. The works [17, 39] showed that row-sampling according to leverage scores also provides the desired guarantee with few samples.

Then [38], combined with a quantitative improvement in [9], showed that one can take a Π drawn from an oblivious JL distribution with $\delta = 2^{-\Theta(r)}$ where $r(\cdot)$ denotes rank and $r = r(A) + r(B)$. Then for Π with $m = O((r + \log(1/\delta))/\varepsilon^2)$, with probability at least $1 - \delta$ over Π,

$$\|(\Pi A)^T(\Pi B) - A^TB\| \le \varepsilon\|A\|\|B\|. \tag{1}$$

As we shall see shortly via a very simple lemma (Lemma 3), a sufficient deterministic condition implying Eq. (1) is that Π is an $O(\varepsilon)$-*subspace embedding* for the r-dimensional subspace spanned by the columns of A, B. The notion of a subspace embedding was introduced by [38].

▶ **Definition 1.** Π is an *ε-subspace embedding* for $U \in \mathbb{R}^{n\times r}$, $U^TU = I$, if Π satisfies Eq. (1) with $A = B = U$, i.e. $\|(\Pi U)^T(\Pi U) - I\| \leq \varepsilon$. This is equivalent to $\forall x \in \mathbb{R}^r,\ (1-\varepsilon)\|x\|_2^2 \leq \|\Pi U x\|_2^2 \leq (1+\varepsilon)\|x\|_2^2$, i.e. Π preserves norms of all vectors in the subspace spanned by the columns U.

An *(ε, δ, r)-oblivious subspace embedding (OSE)* is a distribution $\mathcal{D}$ over $\mathbb{R}^{m\times n}$ such that $\forall U \in \mathbb{R}^{n\times r},\ U^TU = I$, it holds that $\mathbb{P}_{\Pi\sim\mathcal{D}}(\|(\Pi U)^T(\Pi U) - I\| > \varepsilon) < \delta$.

Fast subspace embeddings Π, i.e. such that the products ΠA and ΠB can be computed quickly, are known using variants on the Fast JL transform such as the Subsampled Randomized Hadamard Transform (SRHT) [38, 29, 41, 30], or via sparse subspace embeddings [9, 33, 34, 27, 12, 10]. We also refer the reader to a slightly improved analysis of the SRHT in our full version [13]. In most applications it is important to have a fast subspace embedding to shrink the time it takes to transform the input data to a lower-dimensional form. The SRHT is a randomized Π with the property that ΠA can be computed in time $O(nd\log n)$. The sparse subspace embedding constructions have some parameter m rows and exactly s non-zero entries per column, so that ΠA can be computed in time $O(s \cdot nnz(A))$, where $nnz(\cdot)$ is the number of non-zero entries, and there is a tradeoff in the upper bounds between m and s.

An issue addressed by the work of [31] is that of robustness. As stated above, achieving Eq. (1) requires Π be a subspace embedding for an r-dimensional subspace. However, consider the case when A (and similarly for B) is of high rank but can be expressed as the sum of a low-rank matrix plus high-rank noise of small magnitude, i.e., $A = \tilde{A} + E_A$ for $\tilde{A}$ of rank $r(\tilde{A}) \ll r$, and where $\|E_A\|$ is very small but E_A has high (even full) rank. One would hope the noise could be ignored, but standard results require Π to have a number of rows at least as large as r, regardless of how small the magnitude of the noise is. Another case of interest (as we will see in Section 3) is when A and B are each of high rank, but their singular values decay at some appropriate rate. As discussed in Section 3, in several applications where AMM is not the final goal but rather is used as a primitive in analyzing an algorithm for some other problem (such as k-means clustering or nonparametric regression), the matrices that arise do indeed have such decaying singular values.

The work [31] remedied this by considering the *stable ranks* $\tilde{r}(A), \tilde{r}(B)$ of A and B. Define $\tilde{r}(A) = \|A\|_F^2/\|A\|^2$. Note $\tilde{r}(A) \leq r(A)$ always, but can be much less if A has a small tail of singular values. Let $\tilde{r}$ denote $\tilde{r}(A) + \tilde{r}(B)$. Among other results, [31] showed that to achieve Eq. (1) with good probability, one can take Π to be a random (scaled) sign matrix with either $m = \Omega(\tilde{r}/\varepsilon^4)$ or $m = \Omega(\tilde{r}\log(d+p)/\varepsilon^2)$ rows. As noted in follow-up work [25], both the $1/\varepsilon^4$ dependence and the $\log(d+p)$ factor are undesirable. In their data-driven low dimensional embedding application, they wanted a dimension m independent of the original dimensions, which are assumed much larger than the stable rank, and also wanted lower dependence on $1/\varepsilon$. To this end, [25] defined the *nuclear rank* as $\tilde{nr}(A) = \|A\|_*/\|A\|$ and showed $m = \Omega(\tilde{nr}/\varepsilon^2)$ rows suffice for $\tilde{nr} = \tilde{nr}(A) + \tilde{nr}(B)$. Here $\|A\|_*$ is the nuclear norm, i.e., sum of singular values of A. Since $\|A\|_F^2$ is the sum of squared singular values, it is straightforward to see that $\tilde{nr}(A) \geq \tilde{r}(A)$ always. Thus there is a tradeoff: the stable rank guarantee is worsened to nuclear rank, but dependence on $1/\varepsilon$ is improved to quadratic.

We show switching to the weaker $\tilde{nr}$ guarantee is unnecessary by showing quadratic dependence on $1/\varepsilon$ holds even with stable rank. This answers the main open question of [31, 25].

To state our results in a more natural way, we rephrase our main result to say that we achieve

$$\|(\Pi A)^T(\Pi B) - A^T B\| \leq \varepsilon\sqrt{\left(\|A\|^2 + \frac{\|A\|_F^2}{k}\right)\left(\|B\|^2 + \frac{\|B\|_F^2}{k}\right)}. \tag{2}$$

for an arbitrary $k \geq 1$, and we do so by using subspace embeddings for $O(k)$-dimensional subspaces in a certain black box way (which will be made precise soon) regardless of the ranks of A, B.

▶ Remark 1. Note that our previously stated main contribution is equivalent, since one could set $k = \tilde{r}(A) + \tilde{r}(B)$ to arrive at the conclusion that subspace embeddings for $O(\tilde{r})$-dimensional subspaces yield the guarantee in Eq. (1). Alternatively one could obtain the Eq. (2) guarantee via Eq. (1) with error parameter $\varepsilon' = \Theta(\varepsilon \cdot \min\{1, \sqrt{(\tilde{r}(A) \cdot \tilde{r}(B))/k}\})$.

Henceforth, we use the following definition.

▶ **Definition 2.** For conforming matrices A^T, B, we say Π satisfies the (k, ε)-*approximate spectral norm matrix multiplication property ((k, ε)-AMM) for* A, B if Eq. (2) holds. If Π is random and satisfies (k, ε)-AMM with probability $1 - \delta$ for any fixed A, B, then we say Π satisfies (k, ε, δ)-AMM.

Our main contribution: We give two different characterizations for Π supporting (k, ε)-AMM, both of which imply (k, ε, δ)-AMM Π having $m = O((k + \log(1/\delta))/\varepsilon^2)$ rows. The first characterization applies to any OSE distribution for which a moment bound has been proven for $\|(\Pi U)^T(\Pi U) - I\|$ (which is true for the best analyses of all known OSE's). In this case, we show a black box theorem: any $(\varepsilon, \delta, 2k)$-OSE provides (k, ε, δ)-AMM. Since matrices with subgaussian entries and $m = \Omega((k + \log(1/\delta))/\varepsilon^2)$ are $(\varepsilon, \delta, 2k)$-OSE's, our originally stated main result follows. This result is optimal, since [35] shows any randomized distribution over Π with m rows having the (k, ε, δ)-AMM property must have $m = \Omega((k + \log(1/\delta))/\varepsilon^2)$ (the hard instance there is when $A = B = U$ has orthonormal columns, and thus rank and stable rank are equal).

Our second characterization (appearing in the full version) identifies certain deterministic conditions which, if satisfied by Π, imply the desired (k, ε)-AMM property. These conditions are of the form: (1) Π should preserve a certain set of $O(\log(1/\varepsilon))$ different subspaces of varying dimensions (all depending on k, ε and not on the ranks of A, B) with varying distortions, and (2) for a certain two matrices in our analysis, left-multiplication by Π should not increase their operator norms by more than an $O(1)$ factor. These conditions are chosen carefully so that matrices with subgaussian entries and $m = \Omega(k/\varepsilon^2)$ satisfy all conditions simultaneously with high probability, again thus proving our main result while also suggesting that the conditions we have identified are the "right" ones.

Due to the black box reliance on the subspace embedding primitive in our proofs, Π need not only be a subgaussian map. Thus not only do we improve on m compared with previous work, but also in terms of the general class of Π our result applies to. For example given our first characterization, not only does it suffice to use a random sign matrix with $\Omega(k/\varepsilon^2)$ rows, but in fact one can apply our theorem to more efficient subspace embeddings such as the SRHT or sparse subspace embeddings, or even constructions discovered in the future. That is, one can automatically transfer bounds proven for the subspace embedding property to the (k, ε)-AMM property. Thus, for example, the best known SRHT analysis (see the full version) implies (k, ε, δ)-AMM for $m = \Omega((k + \log(1/(\varepsilon\delta))\log(k/\delta))/\varepsilon^2)$ rows. For sparse subspace embeddings, the analysis in [10] implies $m = \Omega(k\log(k/\delta)/\varepsilon^2)$ suffices

with $s = O(\log(k/\delta)/\varepsilon)$ non-zeroes per column of Π. The only reason for the $\log k$ loss in m for these particular distributions is not due to our theorems, but rather due to the best analyses for the simpler *subspace embedding* property in previous work already incurring the extra $\log k$ factor (note being a subspace embedding for a k-dimensional subspace is simply a special case of (k, ε)-AMM where $A = B = U$ has k orthonormal columns). In the case of the SRHT, this extra $\log k$ factor is actually necessary [41]; for sparse subspace embeddings, it is conjectured that the $\log k$ factor can be removed and that $m = \Omega((k + \log(1/\delta))/\varepsilon^2)$ actually suffices to obtain an OSE [34, Conjecture 14]. We also discuss in Remark 2 that one can set Π to be $\Pi_1 \cdot \Pi_2$ where Π_1 has subgaussian entries with $O(k/\varepsilon^2)$ rows, and Π_2 is some other fast OSE (such as the SRHT or sparse subspace embedding), and thus one could obtain the best of both worlds: (1) Π has $O(k/\varepsilon^2)$ rows, and (2) can be applied to any $A \in \mathbb{R}^{n\times d}$ in time $T + O(km'd/\varepsilon^2)$, where T is the (fast) time to apply Π_2 to A, and m' is the number of rows of Π_2. For example, by appropriate composition as discussed in Remark 2, Π can have $O(k/\varepsilon^2)$ rows and support multiplying ΠA for $A \in \mathbb{R}^{n\times d}$ in time $O(nnz(A)) + \tilde{O}(\varepsilon^{-O(1)}(k^3 + k^2 d))$.

We also observe the proof of the main result of [3] can be modified to show that given any A, B each with n rows, and given any $\varepsilon \in (0, 1/2)$, there exists a diagonal matrix $\Pi \in \mathbb{R}^{n\times n}$ with $O(k/\varepsilon^2)$ non-zero entries, and that can be computed by a deterministic polynomial time algorithm, achieving (k, ε)-AMM. The original work of [3] achieved Eq. (1) with $m = O(r/\varepsilon^2)$ for r being the sum of ranks of A, B. The work [3] stated their result for the case $A = B$, but the general case of potentially unequal matrices reduces to this case; see Section 4. Our observation also turns out to yield a stronger form of [23, Theorem 3.3]; also see Section 4.

As mentioned, aside from AMM being interesting on its own, it is a useful primitive widely used in analyses of algorithms for several other problems, including k-means clustering [5, 11], nonparametric regression [43], linear least squares regression and low-rank approximation [38], approximating leverage scores [16], and several other problems (see [42] for a recent summary). For all these, analyses of correctness for algorithms based on dimensionality reduction via some Π rely on Π satisfying AMM for certain matrices in the analysis.

After making certain quantitative improvements to connections between AMM and applications, and combining them with our main result, in Section 3 we obtain the following new results.

1. **Generalized regression:** Given $A \in \mathbb{R}^{n\times d}$ and $B \in \mathbb{R}^{n\times p}$, consider the problem of computing $X^* = argmin_{X\in\mathbb{R}^{d\times p}} \|AX - B\|$. It is standard that $X^* = (A^TA)^+A^TB$ where $(\cdot)^+$ is the Moore-Penrose pseudoinverse. The bottleneck here is computing A^TA, taking $O(nd^2)$ time. A popular approach is to instead compute $\tilde{X} = ((\Pi A)^T(\Pi A))^+(\Pi A)^T\Pi B$, i.e., the minimizer of $\|\Pi AX - \Pi B\|$. Note that computing $(\Pi A)^T(\Pi A)$ (given ΠA) only takes a smaller $O(md^2)$ amount of time. We show that if Π satisfies $(k, O(\sqrt{\varepsilon}))$-AMM for $U_A, P_{\bar{A}}B$, and is also an $O(1)$-subspace embedding for a certain $r(A)$-dimensional subspace (see Theorem 7), then

   $$\|A\tilde{X} - B\|^2 \leq (1 + \varepsilon)\|P_AB - B\|^2 + (\varepsilon/k)\|P_AB - B\|_F^2$$

   where P_A is the orthogonal projection onto the column space of A, $P_{\bar{A}} = I - P_A$, and U_A has orthonormal columns forming a basis for the column space of A. The punchline is that if the regression error $P_{\bar{A}}B$ has high actual rank but stable rank only on the order of $r(A)$, then we obtain multiplicative spectral norm error with Π having fewer rows. Generalized regression is a natural extension of the case when B is a vector, and arises for

example in Regularized Least Squares Classification, where one has multiple (non-binary) labels, and for each label one creates a column of B; see e.g. [7] for this and variations.

2. **Low-rank approximation:** We are given $A \in \mathbb{R}^{n\times d}$ and integer $k \geq 1$, and we want to compute $A_k = argmin_{r(X)\leq k} \|A - X\|$. The Eckart-Young theorem implies A_k is obtained by truncating the SVD of A to the top k singular vectors. The standard way to use dimensionality reduction for speedup, introduced in [38], is to let $S = \Pi A$ then compute $\tilde{A} = AP_S$. Then return $\tilde{A}_k$, the best rank-k approximation of $\tilde{A}$, instead of A_k (it is known $\tilde{A}_k$ can be computed more efficiently than A_k; see [8, Lemma 4.3]). We show if Π satisfies $(k, O(\sqrt{\varepsilon}))$-AMM for U_k and $A - A_k$, and is a $(1/2)$-subspace embedding for the column space of A_k, then

$$\|\tilde{A}_k - A\|^2 \leq (1+\varepsilon)\|A - A_k\|^2 + (\varepsilon/k)\|A - A_k\|_F^2.$$

The punchline is that if the stable rank of the tail $A - A_k$ is on the same order as the rank parameter k, then standard algorithms from previous work for Frobenius multiplicative error actually in fact also provide *spectral* multiplicative error. This property indeed holds for any k for popular kernel matrices in machine learning such as the gaussian and Sobolev kernels (see [37] and Examples 2 and 3 of [43]), and low-rank approximation of kernel matrices has been applied to several machine learning problems; see [18] for a discussion.

We also explain in Section 3 how our result has already been applied in recent work on dimensionality reduction for k-means clustering [12], and how it generalizes results in [43] on dimensionality reduction for nonparametric regression to use a larger class of embeddings Π.

1.1 Preliminaries and notation

We frequently use the singular value decomposition (SVD). For a matrix $A \in \mathbb{R}^{n\times d}$ of rank r, consider the compact SVD $A = U_A \Sigma_A V_A^T$ where $U_A \in \mathbb{R}^{n\times r}$ and $V_A \in \mathbb{R}^{d\times r}$ each have orthonormal columns, and Σ_A is diagonal with strictly positive diagonal entries (the singular values of A). We assume $(\Sigma_A)_{i,i} \geq (\Sigma_A)_{j,j}$ for $i < j$. We let $P_A = U_A U_A^T$ denote the orthogonal projection operator onto the column space of A. We use $\text{span}(A)$ to refer to the subspace spanned by A's columns.

Often for a matrix A we write A_k as the best rank-k approximation to A under Frobenius or spectral error (obtained by writing the SVD of A then setting all $(\Sigma_A)_{i,i}$ to 0 for $i > k$). We often denote $A - A_k$ as $A_{\bar{k}}$. For matrices with orthonormal columns, such as U_A, $(U_A)_k$ denotes the $n \times k$ matrix formed by removing all but the first k columns of U. When A is understood from context, we often write $U\Sigma V^T$ instead of $U_A \Sigma_A V_A^T$, and U_k to denote $(U_A)_k$ (and Σ_k for $(\Sigma_A)_k$, etc.).

2 Analysis of matrix multiplication for stable rank

First we record a simple lemma relating subspace embeddings and AMM; proof in full version [13].

▶ **Lemma 3.** *Let $E = \text{span}\{A, B\}$, and let Π be an ε-subspace embedding for E. Then Eq. (1) holds.*

Lemma 3 implies that if A, B each have rank at most r, it suffices for Π to have $\Omega(r/\varepsilon^2)$ rows.

In the following subsection, we give one characterization for Π to provide (k, ε, δ)-AMM, only requiring Π to have $\Omega((k + \log(1/\delta))/\varepsilon^2)$ rows, independent of r. The other characterization also allows for this many rows, but is different in that it identifies certain deterministic conditions such that, if those hold, Π provides (k, ε)-AMM. Thus, the second characterization can even apply to deterministic Π such as the truncated SVD. We provide this second characterization only in the full version.

2.1 Characterization for (k, ε, δ)-AMM via a moment property

Here we provide a way to obtain (k, ε)-AMM for any Π whose subspace embedding property has been established using the moment method, e.g. sparse subspace embeddings [33, 34, 10], dense subgaussian matrices (as analyzed in the full version), or even the SRHT (also, as analyzed in the full version). Our approach in this subsection is inspired by the introduction of the "JL-moment property" in [22] to analyze approximate matrix multiplication with Frobenius error. The following is a generalization of [22, Definition 6.1], which was only concerned with $d = 1$.

▶ **Definition 4.** A distribution $\mathcal{D}$ over $\mathbb{R}^{m\times n}$ has $(\varepsilon, \delta, d, \ell)$-*OSE moments* if for all matrices $U \in \mathbb{R}^{n\times d}$ with orthonormal columns, $\mathbb{E}_{\Pi\sim\mathcal{D}} \left\|(\Pi U)^T(\Pi U) - I\right\|^\ell < \varepsilon^\ell \cdot \delta$.

The acronym "OSE" refers to *oblivious subspace embedding*, a term coined in [34] to refer to distributions over Π yielding a subspace embedding for any fixed subspace of a particular bounded dimension with high probability. We start with a simple lemma; proof in full version.

▶ **Lemma 5.** *Suppose $\mathcal{D}$ satisfies the $(\varepsilon, \delta, 2d, \ell)$-OSE moment property and A, B (1) have the same number of rows, and (2) sum of ranks at most $2d$. Then $\mathbb{E}_{\Pi\sim\mathcal{D}} \left\|(\Pi A)^T(\Pi B) - A^T B\right\|^\ell < \varepsilon^\ell \|A\|^\ell \|B\|^\ell \delta$.*

Then, just as [22, Theorem 6.2] showed that having OSE moments with $d = 1$ implies approximate matrix multiplication with Frobenius norm error, here we show that having OSE moments for larger d implies approximate matrix multiplication with operator norm error.

▶ **Theorem 6.** *Given $k, \varepsilon, \delta \in (0, 1/2)$, let $\mathcal{D}$ be any distribution over matrices with n columns with the $(\varepsilon, \delta, 2k, \ell)$-OSE moment property for some $\ell \geq 2$. Then, for any A, B,*

$$\mathop{\mathbb{P}}_{\Pi\sim\mathcal{D}}\left(\|(\Pi A)^T(\Pi B) - A^T B\| > \varepsilon\sqrt{(\|A\|^2 + \|A\|_F^2/k)(\|B\|^2 + \|B\|_F^2/k)}\right) < \delta \tag{3}$$

Proof. We can assume A, B each have orthogonal columns. This is since, via the full SVD, there exist orthogonal matrices R_A, R_B such that AR_A and BR_B each have orthogonal columns. Since neither left nor right multiplication by an orthogonal matrix changes operator norm,

$$\|(\Pi A)^T(\Pi B) - A^T B\| = \|(\Pi A R_A)^T(\Pi B R_B) - (AR_A)^T B R_B\|.$$

Thus, we replace A by AR_A and similarly for B. We may also assume the columns $a_1, a_2, \ldots$ of A are sorted so that $\|a_i\|_2 \geq \|a_{i+1}\|_2$ for all i. Henceforth we assume A has orthogonal columns in this sorted order (and similarly for B, with columns b_i). Now, treat A as a block matrix in which the columns are blocked into groups of size k, and similarly for B (if the number of columns of either A or B is not divisible by k, then pad them

with all-zero columns until they are). Let the spectral norm of the ith block of A be $s_i = \|a_{(i-1)\cdot k+1}\|_2$, and for B denote the spectral norm of the ith block as $t_i = \|b_{(i-1)\cdot k+1}\|_2$. These equalities for A, B hold since their columns are orthogonal and sorted by norm. We claim $\sum_i s_i^2 \leq \|A\|^2 + \|A\|_F^2/k$ (and similarly for $\sum_i t_i^2$). To see this, let the blocks of A be $A'_1, \ldots, A'_q$ where $s_i = \|A'_i\|$. Note $s_1^2 = \|A'_1\| \leq \|A\|$. Also, for $i > 1$ we have $s_i^2 = \|a_{(i-1)\cdot k+1}\|_2^2 \leq \frac{1}{k}\sum_{(i-2)\cdot k+1 \leq j \leq (i-1)\cdot k} \|a_j\|_2^2 = \frac{1}{k}\|A'_{i-1}\|_F^2$. Thus $\sum_{i>1} s_i^2 \leq \|A\|_F^2/k$.

Define $C = (\Pi A)^T(\Pi B) - A^T B$. Let $v_{\{i\}}$ denote the ith block of a vector v (the k-dimensional vector whose entries consist of entries $(i-1)\cdot k+1$ to $i \cdot k$ of v), and $C_{\{i\},\{j\}}$ the (i,j)th block of C, a $k \times k$ matrix (the entries in C contained in the ith block of rows and jth block of columns).

Now, $\|C\| = \sup_{\|x\|=\|y\|=1} x^T C y$. For any such vectors x and y, we define new vectors x' and y' whose coordinates correspond to entire blocks: we let $x'_i = \|x_{\{i\}}\|$, with y' defined analogously. We similarly define C' with entries corresponding to blocks of C, where $C'_{i,j} = \|C_{\{i\},\{j\}}\|$. Then $x^T C y \leq x'^T C' y'$, simply by bounding the contribution of each block. Thus it suffices to upper bound $\|C'\|$, which we bound by its Frobenius norm $\|C'\|_F$. Now, recalling for a random variable X that $\|X\|_\ell$ denotes $(\mathbb{E}\,|X|^\ell)^{1/\ell}$ and using Minkowski's inequality (that $\|\cdot\|_\ell$ is a norm for $\ell \geq 1$),

$$\begin{aligned}\|\|C'\|_F^2\|_{\ell/2} &= \left\|\sum_{i,j} \|(\Pi A'_i)^T(\Pi B'_j) - A'^T_i B'_j\|^2\right\|_{\ell/2} \leq \sum_{i,j} \|\|(\Pi A'_i)^T(\Pi B'_j) - A'^T_i B'_j\|^2\|_{\ell/2} \\ &\leq \sum_{i,j} \varepsilon^2 s_i^2 t_j^2 \cdot \delta^{2/\ell} \text{ (Lemma 5)} = \varepsilon^2 \left(\sum_i s_i^2\right) \cdot \left(\sum_j t_j^2\right) \delta^{2/\ell},\end{aligned}$$

which is at most $(\varepsilon\sqrt{(\|A\|^2 + \frac{\|A\|_F^2}{k})(\|B\|^2 + \frac{\|B\|_F^2}{k})}\delta^{1/\ell})^2$. Now, $\mathbb{E}\,\|C'\|_F^\ell = \|\|C'\|_F^2\|_{\ell/2}^{\ell/2}$, implying

$$\begin{aligned}&\mathbb{P}\left(\|C'\| > \varepsilon\sqrt{(\|A\|^2 + \frac{\|A\|_F^2}{k})(\|B\|^2 + \frac{\|B\|_F^2}{k})}\right) \\ &\leq \mathbb{P}\left(\|C'\|_F > \varepsilon\sqrt{(\|A\|^2 + \frac{\|A\|_F^2}{k})(\|B\|^2 + \frac{\|B\|_F^2}{k})}\right) \\ &< \frac{\mathbb{E}\,\|C'\|_F^\ell}{\left(\varepsilon\sqrt{(\|A\|^2 + \frac{\|A\|_F^2}{k})(\|B\|^2 + \frac{\|B\|_F^2}{k})}\right)^\ell},\end{aligned}$$

and the latter is at most δ. ◀

We now discuss the implications of applying Theorem 6 to specific OSE's.

2.1.1 Subgaussian maps

In the full version we show that if Π has independent subgaussian entries and $m = \Omega((k + \log(1/\delta))/\varepsilon^2)$ rows, then it satisfies the $(\varepsilon, \delta, 2k, \Theta(k + \log(1/\delta)))$ OSE moment property. Thus Theorem 6 applies to show that such Π will satisfy (k, ε, δ)-AMM.

2.1.2 SRHT

The SRHT is the matrix product $\Pi = SHD$ where $D \in \mathbb{R}^{n\times n}$ is $n \times n$ diagonal with independent ± 1 entries on the diagonal, H is a "bounded orthonormal system" (i.e. an

orthogonal matrix in $\mathbb{R}^{n\times n}$ with $\max_{i,j}|H_{i,j}| = O(1/\sqrt{n})$), and the m rows of S are independent and each samples a uniformly random element of $[n]$. Bounded orthonormal systems include the discrete Fourier matrix and the Hadamard matrix; thus such Π exist supporting matrix-vector multiplication in $O(n\log n)$ time. Thus when computing ΠA for some $n\times d$ matrix A, this takes time $O(nd\log n)$ (by applying Π to A column by column). In the full version we show that the SRHT with $m = \Omega((k+\log(1/(\varepsilon\delta))\log(k/\delta))/\varepsilon^2)$ satisfies the $(\varepsilon,\delta,2k,\log(k/\delta))$-OSE moment property, and thus provides (k,ε,δ)-AMM. Interestingly our analysis of the SRHT in the full version seems to be asymptotically tighter than any other analyses in previous work even for the basic subspace embedding property, and even slightly improves the by now standard analysis of the Fast JL transform given in [1].

2.1.3 Sparse subspace embeddings

The sparse embedding distribution with parameters m, s is as follows [9, 34, 22]. The matrix Π has m rows and n columns. The columns are independent, and for each column exactly s uniformly random entries are chosen without replacement and set to $\pm 1/\sqrt{s}$ independently; other entries in that column are set to zero. Alternatively, one could use the CountSketch [6]: the m rows are equipartitioned into s sets of size m/s each. The columns are independent, and in each column we pick exactly one row from each of the s partitions and set the corresponding entry in that column to $\pm 1/\sqrt{s}$ uniformly; the rest of the entries in the column are set to 0. Note ΠA can be multiplied in time $O(s\cdot nnz(A))$, and thus small s is desirable.

It was shown in [33, 34], slightly improving [9], that either of the above distributions satisfies the $(\varepsilon,\delta,k,2)$-OSE moment property for $m = \Omega(k^2/(\varepsilon^2\delta))$, $s = 1$, and hence (k,ε,δ)-AMM (though this particular conclusion follows easily from [22, Theorem 6.2]). It was also shown in [10], improving upon [34], that they satisfy the $(\varepsilon,\delta,k,\log(k/\delta))$-OSE moment property, and hence also (k,ε,δ)-AMM, for $m = \Omega(Bk\log(k/\delta)/\varepsilon^2), s = \Omega(\log_B(k/\delta)/\varepsilon)$ for any $B > 2$. The work [10] does not explicitly discuss the OSE moment property for sparse subspace embeddings, but it is implied; see the full version. It is conjectured that for $B = O(1)$, $m = \Omega((k+\log(1/\delta))/\varepsilon^2)$ should suffice [34, Conjecture 14].

▶ Remark 2. Currently there appears to be a tradeoff: one can either use Π s.t. ΠA can be computed quickly, such as sparse subspace embeddings or the SRHT, but then m is at least $k\log k$. Alternatively one could achieve the optimal $m = O(k/\varepsilon^2)$ using subgaussian Π, but then multiplying by Π is slower: $O(mnd)$ time for $A\in\mathbb{R}^{n\times d}$. However, settling for a tradeoff is unnecessary. One can obtain the "best of both worlds" by composition so that ΠA will have the desired $O(k/\varepsilon^2)$ rows and ΠA computed in time $O(nnz(A)) + \tilde{O}(\varepsilon^{-O(1)}(k^3+k^2d))$; see full version.

3 Applications

Spectral norm approximate matrix multiplication with dimension bounds depending on stable rank has immediate applications for the analysis of generalized regression and low-rank approximation problems. We also point out to the reader recent applications of this result to kernelized ridge regression [43] and k-means clustering [11].

3.1 Generalized regression

Here we consider generalized regression: attempting to approximate a matrix B as AX, with A of rank at most k. Let P_A be the orthogonal projection operator to the column space of A, with $P_{\bar{A}} = I - P$; then the natural best approximation will satisfy $AX = P_A B$. This minimizes

both the Frobenius and spectral norms of $AX - B$. A standard approximation algorithm for this is to replace A and B with sketches ΠA and ΠB, then solve the reduced problem exactly (see e.g. [8], Theorem 3.1). This will produce $\tilde{X} = U_A((\Pi U_A)^T \Pi U_A)^{-1}(\Pi U_A)^T \Pi B$. Below we give a lemma on the guarantees of the sketched solution in terms of properties of Π; proof is in full version.

▶ **Theorem 7.** *If* Π (1) *satisfies the* $(k, \sqrt{\varepsilon/8})$*-approximate spectral norm matrix multiplication property for* $U_A, P_{\bar{A}}B$*, and* (2) *is a* $(1/2)$*-subspace embedding for the column space of* A *(which is implied by* Π *satisfying the spectral norm approximate matrix multiplication property for* U_A *with itself), then* $\|A\tilde{X} - B\|^2 \leq (1+\varepsilon)\|P_A B - B\|^2 + (\varepsilon/k) \cdot \|P_A B - B\|_F^2$.

3.2 Low-rank approximation

Now we apply the generalized regression result from Section 3.1 to obtain a result on low-rank approximation: approximating A in the form $\tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^T$, where $\tilde{U}_k$ has only k columns and both $\tilde{U}_k$ and $\tilde{V}_k$ have orthonormal columns. Here, we consider a previous approach (see e.g. [38]): (1) let $S = \Pi A$, (2) let P_S be the orthogonal projection operator to the row space of S and $\tilde{A} = AP_S$, and (3) compute an SVD of $\tilde{A}$ and keep only the top k singular vectors, then return the resulting low rank approximation $\tilde{A}_k$ of $\tilde{A}$. It turns out computing $\tilde{A}_k$ can be done much more quickly than computing A_k; see details in [8, Lemma 4.3]. Let $A_k, U_k, A_{\bar{k}}$ be as in Section 1.1.

▶ **Theorem 8.** *If* Π (1) *satisfies the* $(k, \sqrt{\varepsilon/8})$*-approximate spectral norm matrix multiplication property for* $U_k, A_{\bar{k}}$*, and* (2) *is a* $(1/2)$*-subspace embedding for the column space of* U_k *then* $\|A - \tilde{A}_k\|^2 \leq (1+\varepsilon)\|A - A_k\|^2 + (\varepsilon/k)\|A - A_k\|_F^2$

3.3 Kernelized ridge regression

In nonparametric regression one is given data $y_i = f^*(x_i) + w_i$ for $i = 1, \ldots, n$, and the goal is to recover a good estimate for the function f^*. Here the y_i are scalars, the x_i are vectors, and the w_i are independent noise, often assumed to be distributed as mean-zero gaussian with some variance σ^2. Unlike linear regression where $f^*(x_i)$ is assumed to take the form $\langle \beta, x \rangle$ for some vector β, in nonparametric regression we allow f^* to be an arbitrary function from some function space. Naturally the goal then is to recover some $\tilde{f}$ from the data that is close to f^* whp over the noise.

Recent work [43] considers the well studied problem of obtaining $\tilde{f}$ so that $\|\tilde{f} - f^*\|_n^2$ is small with high probability over the noise w, where one uses the definition $\|f - g\|_n^2 = \frac{1}{n}\sum_{i=1}^n (f(x_i) - g(x_i))^2$. The work [43] considers the case where f^* comes from a space of functions which is the closure of all functions g expressable as $g(x) = \sum_{i=1}^N \alpha_i k(x, z_i)$ over all N, $\alpha \in \mathbb{R}^N$, and vectors z_i for some PSD kernel function k. See the full version for details, but the punchline is the maximum likelihood estimator for $\tilde{f}$ is then the solution f^{LS} to a Kernelized Ridge Regression (KRR) problem, and $f^{LS}(x)$ can be expressed as a linear combination of kernel evaluations $\sum_{i=1}^n \alpha_i k(x, x_i)$. Then defining matrix K with $K_{i,j} = k(x_i, x_j)$, KRR is equivalent to computing

$$\alpha^{LS} = \underset{\alpha \in \mathbb{R}^n}{argmin} \left\{ \frac{1}{2n}\alpha^T K^2 \alpha - \frac{1}{n}\alpha^T K y + \lambda_n \alpha^T K \alpha \right\} = \left(\frac{1}{n}K^2 + 2\lambda_n K \right)^{-1} \cdot \frac{1}{n} K y,$$

which can be computed in $O(n^3)$ time. The work [43] then focuses on speeding this up, by

instead computing a solution to the lower-dimensional problem

$$\tilde{\alpha}^{LS} = \underset{\alpha\in\mathbb{R}^m}{argmin}\left\{\frac{1}{2n}\alpha^T\Pi K^2\Pi^T\alpha - \frac{1}{n}\alpha^T\Pi K y + \lambda_n\alpha^T\Pi K\Pi^T\alpha\right\}$$
$$= \left(\frac{1}{n}\Pi K^2\Pi^T + 2\lambda_n\Pi K\Pi^T\right)^{-1}\cdot\frac{1}{n}\Pi K y$$

and then returning as $\tilde{f}$ the function specified by the weight vector $\tilde{\alpha} = \Pi^T\tilde{\alpha}^{LS}$. Note that once various matrix products are formed (where the running time complexity depends on the Π being used), one only needs to invert an $m\times m$ matrix thus taking $O(m^3)$ time. They then prove that $\|\tilde{f} - f^*\|_n$ is small with high probability as long as Π satisfies two deterministic conditions (see the proof of Lemma 2 [43, Section 4.1.2], specifically equation (26) in that work): (1) Π is a $(1/2)$-subspace embedding for a particular low-dimensional subspace, and (2) $\|\Pi B\| = O(\|B\|)$ for a particular matrix B of low stable rank (B is UD_2 in [43]). Note by the triangle inequality, $\|\Pi B\| \leq \|(\Pi B)^T\Pi B - B^TB\|^{1/2} + \|B\|$, and thus it suffices for Π to provide AMM for the product B^TB, where B has low stable rank. Item (1) simply requires a subspace embedding in the standard sense, and for item (2) [43] avoided AMM by obtaining a bound on $\|\Pi B\|$ directly by their own analyses for gaussian Π and the SRHT. Our result thus provides a unifying analysis which works for a larger and general class of Π, including for example sparse subspace embeddings.

3.4 *k*-means clustering

In the works [5, 11], the authors considered dimensionality reduction methods for k-means clustering. Recall in k-means clustering one is given n points $x_1,\ldots,x_n\in\mathbb{R}^d$, as well as an integer $k\geq 1$, and the goal is to find k points $y_1,\ldots,y_k\in\mathbb{R}^d$ minimizing $\sum_{i=1}^n\min_{j=1}^k\|x_i - y_j\|_2^2$. One key observation common to both [5, 11] is that k-means clustering is closely related to the problem of low-rank approximation. More specifically, given a partition $\mathcal{P} = \{P_1,\ldots,P_k\}$, define the $n\times k$ matrix $X_{\mathcal{P}}$ by $(X_{\mathcal{P}})_{i,j}$ is $1/\sqrt{|P_j|}$ if $i\in P_j$, and zero otherwise. Let $A\in\mathbb{R}^{n\times d}$ have rows $x_1,\ldots,x_n$. Then the k-means problem can be rewritten as computing $\mathcal{P}^* = \text{argmin}_{\mathcal{P}}\|A - X_{\mathcal{P}}X_{\mathcal{P}}^TA\|_F^2$, where $\mathcal{P}$ ranges over all partitions of $\{1,\ldots,n\}$ into k sets (the y_i are the distinct rows of $X_{\mathcal{P}}X_{\mathcal{P}}^TA$). It is easy to verify the columns of $X_{\mathcal{P}}$ are orthonormal, so $X_{\mathcal{P}}X_{\mathcal{P}}^T$ is the orthogonal projection onto the column space of $X_{\mathcal{P}}$. Thus if one defines $\mathcal{S}$ as the set of all rank k orthogonal projections obtained as $X_{\mathcal{P}}X_{\mathcal{P}}^T$ for some k-partition $\mathcal{P}$, then the above can be rewritten as the *constrained rank-k projection problem* of computing $\mathcal{P}^* = \text{argmin}_{P\in\mathcal{S}}\|(I-P)A\|_F^2$.

The work [11] showed that if $\mathcal{S}$ is any subset of projections of rank at most k (henceforth *rank-k projections*) and $\Pi\in\mathbb{R}^{m\times d}$ satisfies certain technical conditions to be divulged soon, then if $\tilde{P}\in\mathcal{S}$ minimizes the Π-reduced problem $\min_{P\in\mathcal{S}}\|(I-P)A\Pi^T\|_F^2$ up to a factor of γ, then $\tilde{P}$ minimizes the original problem $\min_{P\in\mathcal{S}}\|(I-P)A\|_F^2$ up to $(1+O(\varepsilon))\gamma$.

One set of sufficient conditions for Π is as follows (see [11, Lemma 10]). There is a matrix $B\in\mathbb{R}^{(n+2k)\times d}$ of stable rank $O(k)$, where k is the number of cluster centers y_i above, such that if

$$\|(\Pi B^T)^T(\Pi B^T) - BB^T\| < \varepsilon, \tag{4}$$
$$\text{and } \left|\|\Pi B_2\|_F^2 - \|B_2\|_F^2\right| \leq \varepsilon k \tag{5}$$

then $\tilde{P}$ provides good error as discussed above. Thus for Eq. (4) it suffices for Π to provide $(O(k),\varepsilon/2)$-AMM for B^T, B^T, and our results apply. Obtaining Eq. (5) is much simpler and can be derived from the JL moment property (see the proof of [22, Theorem 6.2]).

Without our results on stable-rank AMM provided in this current work, [11] gave a different analysis, avoiding [11, Lemma 10], which for subgaussian Π required $m = \Theta(k \cdot \log(1/\delta)/\varepsilon^2)$ rows (note the product between k and $\log(1/\delta)$ instead of the sum).

4 Stable rank and row selection

As well as random projections, AMM (and subspace embeddings) by row selection are also common in algorithms. This corresponds to setting Π to a diagonal matrix S with relatively few nonzero entries. Unlike random projections, there are no *oblivious* distributions of such matrices S with universal guarantees. Instead, S must be determined (either randomly or deterministically) from the matrices being embedded.

There are two particularly algorithmically useful methods for obtaining such S. The first is importance sampling: independent random sampling of the rows, but with nonuniform sampling probabilities. For rank-k matrices, $O(k(\log k)/\varepsilon^2)$ samples suffice [17, 39]. The second method is the deterministic selection method given in [3], often called "BSS", choosing only $O(k/\varepsilon^2)$ rows. This still runs in polynomial time, but originally required many expensive linear algebra steps and thus was slower in general; see [26] for runtime improvements.

The method used in [39] (matrix Chernoff bound) can be extended to the stable-rank case, making even the log factor in the number of samples depend only on the stable rank; see the full version for details. We here give an extension of BSS that covers low stable rank matrices as well. The proof is in the full version, and follows by observing that it suffices to just perform a slight modification of the original BSS proof.

▶ **Theorem 9.** *Given two matrices A and B, each with n rows, and an $\varepsilon \in (0, 1)$, there exists a diagonal matrix S with $O(k/\varepsilon^2)$ nonzero entries satisfying the (k, ε)-AMM property for A, B. Such an S can be computed by a polynomial-time algorithm.*

When $A = B$ and $A^T A$ is the identity, this is just the original BSS result. It is also stronger than Theorem 3.3 of [23], implying it when A is the combination of the rows $\sqrt{N/T} \cdot v_i$ from that theorem statement with an extra column containing the costs, and a constant ϵ. The techniques in that paper, on the other hand, can prove a result comparable to Theorem 9, but with the row count scaling as k/ε^3 rather than k/ε^2.

Acknowledgments. We thank Jarosław Błasiok for pointing out the connection between low stable rank approximate matrix multiplication and the analyses in [43].

References

1 Nir Ailon and Bernard Chazelle. The fast Johnson-Lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

2 Nir Ailon and Edo Liberty. An almost optimal unrestricted fast Johnson-Lindenstrauss transform. *ACM Transactions on Algorithms*, 9(3):21, 2013.

3 Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM J. Comput.*, 41(6):1704–1721, 2012.

4 Jean Bourgain. An improved estimate in the restricted isometry problem. *Geometric Aspects of Functional Analysis*, 2116:65–70, 2014.

5 Christos Boutsidis, Anastasios Zouzias, Michael W. Mahoney, and Petros Drineas. Randomized dimensionality reduction for k-means clustering. *IEEE Transactions on Information Theory*, 61(2):1045–1062, 2015.

6 Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

7 Pei-Chun Chen, Kuang-Yao Lee, Tsung-Ju Lee, Yuh-Jye Lee, and Su-Yun Huang. Multiclass support vector classification via coding and regression. *Neurocomputing*, 73(7-9):1501–1512, 2010.

8 Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 205–214, 2009.

9 Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2013. Full version at `http://arxiv.org/abs/1207.6365v4`.

10 Michael B. Cohen. Nearly tight oblivious subspace embeddings by trace inequalities. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 278–287, 2016.

11 Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Mădălina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47th ACM Symposium on Theory of Computing (STOC)*, 2015. Full version at `http://arxiv.org/abs/1410.6801v3`.

12 Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proc. of the 6th Annual Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 181–190, 2015.

13 Michael B. Cohen, Jelani Nelson, and David P. Woodruff. Optimal approximate matrix product in terms of stable rank. *CoRR*, abs/1507.02268, 2015.

14 Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse Johnson-Lindenstrauss transform. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, 2010.

15 Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices I: approximating matrix multiplication. *SIAM J. Comput.*, 36(1):132–157, 2006.

16 Petros Drineas, Malik Magdon-Ismail, Michael W. Mahoney, and David P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13:3475–3506, 2012.

17 Petros Drineas, Michael W. Mahoney, and S. Muthukrishnan. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1127–1136, 2006.

18 Alex Gittens and Michael W. Mahoney. Revisiting the nystrom method for improved large-scale machine learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 567–575, 2013.

19 Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

20 Ishay Haviv and Oded Regev. The restricted isometry property of subsampled Fourier matrices. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), to appear*, 2016.

21 William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

22 Daniel M. Kane and Jelani Nelson. Sparser Johnson-Lindenstrauss transforms. *J. ACM*, 61(1):4, 2014.

23 Alexandra Kolla, Yury Makarychev, Amin Saberi, and Shang-Hua Teng. Subgraph sparsification and nearly optimal ultrasparsifiers. In *Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC)*, pages 57–66, 2010.

24 Felix Krahmer and Rachel Ward. New and improved Johnson-Lindenstrauss embeddings via the Restricted Isometry Property. *SIAM J. Math. Anal.*, 43(3):1269–1281, 2011.

25 Anastasios T. Kyrillidis, Michail Vlachos, and Anastasios Zouzias. Approximate matrix multiplication with application to linear embeddings. *CoRR*, abs/1403.7683, 2014.

26 Yin Tat Lee and He Sun. Constructing linear sized spectral sparsification in almost linear time. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 250–269, 2015.

27 Mu Li, Gary L. Miller, and Richard Peng. Iterative row sampling. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2013.

28 Yingyu Liang, Maria-Florina Balcan, Vandana Kanchanapally, and David P. Woodruff. Improved distributed principal component analysis. In *Proceedings of the 27th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, 2014.

29 Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172, 2007.

30 Yichao Lu, Paramveer Dhillon, Dean Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized Hadamard transform. In *Proceedings of the 26th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, 2013.

31 Avner Magen and Anastasios Zouzias. Low rank matrix-valued Chernoff bounds and approximate matrix multiplication. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1422–1436, 2011.

32 Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011.

33 Xiangrui Meng and Michael W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the 45th ACM Symposium on Theory of Computing (STOC)*, pages 91–100, 2013.

34 Jelani Nelson and Huy L. Nguyễn. OSNAP: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 117–126, 2013.

35 Jelani Nelson and Huy L. Nguyễn. Lower bounds for oblivious subspace embeddings. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 883–894, 2014.

36 Jelani Nelson, Eric Price, and Mary Wootters. New constructions of RIP matrices with fast multiplication and fewer rows. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

37 Nima Reyhani, Hideitsu Hino, and Ricardo Vigário. New probabilistic bounds on eigenvalues and eigenvectors of random kernel matrices. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 627–634, 2011.

38 Tamás Sarlós. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 143–152, 2006.

39 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, 2011.

40 Mikkel Thorup and Yin Zhang. Tabulation-based 5-independent hashing with applications to linear probing and second moment estimation. *SIAM J. Comput.*, 41(2):293–331, 2012.

41 Joel A. Tropp. Improved analysis of the subsampled randomized Hadamard transform. *Adv. Adapt. Data Anal.*, 3(1–2):115–126, 2011.

42 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

43 Yun Yang, Mert Pilanci, and Martin J. Wainwright. Randomized sketches for kernels: Fast and optimal non-parametric regression. *CoRR*, abs/1501.06195, 2015.

Approximate Span Programs*

Tsuyoshi Ito[1] and Stacey Jeffery[2]

1 **NEC Labs, Princeton, NJ, USA**
2 **Institute for Quantum Information and Matter, California Institute of Technology, Pasadena, CA, USA**
sjeffery@caltech.edu

Abstract

Span programs are a model of computation that have been used to design quantum algorithms, mainly in the query model. It is known that for any decision problem, there exists a span program that leads to an algorithm with optimal quantum query complexity, however finding such an algorithm is generally challenging. In this work, we consider new ways of designing quantum algorithms using span programs. We show how any span program that decides a problem f can also be used to decide "property testing" versions of the function f, or more generally, approximate a quantity called the *span program witness size*, which is some property of the input related to f. For example, using our techniques, the span program for OR, which can be used to design an optimal algorithm for the OR function, can also be used to design optimal algorithms for: threshold functions, in which we want to decide if the Hamming weight of a string is above a threshold, or far below, given the promise that one of these is true; and approximate counting, in which we want to estimate the Hamming weight of the input up to some desired accuracy. We achieve these results by relaxing the requirement that 1-inputs hit some *target* exactly in the span program, which could potentially make design of span programs significantly easier. In addition, we give an exposition of span program structure, which increases the general understanding of this important model. One implication of this is alternative algorithms for estimating the witness size when the phase gap of a certain unitary can be lower bounded. We show how to lower bound this phase gap in certain cases.

As an application, we give the first upper bounds in the adjacency query model on the quantum time complexity of estimating the effective resistance between s and t, $R_{s,t}(G)$. For this problem we obtain $\widetilde{O}(\frac{1}{\varepsilon^{3/2}} n\sqrt{R_{s,t}(G)})$, using $O(\log n)$ space. In addition, when μ is a lower bound on $\lambda_2(G)$, by our phase gap lower bound, we can obtain an upper bound of $\widetilde{O}\left(\frac{1}{\varepsilon} n\sqrt{R_{s,t}(G)/\mu}\right)$ for estimating effective resistance, also using $O(\log n)$ space.

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

Keywords and phrases Quantum algorithms, span programs, quantum query complexity, effective resistance

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.12

1 Introduction

Span programs are a model of computation first used to study logspace complexity [13], and more recently, introduced to the study of quantum algorithms in [20]. They are of immense theoretical importance, having been used to show that the general adversary bound gives a

* Full version: https://arxiv.org/abs/1507.00432 [quant-ph]. This work supported by the Institute for Quantum Information and Matter, an NSF Physics Frontiers Center (NFS Grant PHY-1125565) with support of the Gordon and Betty Moore Foundation (GBMF-12500028).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 12; pp. 12:1–12:14

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

tight lower bound on the quantum query complexity of any decision problem [18, 19]. As a means of designing quantum algorithms, it is known that for any decision problem, there exists a span-program-based algorithm with asymptotically optimal quantum query complexity, but this fact alone gives no indication of how to find such an algorithm. Despite the relative difficulty in designing quantum algorithms this way, there are many applications, including formula evaluation [20, 19], a number of algorithms based on the learning graph framework [3], st-connectivity [5] and k-distinctness [2]. Although generally quantum algorithms designed via span programs can only be analyzed in terms of their query complexity, in some cases their time complexity can also be analyzed, as is the case with the quantum algorithm for st-connectivity. In the case of the quantum algorithm for k-distinctness, the ideas used in designing the span program could be turned into a quantum algorithm for 3-distinctness with time complexity matching its query complexity up to logarithmic factors [4].

In this work, we consider new ways of designing quantum algorithms via span programs. Consider Grover's quantum search algorithm, which, on input $x \in \{0,1\}^n$, decides if there is some $i \in [n]$ such that $x_i = 1$ using only $O(\sqrt{n})$ quantum operations [10]. The ideas behind this algorithm have been used in innumerable contexts, but in particular, a careful analysis of the ideas behind Grover's algorithm led to algorithms for similar problems, including a class of *threshold functions*: given $x \in \{0,1\}^n$, decide if $|x| \geq t$ or $|x| < \varepsilon t$, where $|x|$ denotes the Hamming weight; and approximate counting: given $x \in \{0,1\}^n$, output an estimate of $|x|$ to some desired accuracy. The results in this paper offer the possibility of obtaining analogous results for any span program. That is, given a span program for some problem f, our results show that one can obtain, not only an algorithm for f, but algorithms for a related class of threshold functions, as well as an algorithm for estimating a quantity called the *span program witness size*, which is analogous to $|x|$ in the above example (and is in fact exactly $1/|x|$ in the span program for the OR function — see Section 2.3).

We give several new means of constructing quantum algorithms from span programs. Roughly speaking, a span program can be turned into a quantum algorithm that decides between two types of inputs: those that "hit" a certain "target vector", and those that don't. We show how to turn a span program into an algorithm that decides between inputs that get "close to" the target vector, and those that don't. Whereas traditionally a span program has been associated with some decision problem, this allows us to now associate, with one span program, a whole class of threshold problems.

In addition, for any span program P, we can construct a quantum algorithm that estimates the *positive witness size*, $w_+(x)$, to accuracy ε in $\frac{1}{\varepsilon^{3/2}}\sqrt{w_+(x)\widetilde{W}_-}$ queries, where $\widetilde{W}_-$ is the *approximate negative witness complexity* of P. This construction is useful whenever we can construct a span program for which $w_+(x)$ corresponds to some function we care to estimate, as is the case with the span program for OR, in which $w_+(x) = \frac{1}{|x|}$, or the span from for st-connectivity, in which $w_+(G) = \frac{1}{2}R_{s,t}(G)$, where G is a graph, and $R_{s,t}(G)$ is the *effective resistance* between s and t in G. We show similar results for estimating the negative witness size as well.

Our analysis of the structure of span programs increases the theoretical understanding of this important model. One implication of this is alternative algorithms for estimating the witness size when the phase gap (or spectral gap) of a certain unitary associated with the span program can be lower bounded. This is in contrast to previous span program algorithms, including those mentioned in the previous paragraph, which have all relied on *effective spectral gap* analysis. We show how the phase gap can be lower bounded by $\frac{\sigma_{\max}(A)}{\sigma_{\min}(A(x))}$, where A and $A(x)$ are linear operators associated with the span program and some input x, and $\sigma_{\min}$ and $\sigma_{\max}$ are the smallest and largest nonzero singular values.

In addition, our exposition highlights the relationship between span programs and estimating the size of the smallest solution to a linear system, which is a problem solved by Harrow, Hassidim and Lloyd in [11]. It is not yet clear if this relationship can lead to new algorithms, but it is an interesting direction for future work.

An immediate application of our results is a quantum algorithm for estimating the effective resistance between two vertices in a graph, $R_{s,t}(G)$. This example is immediate, because in [5], a span program for st-connectivity was presented, in which the positive witness size corresponds to $R_{s,t}(G)$. The results of [5], combined with our new span program algorithms, immediately yield an upper bound of $\widetilde{O}(\frac{1}{\varepsilon^{3/2}} n\sqrt{R_{s,t}(G)})$ for estimating the effective resistance to relative accuracy ε. This upper bound also holds for time complexity, due to the time complexity analysis of [5]. Using our new spectral analysis techniques, we are also able to get an often better upper bound of $\widetilde{O}\left(\frac{1}{\varepsilon} n\sqrt{R_{s,t}(G)/\mu}\right)$, on the time complexity of estimating effective resistance, where μ is a lower bound on $\lambda_2(G)$, the second smallest eigenvalue of the Laplacian. Both algorithms use $O(\log n)$ space. We also show that a linear dependence on n is necessary, so our results cannot be significantly improved.

These are the first quantum algorithms for this problem in the adjacency query model. Previous quantum algorithms have been in the edge-list model for d-regular graphs [22]. These results can be naively extended to the adjacency query model by simulating an edge query with $\sqrt{n/d}$ adjacency queries, using quantum search, which gives an upper bound of $\widetilde{O}\left(\frac{d^{3/2}}{\Phi^2\varepsilon}\sqrt{n/d}\right)$ queries, where Φ is the *conductance* of the input graph. Our upper bounds improve on this in many cases, including, but not limited to, d-regular graphs with $d > \sqrt[4]{n}$, and furthermore, our results do not assume the input graph is regular. Classically, the effective resistance can be computed exactly by inverting the Laplacian, which costs $O(m) = O(n^2)$, where m is the number of edges in the input graph.

1.1 Preliminaries

To begin, we fix notation. For vector spaces V and W, we let $\mathcal{L}(V, W)$ denote the set of linear operators from V to W. For any operator $A \in \mathcal{L}(V, W)$, we denote by $\mathrm{col}A$ the columnspace, $\mathrm{row}A$ the rowspace, and $\ker A$ the kernel of A. $\sigma_{\min}(A)$ and $\sigma_{\max}(A)$ denote the smallest and largest non-zero singular values, respectively. A^+ denotes the Moore-Penrose pseudo-inverse.

The algorithms in this paper solve either decision problems, or estimation problems. For $f : X \subseteq [q]^n \to \{0, 1\}$, we say that an algorithm *decides f with bounded error* if for any $x \in X$, with probability at least $2/3$, the algorithm outputs $f(x)$ on input x. For $f : X \subseteq [q]^n \to \mathbb{R}_{\geq 0}$, we say that an algorithm *estimates f to relative accuracy ε* with bounded error if for any $x \in X$, with probability at least $2/3$, on input x the algorithm outputs $\tilde{f}$ such that $|f(x) - \tilde{f}| \leq \varepsilon f(x)$. In both cases, using $2/3$ is without loss of generality: any algorithm with success probability bounded above $1/2$ by a constant can be amplified to success probability arbitrarily close to 1 by taking the median of the outputs of a constant number of repetitions of the algorithm. We generally omit the description "with bounded error", as all of our algorithms have bounded error.

All algorithms presented in this paper are based on the following structure. We have some initial state $|\phi_0\rangle$, and some unitary operator U, and we want to estimate $\|\Pi_0|\phi_0\rangle\|$, where Π_0 is the orthogonal projector onto the 1-eigenspace of U. The first step in this process is a quantum algorithm that estimates, in a new register, the phase of U applied to the input state.

▶ **Theorem 1** (Phase Estimation [14, 9]). *Let $U = \sum_{j=1}^{m} e^{i\theta_j}|\psi_j\rangle\langle\psi_j|$ be the spectral decomposition of a unitary, with $\theta_1, \ldots, \theta_m \in (-\pi, \pi]$. For any $\Theta \in (0, \pi)$ and $\varepsilon \in (0, 1)$, there*

exists a quantum algorithm that makes $O\left(\frac{1}{\Theta}\log\frac{1}{\varepsilon}\right)$ *controlled calls to* U *and, on input* $|\psi_j\rangle$*, outputs a state* $|\psi_j\rangle|\omega\rangle$ *such that if* $\theta_j = 0$*, then* $|\omega\rangle = |0\rangle$*, and if* $|\theta_j| \geq \Theta$*,* $|\langle 0|\omega\rangle|^2 \leq \varepsilon$*. If* U *acts on* s *qubits, the algorithm uses* $O(s + \log\frac{1}{\Theta})$ *space.*

The precision needed to isolate $\Pi_0|\phi_0\rangle$ depends on the smallest nonzero phase of U, the *phase gap*.

▶ **Definition 2** (Phase Gap). Let $\{e^{i\theta_j}\}_{j\in S}$ be the eigenvalues of a unitary operator U, with $\{\theta_j\}_{j\in S} \subset (-\pi, \pi]$. Then the *phase gap* of U is $\Delta(U) := \min\{|\theta_j| : \theta_j \neq 0\}$.

In order to estimate $\|\Pi_0|\phi_0\rangle\|^2$, given a state $|0\rangle\Pi_0|\phi_0\rangle + |1\rangle(I-\Pi_0)|\phi_0\rangle$, we use the following.

▶ **Theorem 3** (Amplitude Estimation [7]). *Let* $\mathcal{A}$ *be a quantum algorithm that outputs* $\sqrt{p(x)}|0\rangle|\Psi_x(0)\rangle + \sqrt{1-p(x)}|1\rangle|\Psi_x(1)\rangle$ *on input* x*. Then there exists a quantum algorithm that estimates* $p(x)$ *to precision* ε *using* $O\left(\frac{1}{\varepsilon}\frac{1}{\sqrt{p(x)}}\right)$ *calls to* $\mathcal{A}$*.*

If we know the amplitude is either $\leq p_0$ or $\geq p_1$ for some $p_0 < p_1$, then we can use amplitude estimation to distinguish between these two cases.

▶ **Corollary 4** (Amplitude Gap). *Let* $\mathcal{A}$ *be a quantum algorithm that, on input* x*, outputs* $\sqrt{p(x)}|0\rangle|\Psi_x(0)\rangle + \sqrt{1-p(x)}|1\rangle|\Psi_x(1)\rangle$*. For any* $0 \leq p_0 < p_1 \leq 1$*, we can distinguish between the cases* $p(x) \geq p_1$ *and* $p(x) \leq p_0$ *with bounded error using* $O\left(\frac{\sqrt{p_1}}{p_1-p_0}\right)$ *calls to* $\mathcal{A}$*.*

In order to make use of phase estimation, we will need to analyze the spectrum of a particular unitary, which, in our case, consists of a pair of reflections. The following lemma first appeared in this form in [15]:

▶ **Lemma 5** (Effective Spectral Gap Lemma). *Let* $U = (2\Pi_A - I)(2\Pi_B - I)$ *be the product of two reflections, and let* Π_Θ *be the orthogonal projector onto* $\mathrm{span}\{|u\rangle : U|u\rangle = e^{i\theta}|u\rangle, |\theta| \leq \Theta\}$*. Then if* $\Pi_A|u\rangle = 0$*,* $\|\Pi_\Theta\Pi_B|u\rangle\| \leq \frac{\Theta}{2}\||u\rangle\|$*.*

The following theorem was first used in the context of quantum algorithms by Szegedy [21]:

▶ **Theorem 6** ([21]). *Let* $U = (2\Pi_A - I)(2\Pi_B - I)$ *be a unitary on a space* H *containing* $A = \mathrm{span}\{|\psi_1\rangle, \ldots, |\psi_a\rangle\}$ *and* $B = \mathrm{span}\{|\phi_1\rangle, \ldots, |\phi_b\rangle\}$*. Let* $\Pi_A = \sum_{i=1}^a |\psi_i\rangle\langle\psi_i|$ *and* $\Pi_B = \sum_{i=1}^b |\phi_i\rangle\langle\phi_i|$ *be the orthogonal projectors onto these spaces. Let* $D = \Pi_A\Pi_B$ *be the discriminant of* U*, and suppose it has singular value decomposition* $\sum_{j=1}^r \cos\theta_j|\alpha_j\rangle\langle\beta_j|$*, with* $\theta_j \in [0, \frac{\pi}{2}]$*. Then the spectrum of* U *is* $\{e^{\pm 2i\theta_j}\}_j$*. The 1-eigenspace of* U *is* $(A\cap B)\oplus(A^\perp\cap B^\perp)$ *and the* (-1)*-eigenspace is* $(A \cap B^\perp) \oplus (A^\perp \cap B)$*.*

Let $\Lambda_A = \sum_{j=1}^a |\psi_j\rangle\langle j|$ and $\Lambda_B = \sum_{j=1}^b |\phi_j\rangle\langle j|$. We note that in the original statement of Theorem 6, the discriminant is defined $D' = \Lambda_A^\dagger\Lambda_B$. However it is easy to see that D' and D have the same singular values: if $D' = \sum_i \sigma_i|v_i\rangle\langle u_i|$ is a singular value decomposition of D', then $D = \sum_i \sigma_i\Lambda_A|v_i\rangle\langle u_i|\Lambda_B^\dagger$ is a singular value decomposition of D, since Λ_A acts as an isometry on the columns of D', and Λ_B acts as an isometry on the rows of D'.
The following corollary to Theorem 6 will be useful in the analysis of several algorithms.

▶ **Corollary 7** (Phase Gap and Discriminant). *Let* D *be the discriminant of a unitary* $U = (2\Pi_A - I)(2\Pi_B - I)$*. Then* $\Delta(-U) \geq 2\sigma_{\min}(D)$*.*

2 Approximate Span Programs

2.1 Span Programs and Decision Problems

In this section, we review the concept of span programs, and their use in quantum algorithms.

▶ **Definition 8** (Span Program). A span program $P = (H, V, \tau, A)$ on $[q]^n$ consists of

1. finite-dimensional inner product spaces $H = H_1 \oplus \cdots \oplus H_n \oplus H_{\text{true}} \oplus H_{\text{false}}$, and $\{H_{j,a} \subseteq H_j\}_{j\in[n],a\in[q]}$ such that $H_{j,1} + \cdots + H_{j,q} = H_j$,
2. a vector space V,
3. a *target vector* $\tau \in V$, and
4. a linear operator $A \in \mathcal{L}(H, V)$.

To each string $x \in [q]^n$, we associate a subspace $H(x) := H_{1,x_1} \oplus \cdots \oplus H_{n,x_n} \oplus H_{\text{true}}$.

Although our notation in Definition 8 deviates from previous span program definitions, the only difference in the substance of the definition is that the spaces $H_{j,a}$ and $H_{j,b}$ for $a \neq b$ need not be orthogonal in our definition. This has the effect of removing $\log q$ factors in the equivalence between span programs and the dual adversary bound (for details see [12, Sec. 7.1]). The spaces H_{true} and H_{false} can be useful for designing a span program, but are never required, since we can always add an $(n+1)^{\text{th}}$ variable, set $x_{n+1} = 1$, and let $H_{n+1,0} = H_{\text{false}}$ and $H_{n+1,1} = H_{\text{true}}$.

A span program on $[q]^n$ partitions $[q]^n$ into two sets: *positive* inputs, which we call P_1, and *negative* inputs, which we call P_0. The importance of this partition stems from the fact that a span program may be converted into a quantum algorithm for deciding this partition in the quantum query model [18, 19]. Thus, if one can construct a span program whose partition of $[q]^n$ corresponds to a problem one wants to solve, an algorithm follows. In order to describe how a span program partitions $[q]^n$ and the query complexity of the resulting algorithm, we need the concept of positive and negative witnesses and witness size.

▶ **Definition 9** (Positive and Negative Witness). Fix a span program P on $[q]^n$, and a string $x \in [q]^n$. We say that $|w\rangle$ is a *positive witness for x in P* if $|w\rangle \in H(x)$, and $A|w\rangle = \tau$. We define the *positive witness size of x* as:

$$w_+(x, P) = w_+(x) = \min\{\||w\rangle\|^2 : |w\rangle \in H(x), A|w\rangle = \tau\},$$

if there exists a positive witness for x, and $w_+(x) = \infty$ else. We say that $\omega \in \mathcal{L}(V, \mathbb{R})$ is a *negative witness for x in P* if $\omega A \Pi_{H(x)} = 0$ and $\omega\tau = 1$. We define the *negative witness size of x* as:

$$w_-(x, P) = w_-(x) = \min\{\|\omega A\|^2 : \omega \in \mathcal{L}(V, \mathbb{R}), \omega A \Pi_{H(x)} = 0, \omega\tau = 1\},$$

if there exists a negative witness, and $w_-(x) = \infty$ otherwise. If $w_+(x)$ is finite, we say that x is *positive* (with respect to P), and if $w_-(x)$ is finite, we say that x is *negative*. We let P_1 denote the set of positive inputs, and P_0 the set of negative inputs for P. Note that for every $x \in [q]^n$, exactly one of $w_-(x)$ and $w_+(x)$ is finite; that is, (P_0, P_1) partitions $[q]^n$.

For a decision problem $f : X \subseteq [q]^n \to \{0, 1\}$, we say that P *decides* f if $f^{-1}(0) \subseteq P_0$ and $f^{-1}(1) \subseteq P_1$. In that case, we can use P to construct a quantum algorithm that decides f.

▶ **Theorem 10** ([18]). *Fix $f : X \subseteq [q]^n \to \{0, 1\}$, and let P be a span program on $[q]^n$ that decides f. Let $W_+(f, P) = \max_{x\in f^{-1}(1)} w_+(x, P)$ and $W_-(f, P) = \max_{x\in f^{-1}(0)} w_-(x, P)$. Then there exists a quantum algorithm that decides f using $O(\sqrt{W_+(f, P)W_-(f, P)})$ queries.*

We call $\sqrt{W_+(f,P)W_-(f,P)}$ the *complexity* of P. It is known that for any decision problem, there exists a span program whose complexity is equal, up to constants, to its query complexity [18, 19] ([12, Sec. 7.1] removes log factors in this statement), however, it is generally a difficult task to find such an optimal span program.

2.2 Span Programs and Approximate Decision Problems

Consider a span program P and $x \in P_0$. Suppose there is some $|w\rangle \in H(x)$ such that $A|w\rangle$ comes extremely close to τ. We might say that x is very close to being in P_1. If all vectors in $H(y)$ for $y \in P_0 \setminus \{x\}$ are very far from τ, it might be slightly more natural to consider the partition $(P_0 \setminus \{x\}, P_1 \cup \{x\})$ rather than (P_0, P_1).

As further motivation, we mention a construction of Reichardt [18, Sec. 3 of full version] that takes any quantum query algorithm with one-sided error, and converts it into a span program whose complexity matches the query complexity of the algorithm. The target of the span program is the vector $|1,\bar{0}\rangle$, which corresponds to a quantum state with a 1 in the answer register and 0s elsewhere. If an algorithm has no error on 1-inputs, it can be modified so that it always ends in exactly this state, by uncomputing all but the answer register. An algorithm with two-sided error cannot be turned into a span program using this construction, because there is error in the final state. This is intuitively in opposition to the evidence that span programs characterize bounded (two-sided) error quantum query complexity. The exactness required by span programs seems to contrast the spirit of non-exact quantum algorithms.

This motivates us to consider the *positive error* of an input, or how close it comes to being positive. Since there is no meaningful notion of distance in V, we consider closeness in H.

▶ **Definition 11** (Positive Error)**.** For any span program P on $[q]^n$, and $x \in [q]^n$, we define the *positive error of x in P* as:

$$e_+(x) = e_+(x,P) := \min\left\{\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 : A|w\rangle = \tau\right\}.$$

Note that $e_+(x,P) = 0$ if and only if $x \in P_1$. Any $|w\rangle$ such that $\left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 = e_+(x)$ is called a *min-error positive witness for x in P*. We define

$$\tilde{w}_+(x) = \tilde{w}_+(x,P) := \min\left\{\||w\rangle\|^2 : A|w\rangle = \tau, \left\|\Pi_{H(x)^\perp}|w\rangle\right\|^2 = e_+(x)\right\}.$$

A min-error positive witness that also minimizes $\||w\rangle\|^2$ is called an *optimal min-error positive witness for x*.

Note that if $x \in P_1$, then $e_+(x) = 0$. In that case, a min-error positive witness for x is just a positive witness, and $\tilde{w}_+(x) = w_+(x)$.

We can define a similar notion for positive inputs, to measure their closeness to being negative.

▶ **Definition 12** (Negative Error)**.** For any span program P on $[q]^n$ and $x \in [q]^n$, we define the *negative error of x in P* as:

$$e_-(x) = e_-(x,P) := \min\left\{\left\|\omega A\Pi_{H(x)}\right\|^2 : \omega(\tau) = 1\right\}.$$

Again, $e_-(x,P) = 0$ if and only if $x \in P_0$. Any ω such that $\left\|\omega A\Pi_{H(x)}\right\|^2 = e_-(x,P)$ is called a *min-error negative witness for x in P*. We define

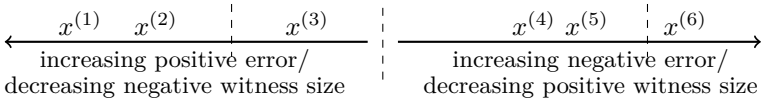
$$\tilde{w}_-(x) = \tilde{w}_-(x,P) := \min\left\{\|\omega A\|^2 : \omega(\tau) = 1, \left\|\omega A\Pi_{H(x)}\right\|^2 = e_-(x,P)\right\}.$$

A min-error negative witness that also minimizes $\|\omega A\|^2$ is called an *optimal min-error negative witness for* x.

It turns out that the notion of span program error has a very nice characterization as exactly the reciprocal of the witness size. We prove in the full version (Theorems 2.10 and 2.11):

$$\forall x \in P_0,\ w_-(x) = \frac{1}{e_+(x)}, \qquad \text{and} \qquad \forall x \in P_1,\ w_+(x) = \frac{1}{e_-(x)}.$$

This is a very nice state of affairs, for a number of reasons. It allows us two ways of thinking about approximate span programs: in terms of how small the error is, or how large the witness size is. That is, we can say that an input $x \in P_0$ is *almost positive* either because its positive error is small, or equivalently, because its negative witness size is large. In general, we can think of P as not only partitioning P into (P_0, P_1), but inducing an ordering on $[q]^n$ from most negative — smallest negative witness, or equivalently, largest positive error — to most positive — smallest positive witness, or equivalently, largest negative error. For example, on the domain $\{x^{(1)}, \ldots, x^{(6)}\} \subset [q]^n$, P might induce the following ordering:



The inputs $\{x^{(1)}, x^{(2)}, x^{(3)}\}$ are in P_0, and $w_-(x^{(1)}) < w_-(x^{(2)}) < w_-(x^{(3)})$ (although it is generally possible for two inputs to have the same witness size). The inputs $\{x^{(4)}, x^{(5)}, x^{(6)}\}$ are in P_1, and $w_+(x^{(4)}) > w_+(x^{(5)}) > w_+(x^{(6)})$. The span program exactly decides the partition $(\{x^{(1)}, x^{(2)}, x^{(3)}\}, \{x^{(4)}, x^{(5)}, x^{(6)}\})$, but we say it *approximates* any partition that respects the ordering. If we obtain a partition by drawing a line somewhere on the left side, for example $(\{x^{(1)}, x^{(2)}\}, \{x^{(3)}, x^{(4)}, x^{(5)}, x^{(6)}\})$, we say P *negatively* approximates the function corresponding to that partition, whereas if we obtain a partition by drawing a line on the right side, for example $(\{x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}, x^{(5)}\}, \{x^{(6)}\})$, we say P *positively* approximates the function.

▶ **Definition 13** (Functions Approximately Associated with P)**.** Let P be a span program on $[q]^n$, and $f : X \subseteq [q]^n \to \{0,1\}$ a decision problem. For any $\lambda \in (0,1)$, we say that P *positively* λ*-approximates* f if $f^{-1}(1) \subseteq P_1$, and for all $x \in f^{-1}(0)$, either $x \in P_0$, or $w_+(x,P) \geq \frac{1}{\lambda} W_+(f,P)$, where $W_+(f,P) := \max_{x \in f^{-1}(1)} w_+(x,P)$. We say that P *negatively* λ*-approximates* f if $f^{-1}(0) \subseteq P_0$, and for all $x \in f^{-1}(1)$, either $x \in P_1$, or $w_-(x,P) \geq \frac{1}{\lambda} W_-(f,P)$, where $W_-(f,P) := \max_{x \in f^{-1}(0)} w_-(x,P)$. If P decides f exactly, then both conditions hold for any value of λ, and so we can say that P 0-approximates f.

This allows us to consider a much broader class of functions associated with a particular span program. This association is useful, because as with the standard notion of association between a function f and a span program, if a function is approximated by a span program, we can convert the span program into a quantum algorithm that decides f using a number of queries related to the witness sizes. Specifically, we get the following theorem.

▶ **Theorem 14** (Approximate Span Program Decision Algorithms)**.** *Fix* $f : X \subseteq [q]^n \to \{0,1\}$*, and let* P *be a span program that positively* λ*-approximates* f*. Define*

$$W_+ = W_+(f,P) := \max_{x \in f^{-1}(1)} w_+(x,P) \qquad \text{and} \qquad \widetilde{W}_- = \widetilde{W}_-(f,P) := \max_{x \in f^{-1}(0)} \tilde{w}_-(x,P).$$

There is a quantum algorithm that decides f with bounded error in $O\left(\frac{\sqrt{W_+\widetilde{W}_-}}{(1-\lambda)^{3/2}}\log\frac{1}{1-\lambda}\right)$ queries. Similarly, let P be a span program that negatively λ-approximates f. Define

$$W_- = W_-(f,P) := \max_{x\in f^{-1}(0)} w_-(x,P) \qquad \text{and} \qquad \widetilde{W}_+ = \widetilde{W}_+(f,P) := \max_{x\in f^{-1}(1)} \tilde{w}_+(x,P).$$

There is a quantum algorithm that decides f with bounded error in $O\left(\frac{\sqrt{W_-\widetilde{W}_+}}{(1-\lambda)^{3/2}}\log\frac{1}{1-\lambda}\right)$ queries.

With the ability to distinguish between different witness sizes, we can obtain algorithms for estimating the witness size.

▶ **Theorem 15** (Witness Size Estimation Algorithm). *Fix $f : X \subseteq [q]^n \to \mathbb{R}_{\geq 0}$. Let P be a span program such that for all $x \in X$, $f(x) = w_+(x,P)$ and define $\widetilde{W}_- = \widetilde{W}_-(f,P) = \max_{x\in X} \tilde{w}_-(x,P)$. There exists a quantum algorithm that estimates f to accuracy ε in $\widetilde{O}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{w_+(x)\widetilde{W}_-}\right)$ queries. Similarly, let P be a span program such that for all $x \in X$, $f(x) = w_-(x,P)$ and define $\widetilde{W}_+ = \widetilde{W}_+(f,P) = \max_{x\in X} \tilde{w}_+(x,P)$. Then there exists a quantum algorithm that estimates f to accuracy ε in $\widetilde{O}\left(\frac{1}{\varepsilon^{3/2}}\sqrt{w_-(x)\widetilde{W}_+}\right)$ queries.*

Proofs of Theorem 14 and 15 can be found in the full version (Theorems 2.7 and 2.8), but we give a high-level outline here. As in the case of algorithms previously constructed from span programs, our algorithms will consist of phase estimation of a unitary on H, applied to some initial state. Unlike previous applications, we will use $|w_0\rangle = A^+\tau$ (discussed more in full version, Section 2.4), as the initial state. This state is independent of the input, and so can be generated with 0 queries. For *negative span program algorithms*, where we want to decide a function negatively approximated by P, we will use a unitary $U(P,x)$, defined as follows:

$$U(P,x) := (2\Pi_{\ker A} - I)(2\Pi_{H(x)} - I).$$

This is similar to the unitary used in previous span program algorithms. Note that $(2\Pi_{\ker A} - I)$ is input-independent, and so can be implemented in 0 queries. However, in order to analyze the time complexity of a span program algorithm, this reflection must be implemented (as we are able to do for our applications, following [5]). The reflection $(2\Pi_{H(x)} - I)$ depends on the input, but requires only two queries to implement.

For *positive span program algorithms*, where we want to decide a function positively approximated by P, or estimate the positive witness size, we will use a slightly different unitary, $U'(P,x)$.

In order to show how these unitaries can be used to distinguish between inputs with small negative (resp. positive) witnesses, and those that only have large negative (resp. positive) witnesses, we analyze the overlap of the initial state, $|w_0\rangle$ with the 1-eigenspace of $U(P,x)$ (resp. $U'(P,x)$) in terms of the witness size. Specifically, we show that the overlap of $|w_0\rangle$ with the 1-eigenspace of $U(P,x)$ is exactly $\frac{1}{w_-(x,P)}$ (full version, Lemma 3.3), and the overlap of $|w_0\rangle$ with the 1-eigenspace of $U'(P,x)$ is exactly $\frac{1}{w_+(x,P)}$ (full version, Lemma 3.5). We can then use phase estimation, followed by amplitude estimation, to estimate the witness size.

There are then two possibilities for bounding the required precision of phase estimation, which also tells us the number of times we will need to call $U(P,x)$ (resp. $U'(P,x)$), and

therefore, the query complexity of the algorithm. Similar to previous span program algorithms we use the effective spectral gap lemma to show that the overlap of $|w_0\rangle$ with $e^{i\theta}$-eigenspaces of $U(P,x)$ (resp. $U'(P,x)$) is not too large for small θ (full version, Lemmas 3.2 and 3.4). This leads to Theorem 14 and Theorem 15.

The second way to bound the required precision of phase estimation is to lower bound the phase gap of $U(P,x)$ (resp. $U'(P,x)$), which may be very difficult in general. However, by relating the phase gap of $U(P,x)$ (resp. $U'(P,x)$) to the spectrum of A and $A(x) = A\Pi_{H(x)}$ in a novel way, we show how to lower bound the phase gap in some cases, which may give better results. This leads to the following theorem.

▶ **Theorem 16** (Witness Size Estimation Algorithm Using Real Phase Gap). *We say that a span program is* normalized *if* $\|A^+\tau\| = 1$. *Any span program can be normalized by scaling* τ.

Fix $f : X \subseteq [q]^n \to \mathbb{R}_{\geq 0}$ *and let* $P = (H, V, \tau, A)$ *be a normalized span program on* $[q]^n$ *such that for all* $x \in X$, $f(x) = w_+(x,P)$ *(resp.* $f(x) = w_-(x)$*). If* $\kappa \geq \frac{\sigma_{\max}(A)}{\sigma_{\min}(A\Pi_{H(x)})}$ *for all* $x \in X$, *then the quantum query complexity of estimating* $f(x)$ *to relative accuracy* ε *is at most* $\widetilde{O}\left(\sqrt{f(x)}\kappa/\varepsilon\right)$.

In particular, in our application to effective resistance, it is not difficult to bound the phase gap in this way, which leads to an improved upper bound.

2.3 Example

To illustrate how these ideas might be useful, we will give a brief example of how a span program that leads to an algorithm for the OR function can be combined with our results to additionally give algorithms for threshold functions and approximate counting. We define a span program P on $\{0,1\}^n$ as follows:

$$V = \mathbb{R}, \qquad \tau = 1, \qquad H_i = H_{i,1} = \text{span}\{|i\rangle\}, \qquad H_{i,0} = \{0\}, \qquad A = \sum_{i=1}^{n} \langle i|.$$

So we have $H = \text{span}\{|i\rangle : i \in [n]\}$ and $H(x) = \text{span}\{|i\rangle : x_i = 1\}$. It's not difficult to see that P decides OR. In particular, we can see that the optimal positive witness for any x such that $|x| > 0$ is $|w_x\rangle = \sum_{i:x_i=1} \frac{1}{|x|}|i\rangle$. The only linear function $\omega : \mathbb{R} \to \mathbb{R}$ that maps τ to 1 is the identity, and indeed, this is a negative witness for the string $\bar{0} = 0 \ldots 0$, since $H(\bar{0}) = \{0\}$, and so $\omega A\Pi_{H(\bar{0})} = 0$.

Let $\lambda \in (0,1)$, $t \in [n]$, and let f be a threshold function defined by $f(x) = 1$ if $|x| \geq t$ and $f(x) = 0$ if $|x| \leq \lambda t$, with the promise that one of these conditions holds. Note that if $f(x) = 1$, then $w_+(x) = \||w_x\rangle\|^2 = \frac{1}{|x|} \leq \frac{1}{t}$, so $W_+(f,P) = \frac{1}{t}$. On the other hand, if $f(x) = 0$, then $w_+(x) = \frac{1}{|x|} \geq \frac{1}{\lambda t} = \frac{1}{\lambda} W_+(f,P)$, so P positively λ-approximates f. The only approximate negative witness is ω the identity, so we have $\widetilde{W}_- = \|\omega A\|^2 = \|A\|^2 = n$. By Theorem 14, there is a quantum algorithm for f with query complexity $\frac{1}{(1-\lambda)^{3/2}}\sqrt{W_+\widetilde{W}_-} = \frac{1}{(1-\lambda)^{3/2}}\sqrt{n/t}$.

Furthermore, since $w_+(x) = \frac{1}{|x|}$, by Theorem 15, we can estimate $\frac{1}{|x|}$ to relative accuracy ε, and therefore we can estimate $|x|$ to relative accuracy 2ε, in quantum query complexity $\frac{1}{\varepsilon^{3/2}}\sqrt{n/|x|}$.

These upper bounds do not have optimal scaling in ε, as the actual quantum query complexities of these problems are $\frac{1}{1-\lambda}\sqrt{n/t}$ and $\frac{1}{\varepsilon}\sqrt{n/|x|}$ [6, 7, 1], however, using Theorem 16, the optimal query complexities can be recovered.

3 Applications

In this section, we will demonstrate how to apply Theorem 15 and 16 to get new quantum algorithms. Specifically, we will give upper bounds of $\widetilde{O}(n\sqrt{R_{s,t}}/\varepsilon^{3/2})$ and $\widetilde{O}(n\sqrt{R_{s,t}/\lambda_2}/\varepsilon)$ on the time complexity of estimating the effective resistance, $R_{s,t}$, between two vertices, s and t, in a graph. Unlike previous upper bounds, we study this problem in the adjacency model.

A *unit flow* from s to t in G is a real-valued function θ on the directed edges $\vec{E}(G) = \{(u,v) : \{u,v\} \in E(G)\}$ such that:

1. for all $(u,v) \in \vec{E}$, $\theta(u,v) = -\theta(v,u)$;
2. for all $u \in [n] \setminus \{s,t\}$, $\sum_{v\in\Gamma(u)} \theta(u,v) = 0$, where $\Gamma(u) = \{v \in [n] : \{u,v\} \in E\}$; and
3. $\sum_{u\in\Gamma(s)} \theta(s,u) = \sum_{u\in\Gamma(t)} \theta(u,t) = 1$.

Let $\mathcal{F}$ be the set of unit flows from s to t in G. The *effective resistance* from s to t in G is defined:

$$R_{s,t}(G) = \min_{\theta\in\mathcal{F}} \sum_{\{u,v\}\in E(G)} \theta(u,v)^2.$$

This quantity gives the resistance of a network of unit resistors described by G, but is also an interesting quantity for graph theoretic reasons. For instance, the commute time between s and t, which is the expected number of steps in a random walk starting from s to reach t, and then return to s, is exactly the product of the number of edges in G, and $R_{s,t}(G)$ [8].

In the adjacency model, the input is a string $x \in \{0,1\}^{n\times n}$, representing a graph $G_x = ([n], \{\{i,j\} : x_{i,j} = 1\})$ (we assume that $x_{i,i} = 0$ for all i, and $x_{i,j} = x_{j,i}$ for all i,j). The problem of st-connectivity is the following. Given $x \in \{0,1\}^{n\times n}$ and $s,t \in [n]$, decide if there exists a path from s to t in G_x. A span-program-based algorithm for this problem was given in [5], with time complexity $\widetilde{O}(n\sqrt{p})$, under the promise that, if s and t are connected in G_x, they are connected by a path of length $\leq p$. They use the following span program, defined on $\{0,1\}^{n\times n}$:

$$H_{(u,v),0} = \{0\},\ \ H_{(u,v),1} = \mathrm{span}\{|u,v\rangle\},\ \ V = \mathbb{R}^n,\ \ A = \sum_{u,v\in[n]} (|u\rangle - |v\rangle)\langle u,v|,\ \ |\tau\rangle = |s\rangle - |t\rangle.$$

We have $H = \mathrm{span}\{|u,v\rangle : u,v \in [n]\}$, and $H(x) = \mathrm{span}\{|u,v\rangle : \{u,v\} \in E(G_x)\}$. Throughout this section, P will denote the above span program. We will use this span program to define algorithms for estimating the effective resistance. Ref. [5] are even able to show how to efficiently implement a unitary similar to $U(P,x)$, giving a time efficient algorithm. In the full version, we adapt their proof to our setting, showing that our algorithms are time efficient as well.

The effective resistance between s and t is related to st-connectivity by the fact that if s and t are not connected, then $R_{s,t}$ is undefined (there is no flow from s to t) and if s and t are connected then $R_{s,t}$ is related to the number and length of paths from s to t. In particular, if s and t are connected by a path of length p, then $R_{s,t}(G) \leq p$ (take the unit flow that simply travels along this path). In general, if s and t are connected in G, then $\frac{2}{n} \leq R_{s,t}(G) \leq n-1$. The span program for st-connectivity is amenable to the task of estimating the effective resistance due to the following.

▶ **Lemma 17** ([5]). *For any graph G_x on $[n]$, $x \in P_1$ if and only if s and t are connected, and in that case, $w_+(x,P) = \frac{1}{2}R_{s,t}(G_x)$.*

A near immediate consequence of this, combined with Theorem 15, is the following.

▶ **Theorem 18.** *There exists a quantum algorithm for estimating $R_{s,t}(G_x)$ to accuracy ε with time complexity $\widetilde{O}\left(\frac{n\sqrt{R_{s,t}(G_x)}}{\varepsilon^{3/2}}\right)$ and space complexity $O(\log n)$.*

By analyzing the spectra of A and $A(x)$, and applying Theorem 16, we can get an often better algorithm (Theorem 19). The *spectral gap* of a graph G, denoted $\lambda_2(G)$, is the second largest eigenvalue (including multiplicity) of the Laplacian of G, which is defined $L_G = \sum_{u\in[n]} d_u|u\rangle\langle u| - \sum_{u\in[n]}\sum_{v\in\Gamma(u)}|u\rangle\langle v|$, where d_u is the degree of u, and $\Gamma(u)$ is the set of neighbours of u. The smallest eigenvalue of L_G is 0 for any graph G. A graph G is connected if and only if $\lambda_2(G) > 0$. A connected graph G has $\frac{2}{n^2} \leq \lambda_2(G) \leq n$.

The following theorem is an improvement over Theorem 18 when $\lambda_2(G) > \varepsilon$. In particular, it is an improvement for all ε when we know that $\lambda_2(G) > 1$.

▶ **Theorem 19.** *Let $\mathcal{G}$ be a family of graphs such that for all $x \in \mathcal{G}$, $\lambda_2(G_x) \geq \mu$. Let $f : \mathcal{G}\times[n]\times[n] \to \mathbb{R}_{>0}$ be defined by $f(x,s,t) = R_{s,t}(G_x)$. There exists a quantum algorithm for estimating f to relative accuracy ε that has time complexity $\widetilde{O}\left(\frac{1}{\varepsilon}n\sqrt{R_{s,t}(G_x)/\mu}\right)$ and space complexity $O(\log n)$.*

Proof. We will apply Theorem 16. We first compute $\||w_0\rangle\|^2$, in order to normalize P.

▶ **Lemma 20.** $\||w_0\rangle\|^2 = \frac{1}{n}$.

Proof. Recall that $|w_0\rangle = A^+\tau$. This is the smallest $|w_0\rangle$ such that $A|w_0\rangle = \tau$. Since $H(x) = H$ when G_x is the complete graph, by Lemma 17, we need only compute $R_{s,t}$ in the complete graph. It's simple to verify that the optimal unit st-flow in the complete graph has $\frac{1}{n}$ units of flow on every path of the form (s,u,t) for $u \in [n]\setminus\{s,t\}$, and $\frac{2}{n}$ units of flow on the edge (s,t). Thus, $R_{s,t}(K_n) = \sum_{u\in[n]\setminus\{s,t\}} 2(1/n)^2 + (2/n)^2 = 2/n$. Thus $\||w_0\rangle\|^2 = \frac{1}{2}R_{s,t}(K_n) = \frac{1}{n}$. ◀

Next, we compute the following:

▶ **Lemma 21.** *For any $x \in \mathcal{G}$, $\frac{\sigma_{\max}(A)}{\sigma_{\min}(A(x))} = \sqrt{\frac{n}{\lambda_2(G_x)}} \leq \sqrt{\frac{n}{\mu}}$, so $\kappa(f) \leq \sqrt{\frac{n}{\mu}}$.*

Proof. Let L_x denote the Laplacian of G_x. We have:

$$A(x)A(x)^T = \sum_{u\in[n]}\sum_{v\in\Gamma(u)}(|u\rangle - |v\rangle)(\langle u| - \langle v|) = 2\sum_{u\in[n]} d_u|u\rangle\langle u| - 2\sum_{u\in[n]}\sum_{v\in\Gamma(u)}|u\rangle\langle v| = 2L_x.$$

Thus, if L denotes the Laplacian of the complete graph, we also have $AA^T = 2L$. Letting J denote the all ones matrix, we have $L = (n-1)I - (J-I) = nI - J$, and since $J = n|u\rangle\langle u|$ where $|u\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^n |i\rangle$, if $|u_1\rangle,\ldots,|u_{n-1}\rangle,|u\rangle$ is any orthonormal basis of $\mathbb{R}^n$, then $L = n\sum_{i=1}^{n-1}|u_i\rangle\langle u_i| + n|u\rangle\langle u| - n|u\rangle\langle u| = \sum_{i=1}^{n-1} n|u_i\rangle\langle u_i|$, so the spectrum of L is 0, with multiplicity 1, and n with multiplicity $n-1$. Thus, the only nonzero singular value of A is $\sqrt{2n} = \sigma_{\max}(A)$. Furthermore, since $\lambda_2(G_x)$ is the smallest nonzero eigenvalue of L_x, and $A(x)A(x)^T = 2L_x$, $\sigma_{\min}(A(x)) = \sqrt{2\lambda_2(G_x)}$. The result follows. ◀

Finally, by scaling τ to $\frac{\tau}{\|A^+\tau\|} = n\tau$ to get a normalized span program, which has the effect of scaling all positive witnesses by n, we can apply Theorem 16 to get an algorithm that makes $\widetilde{O}\left(\frac{\kappa(f)}{\varepsilon}\sqrt{nw_+(x,P)}\right) = \widetilde{O}\left(\frac{1}{\varepsilon}\sqrt{n/\mu}\sqrt{nR_{s,t}}\right)$ calls to $U'(P,x)$. In the full version, we show that this algorithm has time complexity $\widetilde{O}\left(\frac{1}{\varepsilon}n\sqrt{R_{s,t}/\mu}\right)$ and space complexity $O(\log n)$. ◀

Both of our upper bounds have linear dependence on n, and this is optimal (see full version). Classically, the best known method of estimating the effective resistance is to compute it, which costs $O(m) = O(n^2)$, where m is the number of edges in the graph. This is accomplished by inverting the Laplacian.

The algorithms from Theorem 18 and 19 are the first quantum algorithms for estimating the effective resistance in the adjacency model, however, the problem has been studied previously in the edge-list model [22], where Wang obtains a quantum algorithm with complexity $\widetilde{O}\left(\frac{d^{3/2}\log n}{\Phi(G)^2\varepsilon}\right)$, where $\Phi(G) \leq 1$ is the conductance (or edge-expansion) of G. In the edge-list model, the input $x \in [n]^{[n]\times[d]}$ models a d-regular graph (or d-bounded degree graph) G_x by $x_{u,i} = v$ for some $i \in [d]$ whenever $\{u, v\} \in E(G_x)$. Wang requires edge-list queries to simulate walking on the graph, which requires constructing a superposition over all neighbours of a given vertex. This type of edge-list query can be simulated by $\sqrt{n/d}$ adjacency queries to a d-regular graph, using quantum search, so Wang's algorithm can be converted to an algorithm in the adjacency query model with cost $\widetilde{O}\left(\frac{d^{3/2}}{\Phi(G)^2\varepsilon}\sqrt{\frac{n}{d}}\right)$. We can compare our results to this by noticing that $R_{s,t} \leq \frac{1}{\lambda_2(G)}$ [8], implying that our algorithm always runs in time at most $\widetilde{O}\left(\frac{1}{\varepsilon}\frac{n}{\mu}\right)$. If G is a connected d-regular graph, then $\lambda_2(G) = d\delta(G)$, where $\delta(G)$ is the spectral gap of a random walk on G. By Cheeger inequalities, we have $\frac{\Phi^2}{2} \leq \delta$ [16], so the complexity of the algorithm from Theorem 19 is at most $\widetilde{O}\left(\frac{1}{\varepsilon}\frac{n}{d\delta}\right) = \widetilde{O}\left(\frac{1}{\varepsilon}\frac{n}{d\Phi^2}\right)$, which is an improvement over the bound of $\widetilde{O}\left(\frac{1}{\varepsilon}\frac{d^{3/2}}{\Phi^2}\sqrt{\frac{n}{d}}\right) = \widetilde{O}\left(\frac{1}{\varepsilon}\frac{d}{\Phi^2}\sqrt{n}\right)$ given by naively adapting Wang's algorithm to the adjacency model whenever $d > \sqrt[4]{n}$. In general our upper bound may be much better than $\frac{1}{\varepsilon}\frac{n}{d\Phi^2}$, since the Cheeger inequality is not tight, and $R_{s,t}$ can be much smaller than $\frac{1}{\lambda_2}$.

4 Conclusion and Open Problems

We have presented several new techniques for turning span programs into quantum algorithms, which we hope will have future applications. Specifically, given a span program P, in addition to algorithms for deciding any function f such that $f^{-1}(0) \subseteq P_0$ and $f^{-1}(1) \subseteq P_1$, we also show how to get several different algorithms for deciding a number of related threshold problems, as well as estimating the witness size. In addition to algorithms based on the standard effective spectral gap lemma, we also show how to get algorithms by analyzing the real phase gap.

We hope that the importance of this work lies not only in its potential for applications, but in the improved understanding of the structure and power of span programs. A number of very important quantum algorithms rely on a similar structure, using phase estimation of a unitary that depends on the input to distinguish between different types of inputs. Span-program-based algorithms represent a very general class of such algorithms, making them not only important to the study of the quantum query model, but to quantum algorithms in general.

The main avenue for future work is in applications of our techniques to obtain new quantum algorithms. We stress that *any* span program for a decision problem can now be turned into an algorithm for estimating the positive or negative witness size, if these correspond to some meaningful function, or deciding threshold functions related to the witness size. A natural source of potential future applications is in the rich area of property testing problems (for a survey, see [17]).

One final open problem is a possible relationship between estimating the witness size and the HHL algorithm [11]. The HHL algorithm can be used to estimate $\|M^+|u\rangle\|^2$, given the

state $|u\rangle$ and access to a row-computable linear operator M. When $M = A(x)$ and $|u\rangle = \tau$, this quantity is exactly $w_+(x)$, so if $A(x)$ is row-computable — that is, there is an efficient procedure for computing the i^{th} nonzero entry of the j^{th} row of $A(x)$, then HHL gives us yet another means of estimating the witness size, whose time complexity is known, rather than only its query complexity. We note that the complexity of HHL depends on $\frac{\sigma_{\max}(A(x))}{\sigma_{\min}(A(x))}$, the *condition number* of $A(x)$, which is upper bounded by $\frac{\sigma_{\max}(A)}{\sigma_{\min}(A(x))}$, upon which the complexity of some of our algorithms depends as well. We leave further exploration of this connection for future research.

References

1. R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48, 2001.
2. A. Belovs. Learning-graph-based quantum algorithm for k-distinctness. In *Prooceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 207–216, 2012.
3. A. Belovs. Span programs for functions with constant-sized 1-certificates. In *Proceedings of the 44th Symposium on Theory of Computing (STOC 2012)*, pages 77–84, 2012.
4. A. Belovs, A. M. Childs, S. Jeffery, R. Kothari, and F. Magniez. Time efficient quantum walks for 3-distinctness. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP 2013)*, pages 105–122, 2013.
5. A. Belovs and B. Reichardt. Span programs and quantum algorithms for st-connectivity and claw detection. In *Proceedings of the 20th European Symposium on Algorithms (ESA 2012)*, pages 193–204, 2012.
6. C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing (special issue on quantum computing)*, 26:1510–1523, 1997.
7. G. Brassard, P. Høyer, M. Mosca, and A. Tapp. Quantum amplitude amplification and estimation. In S. J. Lomonaca and H. E. Brandt, editors, *Quantum Computation and Quantum Information: A Millennium Volume*, volume 305 of *AMS Contemporary Mathematics Series Millennium Volume*, pages 53–74. AMS, 2002. `arXiv:quant-ph/0005055v1`.
8. A. K. Chandra, P. Raghavan, W. L. Ruzzo, R. Smolensky, and P. Tiwari. The electrical resistance of a graph captures its commute and cover times. *Computational Complexity*, 6(4):312–340, 1996.
9. R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca. Quantum algorithms revisited. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 454(1969):339–354, 1998.
10. L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th ACM Symposium on Theory of Computing (STOC 1996)*, pages 212–219, 1996.
11. A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103:150502, Oct 2009.
12. S. Jeffery. *Frameworks for Quantum Algorithms*. PhD thesis, University of Waterloo, 2014. Available at `http://uwspace.uwaterloo.ca/handle/10012/8710`.
13. M. Karchmer and A. Wigderson. On span programs. In *Proceedings of the IEEE 8th Annual Conference on Structure in Complexity Theory*, pages 102–111, 1993.
14. A. Kitaev. Quantum measurements and the Abelian stabilizer problem, 1995. `arXiv:quant-ph/9511026`.
15. T. Lee, R. Mittal, B. Reichardt, R. Špalek, and M. Szegedy. Quantum query complexity of state conversion. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 344–353, 2011.

16 D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2009.

17 A. Montanaro and R. de Wolf. A survey of quantum property testing, 2013. arXiv:1310.2035.

18 B. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, pages 544–551, 2009.

19 B. Reichardt. Reflections for quantum query algorithms. In *Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 560–569, 2011.

20 B. Reichardt and R. Špalek. Span-program-based quantum algorithm for evaluating formulas. *Theory of Computing*, 8(13):291–319, 2012.

21 M. Szegedy. Quantum speed-up of Markov chain based algorithms. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2004)*, pages 32–41, 2004.

22 G. Wang. Quantum algorithms for approximating the effective resistances in electrical networks, 2013. arXiv:1311.1851.

Power of Quantum Computation with Few Clean Qubits*

Keisuke Fujii[1], Hirotada Kobayashi[2], Tomoyuki Morimae[3], Harumichi Nishimura[4], Shuhei Tamate[5], and Seiichiro Tani[6]

1 **The Hakubi Center for Advanced Research and Quantum Optics Group, Division of Physics and Astronomy, Graduate School of Science, Kyoto University, Kyoto, Japan**
2 **Principles of Informatics Research Division, National Institute of Informatics, Tokyo, Japan**
3 **Advanced Scientific Research Leaders Development Unit, Gunma University, Kiryu, Gunma, Japan**
4 **Department of Computer Science and Mathematical Informatics, Graduate School of Information Science, Nagoya University, Nagoya, Aichi, Japan**
5 **Principles of Informatics Research Division, National Institute of Informatics, Tokyo, Japan**
6 **NTT Communication Science Laboratories, NTT Corporation, Atsugi, Kanagawa, Japan**

Abstract

This paper investigates the power of polynomial-time quantum computation in which only a very limited number of qubits are initially clean in the $|0\rangle$ state, and all the remaining qubits are initially in the totally mixed state. No initializations of qubits are allowed during the computation, nor are intermediate measurements. The main contribution of this paper is to develop unexpectedly strong error-reduction methods for such quantum computations that simultaneously reduce the number of necessary clean qubits. It is proved that any problem solvable by a polynomial-time quantum computation with one-sided bounded error that uses logarithmically many clean qubits is also solvable with exponentially small one-sided error using just two clean qubits, and with polynomially small one-sided error using just one clean qubit. It is further proved in the two-sided-error case that any problem solvable by such a computation with a constant gap between completeness and soundness using logarithmically many clean qubits is also solvable with exponentially small two-sided error using just two clean qubits. If only one clean qubit is available, the problem is again still solvable with exponentially small error in one of the completeness and soundness and with polynomially small error in the other. An immediate consequence is that the Trace Estimation problem defined with *fixed* constant threshold parameters is complete for $\mathrm{BQ}_{[1]}\mathrm{P}$ and $\mathrm{BQ}_{\log}\mathrm{P}$, the classes of problems solvable by polynomial-time quantum computations with completeness 2/3 and soundness 1/3 using just one and logarithmically many clean qubits, respectively. The techniques used for proving the error-reduction results may be of independent interest in themselves, and one of the technical tools can also be used to show the hardness of weak classical simulations of one-clean-qubit computations (i.e., DQC1 computations).

1998 ACM Subject Classification F.1.2 Modes of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases DQC1, quantum computing, complete problems, error reduction

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.13

* A full version [11] of this paper is available at arXiv.org e-Print archive, arXiv:1509.07276 [quant-ph].

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 13; pp. 13:1–13:14

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

1 Introduction

1.1 Background

One of the most important goals in quantum information processing is to realize a quantum mechanical machine whose computational ability is superior to classical computers. The ultimate goal is, of course, to realize a large scale universal quantum computer, which still seems to be many years off despite extensive experimental efforts. Plenty of attention has thus been paid to "intermediate" (i.e., non-universal) models of quantum computation, which are somehow easier to physically implement. Such intermediate models do not offer universal quantum computation, but are believed to still be able to solve some problems that are hard for classical computers.

The *deterministic quantum computation with one quantum bit (DQC1)*, often mentioned as the *one-clean-qubit model*, is one of the most well-studied examples of such intermediate models. This model was introduced by Knill and Laflamme [15] to reflect some actual experimental setups such as nuclear magnetic resonance (NMR), where pure clean qubits are very hard to prepare and therefore are considered as very expensive resources. For example, in nuclear spin ensemble systems such as liquid state NMR systems, it is usually extremely hard, although not impossible, to polarize a spin (i.e., to initialize a qubit to state $|0\rangle$), since energy scale of a nuclear spin qubit is quite small, while it is favorable for long coherence time. A DQC1 computation over w qubits starts with the initial state of the totally mixed state except for a single clean qubit, namely, $|0\rangle\langle 0| \otimes \left(\frac{I}{2}\right)^{\otimes(w-1)}$. After applying a polynomial-size *unitary* quantum circuit to this state, only a single output qubit is measured in the computational basis at the end of the computing in order to read out the computation result. No initializations of qubits are allowed during the computation, nor are intermediate measurements. The DQC1 model is believed not to have full computational power of the standard polynomial-time quantum computation, and is indeed strictly less powerful under some reasonable assumptions [5]. At first glance the model even looks easy to classically simulate and does not seem to offer any quantum advantage, partly because its highly-mixed initial state obviously lacks "quantumness" such as entanglement, coherence, and discord, which are widely believed to be origins of the power of quantum information processing, and also because any time-evolution over a single-qubit state or a totally mixed state is trivially simulatable by a classical computation. Nevertheless, the DQC1 model is not trivial, either, in the sense that it can efficiently solve several problems for which no efficient classical algorithms are known, such as estimating the spectral density [15], testing integrability [20], calculating the fidelity decay [19], approximating the Jones and HOMFLY polynomials [23, 13], and approximating an invariant of 3-manifolds [12]. As many of these problems have physically meaningful applications, the DQC1 model is one of the most important intermediate quantum computation models.

Despite its importance explained thus far and the fact that tons of papers in physics have focused on it, very little has been studied on the genuinely complexity-theoretic aspects of the DQC1 model (to the best knowledge of the authors, no such studies exist other than Refs. [5, 21, 22]). The primal purpose of the present paper is to establish for the first time the fundamental core of detailed complexity-theoretic treatments of the DQC1 model and its generalization. To provide the very base of the study of computational complexity of such models, this paper investigates how robust these models are against computation error.

Computation error is an inherent feature of quantum computing, as the outcome of a computation is inevitably probabilistic and hence may not always be correct. Error reduction, or success-probability amplification, is thus one of the most fundamental issues in quantum

computing. Computation error can be efficiently reduced to be negligibly small in many standard computation models via a simple repetition-based method. Typical examples are polynomial-time quantum computations with bounded error, and in particular, the error can be made exponentially small in BQP both in completeness and in soundness, which provides a reasonable ground for the well-used definition of BQP that employs bounds 2/3 and 1/3 for completeness and soundness, respectively. In many other computation models, however, it is unclear whether the error can be reduced efficiently by the standard repetition-based method, and more generally, whether error reduction itself is possible. Typically, for models with very limited computational resources like space-bounded quantum computations, it is simply impossible to repeat the original computation sufficiently many times, which becomes an enormous obstacle to error reduction when initializations of qubits are disallowed after the computation starts. Indeed, it is impossible in the case of one-way quantum finite state automata to reduce computation error below a certain constant [4]. Also, the reducibility of computation error is unclear in various logarithmic-space quantum computations. For computations of one-sided bounded error performed by logarithmic-space quantum Turing machines, Watrous [28] presented a nontrivial method that reduces the error to be exponentially small. Other than this result, error-reduction techniques have not been developed much for space-bounded quantum computations.[1]

The computation models with few clean qubits, including DQC1, may be viewed as variants of space-bounded quantum computations in a sense, and thus, it is highly nontrivial to reduce computation error in these models. On the other hand, the reducibility of computation error is particularly desirable in these models, as the DQC1 computations mentioned above that solve the classically-hard problems in fact solve the decisional versions of the problems only with two-sided bounded error. Computation error can be quite large in such computations, and the gap between completeness and soundness is allowed to be polynomially small. The only method known for amplifying success probability of these computations is to sequentially repeat an attempt of the computation polynomially many times, but this requires the clean qubit to be initialized every time after finishing one attempt, and moreover, the result of each attempt must be recorded to classical work space prepared outside of the DQC1 model. It is definitely more desirable if computation error can be reduced without such initializations, the operations that are very expensive for the model. The situation is similar even when the number of clean qubits is allowed to be logarithmically many with respect to the input length. It is also known that any quantum computation of two-sided bounded error that uses logarithmically many clean qubits can be simulated by a quantum computation still of two-sided bounded error that uses just one clean qubit, but the known method for this simulation considerably increases the computational error, and the gap between completeness and soundness becomes polynomially small.

1.2 The results

This paper develops methods of reducing computation error in quantum computations with few clean qubits, including the DQC1 model. As will be presented below, the methods

[1] After the completion of this work, Fefferman, Kobayashi, Lin, Morimae, and Nishimura [10] developed methods of error reduction for space-bounded unitary quantum computations. Both of this very recent method and the one by Watrous [28] do not apply to quantum computations with few clean qubits, for these methods assume the easiness of "exact initialization check" (i.e., the easiness of checking whether the given state is *exactly* equal to the initial state of the computation), which is no longer the case for quantum computations with few clean qubits where many qubits are initially in the totally mixed state.

proposed are unexpectedly powerful and are able to simultaneously reduce both computation error and the number of necessary clean qubits, providing an almost fully satisfying solution in the cases of one-sided bounded error. In the two-sided-error case, the methods in this paper are applicable only when there is a constant gap between completeness and soundness in the original computation, but still significantly improve the situation of quantum computations with few clean qubits as to both the reducibility of computation error and the reducibility of the number of necessary clean qubits. These results are the first error-reducible properties for intermediate quantum computation models, not limited to the DQC1 model.

The results may alternatively be interpreted as that any problem solvable by a DQC1 computation with constant computation error is still solvable with constant computation error even when a bit noisy initial state is given instead of the ideal one-clean-qubit state, for the problem is also solvable with very small error when given an ideal one-clean-qubit initial state, thanks to the error-reduction results. This is perhaps very helpful in actual implementation, as the initial state prepared does not need to be very close to the ideal one-clean-qubit state, and can be away from it by, say, a constant δ in trace distance to still have success probability close to $1-\delta$. In particular, the qubit that is supposed to be clean does not need to be thoroughly purified and may be noisy to some extent.

The result for the two-sided-error case has another implication that the power of DQC1 computations with small two-sided error is characterized by the Trace Estimation problem defined with *fixed* constant threshold parameters. This may also be viewed as the first "gap amplification" result for the Trace Estimation problem. The Trace Estimation problem is ubiquitous in quantum many-body physics as observables, related to various important quantities in physics like the fidelity decay characterizing quantum chaos [9] and the Jones polynomials corresponding to the expected values of the Wilson loops in SU(2) Chern-Simons topological quantum field theory [29]. The results thus provide a useful tool to understand computational complexity of such quantum many-body systems, and establish a new bridge between computational complexity theory and quantum many-body physics.

Simultaneous reducibility of computation error and the number of clean qubits. Let $\mathrm{Q_{log}P}(c,s)$, $\mathrm{Q_{[1]}P}(c,s)$, and $\mathrm{Q_{[2]}P}(c,s)$ denote the classes of problems solvable by polynomial-time quantum computations with completeness c and soundness s that uses logarithmically many clean qubits, one clean qubit, and two clean qubits, respectively. First, in the one-sided-error case, it is proved that any problem solvable by a polynomial-time quantum computation with one-sided bounded error that uses logarithmically many clean qubits is also solvable by that with exponentially small one-sided error using just two clean qubits. If only one clean qubit is available, the problem is still solvable with polynomially small one-sided error (and thus with any small constant one-sided error).

▶ **Theorem 1.1.** *For any polynomially bounded function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ and any polynomial-time computable function $s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $1-s \geq \frac{1}{q}$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$,*

$$\mathrm{Q_{log}P}(1,s) \subseteq \mathrm{Q_{[2]}P}(1,2^{-p}).$$

▶ **Theorem 1.2.** *For any polynomially bounded function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ and any polynomial-time computable function $s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $1-s \geq \frac{1}{q}$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$,*

$$\mathrm{Q_{log}P}(1,s) \subseteq \mathrm{Q_{[1]}P}\Big(1, \frac{1}{p}\Big).$$

The above two theorems are for the case of perfect completeness, and similar statements hold even for the case of perfect soundness, by considering the complement of the problem.

▶ **Corollary 1.3.** *For any polynomially bounded function* $p\colon \mathbb{Z}^+ \to \mathbb{N}$ *and any polynomial-time computable function* $c\colon \mathbb{Z}^+ \to [0,1]$ *satisfying* $c \geq \frac{1}{q}$ *for some polynomially bounded function* $q\colon \mathbb{Z}^+ \to \mathbb{N}$*,*

$$\mathrm{Q_{log}P}(c,0) \subseteq \mathrm{Q_{[2]}P}(1-2^{-p},0) \quad \textit{and} \quad \mathrm{Q_{log}P}(c,0) \subseteq \mathrm{Q_{[1]}P}\Big(1-\frac{1}{p},0\Big).$$

In the two-sided-error case, it is proved that any problem solvable by a polynomial-time quantum computation that uses logarithmically many clean qubits and has a constant gap between completeness and soundness can also be solved by that with exponentially small two-sided error using just two clean qubits. If only one clean qubit is available, the problem is again still solvable with exponentially small error in one of the completeness and soundness and polynomially small error in the other.

▶ **Theorem 1.4.** *For any polynomially bounded function* $p\colon \mathbb{Z}^+ \to \mathbb{N}$ *and any constants* c *and* s *in* $(0,1)$ *satisfying* $c > s$*,*

$$\mathrm{Q_{log}P}(c,s) \subseteq \mathrm{Q_{[2]}P}(1-2^{-p},2^{-p}).$$

▶ **Theorem 1.5.** *For any polynomially bounded function* $p\colon \mathbb{Z}^+ \to \mathbb{N}$ *and any constants* c *and* s *in* $(0,1)$ *satisfying* $c > s$*,*

$$\mathrm{Q_{log}P}(c,s) \subseteq \mathrm{Q_{[1]}P}\Big(1-2^{-p},\frac{1}{p}\Big) \cap \mathrm{Q_{[1]}P}\Big(1-\frac{1}{p},2^{-p}\Big).$$

The ideas for the proofs of these statements and techniques developed therein may be of independent interest in themselves, and will be overviewed in Section 2.

Completeness results for Trace Estimation problem. Define the complexity classes $\mathrm{BQ_{log}P}$ and $\mathrm{BQ_{[1]}P}$ by $\mathrm{BQ_{log}P} = \mathrm{Q_{log}P}\big(\frac{2}{3},\frac{1}{3}\big)$ and $\mathrm{BQ_{[1]}P} = \mathrm{Q_{[1]}P}\big(\frac{2}{3},\frac{1}{3}\big)$, respectively. An immediate but important consequence of Theorem 1.5 is that the Trace Estimation problem is complete for $\mathrm{BQ_{log}P}$ and $\mathrm{BQ_{[1]}P}$ under polynomial-time many-one reduction, even when the problem is defined with *fixed* constant parameters that specify the bounds on normalized traces in the yes-instance and no-instance cases.

Given a description of a quantum circuit that specifies a unitary transformation U, the Trace Estimation problem specified with two parameters a and b satisfying $-1 \leq b < a \leq 1$ is the problem of deciding whether the real part of the normalized trace of U is at least a or it is at most b.

Trace Estimation Problem: $\mathrm{TrEst}(a,b)$

Input:	A description of a quantum circuit Q that implements a unitary transformation U over n qubits.
Yes Instances:	$\frac{1}{2^n}\Re(\mathrm{tr}\,U) \geq a$.
No Instances:	$\frac{1}{2^n}\Re(\mathrm{tr}\,U) \leq b$.

The paper by Knill and Laflamme [15] that introduced the DQC1 model already pointed out that this problem is closely related to the DQC1 computation. This point was further clarified in the succeeding literature (see Refs. [21, 22, 23], for instance). More precisely, consider a variant of the Trace Estimation problem where the two parameters a and b may depend on the input length (i.e., the length of the description of Q). It is known that this version of the Trace Estimation problem, for any a and b such that the gap $a - b$ is bounded from below by an inverse-polynomial with respect to the input length, can be solved by a DQC1 computation with *some* two-sided bounded error where the completeness and soundness parameters c and s depend on a and b. It is also known that, for any two nonnegative parameters a and b such that the gap $a - b$ is bounded from below by an inverse-polynomial with respect to the input length, the corresponding version of the Trace Estimation problem is hard for the complexity class $\mathrm{Q}_{[1]}\mathrm{P}(c, s)$ for *some* completeness and soundness parameters c and s that depend on a and b. Hence, the Trace Estimation problem essentially characterizes the power of the DQC1 computation. One subtle matter to be pointed out in the existing arguments above is that, when the parameters a and b are fixed for the Trace Estimation problem, the completeness c and soundness s with which the problem is in $\mathrm{Q}_{[1]}\mathrm{P}(c, s)$ are *different* from the completeness c' and soundness s' with which the problem is hard for $\mathrm{Q}_{[1]}\mathrm{P}(c', s')$. Namely, given two nonnegative parameters a and b of the problem, the computation solves the problem with completeness $c = (1 + a)/2$ and soundness $s = (1 + b)/2$, while the problem is hard for the class with completeness $c' = a/4$ and soundness $s' = b/4$. Therefore, the existing arguments are slightly short for proving $\mathrm{BQ}_{[1]}\mathrm{P}$-completeness of the Trace Estimation problem with fixed parameters a and b (and $\mathrm{Q}_{[1]}\mathrm{P}(c, s)$-completeness of that for fixed completeness and soundness parameters c and s, in general).

In contrast, with Theorem 1.5 in hand, it is immediate to show that the Trace Estimation problem is complete for $\mathrm{BQ}_{\log}\mathrm{P}$ and for $\mathrm{BQ}_{[1]}\mathrm{P}$ for any constants a and b satisfying $0 < b < a < 1$.

▶ **Theorem 1.6.** *For any constants a and b in $(0, 1)$ satisfying $a > b$, $\mathrm{TrEst}(a, b)$ is complete for $\mathrm{BQ}_{\log}\mathrm{P}$ and for $\mathrm{BQ}_{[1]}\mathrm{P}$ under polynomial-time many-one reduction.*

Hardness of weak classical simulations of DQC1 computation. Recently, quite a few number of studies focused on the hardness of *weak* classical simulations of restricted models of quantum computing under some reasonable assumptions [26, 7, 2, 18, 14, 17, 25, 8, 24]. Namely, a plausible assumption in complexity theory leads to the impossibility of efficient sampling by a classical computer according to an output probability distribution generatable with a quantum computing model. Among them are the IQP model [7] and the Boson sampling [2], both of which are proved hard for classical computers to simulate within multiplicative error, unless the polynomial-time hierarchy collapses to the third level (in fact, the main result of Ref. [2] is a much more meaningful hardness result on the weak simulatability of the Boson sampling within *polynomially small additive error*, but which needs a much stronger complexity assumption than the collapse of polynomial-time hierarchy).

An interesting question to ask is whether a similar result holds even for the DQC1 model. Very recently, Morimae, Fujii, and Fitzsimons [17] settled the case of the $\mathrm{DQC1}_m$-type computation, the generalization of the DQC1 model that allows m output qubits to be measured at the end of the computation, by proving that a $\mathrm{DQC1}_m$-type computation with $m \geq 3$ cannot be simulated within multiplicative error unless the polynomial-time hierarchy collapses to the third level. Their proof essentially shows that any PostBQP circuit can be simulated by a $\mathrm{DQC1}_3$-type computation, where PostBQP is the complexity class

corresponding to bounded-error quantum polynomial-time computations with postselection, which is known equivalent to PP [1]. By an argument similar to that in Ref. [7], it follows that PP is in PostBPP (the version of BPP with postselection), if the $\mathrm{DQC1}_3$-type computation is classically simulatable within multiplicative error. Together with Toda's theorem [27], this implies the collapse of the polynomial-time hierarchy to the third level.

One obvious drawback of the existing argument above is an inevitable postselection measurement inherent to the definition of PostBQP. This becomes a quite essential obstacle when trying to extend this argument to the DQC1 model, where only one qubit is allowed to be measured. To deal with the DQC1 model, this paper takes a different approach by considering the complexity class NQP introduced in Ref. [3] or the class SBQP introduced in Ref. [16]. Let $\mathrm{NQ}_{[1]}\mathrm{P}$ and $\mathrm{SBQ}_{[1]}\mathrm{P}$ be the variants of NQP and SBQP, respectively, in which the quantum computation performed is restricted to a DQC1 computation. From one of the technical tools used for proving the main results of this paper, it is immediate to show that the restriction to a DQC1 computation does not change the classes NQP and SBQP.

▶ **Theorem 1.7.** $\mathrm{NQP} = \mathrm{NQ}_{[1]}\mathrm{P}$ *and* $\mathrm{SBQP} = \mathrm{SBQ}_{[1]}\mathrm{P}$.

If any DQC1 computation were classically simulatable within multiplicative error, however, the class $\mathrm{NQ}_{[1]}\mathrm{P}$ would be included in NP and the class $\mathrm{SBQ}_{[1]}\mathrm{P}$ would be included in SBP, where SBP is a classical version of SBQP in short, introduced in Ref. [6]. Similarly, if any DQC1 computation were classically simulatable within exponentially small additive error, both $\mathrm{NQ}_{[1]}\mathrm{P}$ and $\mathrm{SBQ}_{[1]}\mathrm{P}$ would be included in SBP. Combined with Theorem 1.7, any of the inclusions $\mathrm{NQ}_{[1]}\mathrm{P} \subseteq \mathrm{NP}$, $\mathrm{SBQ}_{[1]}\mathrm{P} \subseteq \mathrm{SBP}$, and $\mathrm{NQ}_{[1]}\mathrm{P} \subseteq \mathrm{SBP}$ further implies an implausible consequence that PH = AM, which in particular implies the collapse of the polynomial-time hierarchy to the second level. Accordingly, the following theorem holds.

▶ **Theorem 1.8.** *The DQC1 model is not classically simulatable either within multiplicative error or exponentially small additive error, unless* PH = AM.

The above argument based on NQP and SBQP to prove Theorem 1.8 is very general, and can also be used to show the hardness of weak classical simulations of other quantum computing models. In particular, it can replace the existing argument based on PostBQP, which was developed in Ref. [7] and has appeared frequently in the literature [2, 14, 17, 25, 8, 24]. This also weakens the complexity assumption necessary to prove the hardness results for such models, including the IQP model [7] and the Boson sampling [2] (the polynomial-time hierarchy now collapses to the second level, rather than the third level when using PostBQP). Moreover, the hardness results for such models now hold for any constant multiplicative error $c \geq 1$, rather than only for c satisfying $1 \leq c < \sqrt{2}$ as in Refs. [7, 17].

2 Overview of error-reduction results

This section presents an overview of the proofs for the error reduction results. First, Subsection 2.1 provides high-level descriptions of the proofs of Theorems 1.1 and 1.2, the theorems for the one-sided error case of perfect completeness. Compared with the two-sided-error case, the proof construction is relatively simpler in the perfect-completeness case, but already involves most of key technical ingredients of this paper. Subsection 2.2 then explains the further idea that proves Theorems 1.4 and 1.5, the theorems for the two-sided-error case.

2.1 Proof ideas of Theorems 1.1 and 1.2

Let $A = (A_{\mathrm{yes}}, A_{\mathrm{no}})$ be any problem in $\mathrm{Q_{log}P}(1, s)$, where the function s defining the soundness is bounded away from one by an inverse-polynomial, and consider a polynomial-time

uniformly generated family of quantum circuits that puts A in $\mathrm{Q_{log}P}(1, s)$. Let Q_x denote the quantum circuit from this family when the input is x, where Q_x acts over $w(|x|)$ qubits for some polynomially bounded function w, and is supposed to be applied to the initial state $(|0\rangle\langle 0|)^{\otimes k(|x|)} \otimes \left(\frac{I}{2}\right)^{\otimes(w(|x|)-k(|x|))}$ that contains exactly $k(|x|)$ clean qubits, for some logarithmically bounded function k.

Theorems 1.1 and 1.2 are proved by constructing circuits with desirable properties from the original circuit Q_x. The construction is essentially the same for both of the two theorems and consists of three stages of transformations of circuits: The first stage reduces the number of necessary clean qubits to just one, while keeping perfect completeness and soundness still bounded away from one by an inverse-polynomial. The second stage then makes the acceptance probability of no-instances arbitrarily close to 1/2, still using just one clean qubit and keeping perfect completeness. Here, it not only makes the soundness (i.e., the upper bound of the acceptance probability of no-instances) close to 1/2, but also makes the acceptance probability of no-instances *at least* 1/2. Finally, in the case of Theorem 1.2, the third stage further reduces soundness error to be polynomially small with the use of just one clean qubit, while preserving the perfect completeness property. If one more clean qubit is available, the third stage can achieve exponentially small soundness, which leads to Theorem 1.1. The analyses of the third stage effectively use the fact that the acceptance probability of no-instances is close to 1/2 after the transformation of the second stage.

The rest of this subsection sketches the ideas that realize each of these three stages.

One-Clean-Qubit Simulation Procedure. The first stage uses a procedure called the One-Clean-Qubit Simulation Procedure. Given the quantum circuit Q_x with a specification of the number $k(|x|)$ of clean qubits, this procedure results in a quantum circuit R_x such that the input state to R_x is supposed to contain just one clean qubit, and when applied to the one-clean-qubit initial state, the acceptance probability of R_x is still one if x is in A_{yes}, while it is at most $1 - \delta(|x|)$ if x is in A_{no}, where δ is an inverse-polynomial function determined by $\delta = 2^{-k}(1 - s)$. It is stressed that the One-Clean-Qubit Simulation Procedure preserves perfect completeness, which is in stark contrast to the straightforward method of one-clean-qubit simulation.

Consider the $k(|x|)$-clean-qubit computation performed with Q_x. Let Q denote the quantum register consisting of the $k(|x|)$ initially clean qubits, and let R denote the quantum register consisting of the remaining $w(|x|) - k(|x|)$ qubits that are initially in the totally mixed state. Further let $\mathsf{Q}^{(1)}$ denote the single-qubit quantum register consisting of the first qubit of Q, which corresponds to the output qubit of Q_x. In the one-clean-qubit simulation of Q_x by R_x, the $k(|x|)$ qubits in Q are supposed to be in the totally mixed state initially and R_x tries to simulate Q_x only when Q initially contains the clean all-zero state. To do so, R_x uses another quantum register O consisting of just a single qubit, and this qubit in O is the only qubit that is supposed to be initially clean.

For ease of explanations, assume for a while that all the qubits in Q are also initially clean even in the case of R_x. The key idea in the construction of R_x is the following simulation of Q_x that makes use of the phase-flip transformation: The simulation first applies the Hadamard transformation H to the qubit in O and then flips the phase if and only if the content of O is 1 *and* the simulation of Q_x results in rejection (which is realized by performing Q_x to (Q, R) and then applying the controlled-Z transformation to $(\mathsf{O}, \mathsf{Q}^{(1)})$, where the content 1 in $\mathsf{Q}^{(1)}$ is assumed to correspond to the rejection in the original computation by Q_x). The simulation further performs the inverse of Q_x to (Q, R) and again applies H to O. At the end of the simulation, the qubit in O is measured in the computational basis, where measuring 0

corresponds to acceptance. The point is that this phase-flip-based construction provides a quite "faithful" simulation of Q_x, meaning that the rejection probability of the simulation is polynomially related to the rejection probability of the original computation of Q_x (and in particular, the simulation never rejects when the original computation never rejects, i.e., it preserves the perfect completeness property).

As mentioned before, all the qubits in Q are supposed to be in the totally mixed state initially in the one-clean-qubit simulation of Q_x by R_x, and R_x tries to simulate Q_x only when Q initially contains the clean all-zero state. To achieve this, each of the applications of the Hadamard transformation H is replaced by an application of the controlled-H transformation so that H is applied only when all the qubits in Q are in state $|0\rangle$. By considering the one-clean-qubit computations with the circuit family induced by R_x, the perfect completeness property is preserved and soundness is still bounded away from one by an inverse-polynomial (although the rejection probability becomes smaller for no-instances by a multiplicative factor of 2^{-k}, where notice that 2^{-k} is an inverse-polynomial as k is a logarithmically bounded function).

Randomness Amplification Procedure. The second stage uses the procedure called the Randomness Amplification Procedure. Given the circuit R_x constructed in the first stage, this procedure results in a quantum circuit R'_x such that the input state to R'_x is still supposed to contain just one clean qubit, and when applied to the one-clean-qubit initial state, the acceptance probability of R'_x is still one if x is in $A_{\rm yes}$, while it is in the interval $\left[\frac{1}{2}, \frac{1}{2} + \varepsilon(|x|)\right]$ if x is in $A_{\rm no}$ for some sufficiently small function ε.

Consider the one-clean-qubit computation performed with R_x. Let O denote the single-qubit register consisting of the initially clean qubit, which is also the output qubit of R_x. Let R denote the quantum register consisting of all the other qubits that are initially in the totally mixed state (by the construction of R_x, R consists of $w(|x|)$ qubits).

Suppose that the qubit in O is measured in the computational basis after R_x is applied to the one-clean-qubit initial state $|0\rangle\langle 0| \otimes \left(\frac{I}{2}\right)^{\otimes w(|x|)}$ in (O, R). Obviously from the property of R_x, the measurement results in 0 with probability exactly equal to the acceptance probability $p_{\rm acc}$ of the one-clean-qubit computation with R_x. Now suppose that R_x is applied to a slightly different initial state $|1\rangle\langle 1| \otimes \left(\frac{I}{2}\right)^{\otimes w(|x|)}$ in (O, R), where O initially contains $|1\rangle$ instead of $|0\rangle$ and all the qubits in R are again initially in the totally mixed state. The key property here to be proved is that, in this case, the measurement over the qubit in O in the computational basis results in 1 again with probability exactly $p_{\rm acc}$, the acceptance probability of the one-clean-qubit computation with R_x. This implies that, after the application of R_x to (O, R) with all the qubits in R being in the totally mixed state, the content of O remains the same with probability exactly $p_{\rm acc}$, and is flipped with probability exactly $1 - p_{\rm acc}$, the rejection probability of the original one-clean-qubit computation with R_x, regardless of the initial content of O.

The above observation leads to the following construction of the circuit R'_x. The construction of R'_x is basically a sequential repetition of the original circuit R_x. The number N of repetitions is polynomially many with respect to the input length $|x|$, and the point is that the register O is reused for each repetition, and only the qubits in R are refreshed after each repetition (by preparing N registers $\mathsf{R}_1, \ldots, \mathsf{R}_N$, each of which consists of $w(|x|)$ qubits, the same number of qubits as R, all of which are initially in the totally mixed state). After each repetition the qubit in O is measured in the computational basis (in the actual construction, this step is exactly simulated without any measurement – a single-qubit totally mixed state is prepared as a fresh ancilla qubit for each repetition so that the content of O is copied to

this ancilla qubit using the CNOT transformation, and this ancilla qubit is never touched after this CNOT application). Now, no matter which measurement result is obtained at the jth repetition for every j in $\{1,\ldots,N\}$, the register O is reused as it is, and the circuit R_x is simply applied to $(\mathsf{O},\mathsf{R}_{j+1})$ at the $(j+1)$st repetition. After the N repetitions, the qubit in O is measured in the computational basis, which is the output of R'_x (the output 0 corresponds to acceptance). The point is that at each repetition, the content of O is flipped with probability exactly equal to the rejection probability of the original one-clean-qubit computation of R_x. Taking into account that O is initially in state $|0\rangle$, the computation of R'_x results in acceptance if and only if the content of O is flipped even number of times during the N repetitions. An analysis on Bernoulli trials then shows that, when the acceptance probability of the original one-clean-qubit computation of R_x was in the interval $\left[\frac{1}{2},1\right)$, the acceptance probability of the one-clean-qubit computation of R'_x is at least $1/2$ and converges linearly to $1/2$ with respect to the repetition number. On the other hand, when the acceptance probability of the original R_x was one, the content of O is never flipped during the computation of R'_x, and thus the acceptance probability of R'_x remains one.

Stability Checking Procedures. In the case of Theorem 1.2, the third stage uses the procedure called the One-Clean-Qubit Stability Checking Procedure. Given the circuit R'_x constructed in the second stage, this procedure results in a quantum circuit R''_x such that the input state to R''_x is still supposed to contain just one clean qubit, and when applied to the one-clean-qubit initial state, the acceptance probability of R''_x is still one if x is in A_{yes}, while it is $1/p(|x|)$ if x is in A_{no} for a polynomially bounded function p predetermined arbitrarily.

Consider the one-clean-qubit computation performed with R'_x. Let Q denote the single-qubit register consisting of the initially clean qubit, which is also the output qubit of R'_x. Let R denote the quantum register consisting of all the other qubits that are initially in the totally mixed state, and let $w'(|x|)$ denote the number of qubits in R.

Again the key observation is that, after the application of R'_x to (Q,R) with all the qubits in R being in the totally mixed state (followed by the measurement over the qubit in Q in the computational basis), the content of Q is flipped with probability exactly equal to the rejection probability of the original one-clean-qubit computation with R'_x, regardless of the initial content of Q.

This leads to the following construction of the circuit R''_x. The construction of R''_x is again basically a sequential repetition of the original circuit R'_x, but this time the qubit in Q is also supposed to be initially in the totally mixed state. The circuit R'_x is repeatedly applied $2N$ times, where N is a power of two and is polynomially many with respect to the input length $|x|$, and again the register Q is reused for each repetition, and only the qubits in R are refreshed after each repetition (by preparing $2N$ registers $\mathsf{R}_1,\ldots,\mathsf{R}_{2N}$, each of which consists of $w'(|x|)$ qubits, all of which are initially in the totally mixed state). The key idea for the construction of R''_x is to use a counter that counts the number of attempts such that the measurement over the qubit in Q results in $|1\rangle$ after the application of R'_x (again each measurement is simulated by a CNOT application using an ancilla qubit of a totally mixed state). Notice that the content of Q is never flipped regardless of the initial content of Q, if the original acceptance probability is one in the one-clean-qubit computation with R'_x. Hence, in this case the counter value either stationarily remains its initial value or is increased exactly by $2N$, the number of repetitions. On the other hand, if the original acceptance probability is close to $1/2$ in the one-clean-qubit computation with R'_x, the content of Q is flipped with probability close to $1/2$ after each application of R'_x regardless of the initial

content of Q. This means that, after each application of R'_x, the measurement over the qubit in Q results in $|1\rangle$ with probability close to 1/2 regardless of the initial content of Q, and thus, the increment of the counter value must be distributed around $\frac{1}{2} \cdot 2N = N$ with very high probability. Now, if the counter value is taken modulo $2N$ and if the unique initially clean qubit is prepared for the most significant bit of the counter (which picks the initial counter value from the set $\{0, \ldots, N-1\}$ uniformly at random), the computational-basis measurement over this most significant qubit of the counter always results in $|0\rangle$ if x is in A_{yes}, while it results in $|1\rangle$ with very high probability if x is in A_{no} (which can be made at least $1 - \frac{1}{p(|x|)}$ for an arbitrarily chosen polynomially bounded function p, by taking an appropriately large number N).

One drawback of the construction of R''_x above via the One-Clean-Qubit Stability Checking Procedure is that, in the case of no-instances, there inevitably exist some "bad" initial counter values in $\{0, \ldots, N-1\}$ with which R''_x is forced to accept with unallowably high probability. For instance, if the initial counter value is 0, R''_x is forced to accept when the increment of the counter is less than N, which happens with probability at least a constant. This is the essential reason why the current approach achieves only a polynomially small soundness in the one-clean-qubit case in Theorem 1.2, as the number of possible initial counter values can be at most polynomially many (otherwise the number of repetitions must be super-polynomially many) and even just one "bad" initial value is problematic to go beyond polynomially small soundness. In contrast, if not just one but two clean qubits are available, one can remove the possibility of "bad" initial counter values, which results in the Two-Clean-Qubit Stability Checking Procedure. This time, the circuit R'_x is repeatedly applied $8N$ times, and the counter value is taken modulo $8N$. The two initially clean qubits are prepared for the most and second-most significant bits of the counter, which results in picking the initial counter value from the set $\{0, \ldots, 2N-1\}$ uniformly at random. Now the point is that the counter value can be increased by N before the repetition so that the actual initial value of the counter is in $\{N, \ldots, 3N-1\}$, which discards the tail sets $\{0, \ldots, N-1\}$ and $\{3N, \ldots, 4N-1\}$ of the set $\{0, \ldots, 4N-1\}$. As the size of the tail sets discarded is sufficiently large, there no longer exists any "bad" initial counter value, which leads to the exponentially small soundness in the two-clean-qubit case in Theorem 1.1.

2.2 Proof ideas of Theorems 1.4 and 1.5

The results for the two-sided error case need more complicated arguments and is proved in eight stages of transformations in total, which are split into three parts.

The first part consists of three stages, and proves that any problem solvable with constant completeness and soundness using logarithmically many clean qubits is also solvable with constant completeness and soundness using just one clean qubit. At the first stage of the first part, by a standard repetition with a threshold-value decision, one first reduces errors to be sufficiently small constants, say, completeness 15/16 and soundness 1/16. For this, if the starting computation has a constant gap between completeness and soundness, one requires only a constant number of repetitions, and thus, the resulting computation still requires only logarithmically many clean qubits. The second stage of the first part then reduces the number of clean qubits to just one. The procedure in this stage is exactly the One-Clean-Qubit Simulation Procedure developed in the first stage of the one-sided error case. The gap between completeness and soundness becomes only an inverse-polynomial by this transformation, but the point is that the gap is still sufficiently larger (i.e., a constant times larger) than the completeness error. Now the third stage of the first part transforms the computation resulting from the second stage to the computation that still uses only one

clean qubit and has constant completeness and soundness. The procedure in this stage is exactly the Randomness Amplification Procedure, developed in the second stage of the one-sided error case, and it makes use of the difference of the rates of convergence to 1/2 of the acceptance probability between the yes- and no-instance cases.

The second part consists of two stages, and proves that any problem solvable with constant completeness and soundness using just one clean qubit is also solvable with almost-perfect (i.e., exponentially close to one) completeness and soundness below 1/2 using just logarithmically many clean qubits. At the first stage of the second part, one reduces both of the completeness and soundness errors to be polynomially small, again by a standard repetition with a threshold-value decision. Note that the computation resulting from the first part requires only one clean qubit. Thus, even when repeated logarithmically many times, the resulting computation uses just logarithmically many clean qubits, and achieves polynomially small errors. The second stage of the second part then repeatedly attempts the computation resulting from the first stage polynomially many times, and accepts if at least one of the attempts results in acceptance (i.e., takes OR of the attempts). A straightforward repetition requires polynomially many clean qubits, and to avoid this problem, after each repetition one tries to recover the clean qubits for reuse by applying the inverse of the computation (the failure of this recovery step is counted as an "acceptance" when taking the OR). This results in a computation that still requires only logarithmically many clean qubits, and has completeness exponentially close to one, while soundness is still below 1/2.

Now the third part is essentially the same as the three-stage transformation of the one-sided error case. From the computation resulting from the second part, the first stage of the third part decreases the number of clean qubits to just one, via the One-Clean-Qubit Simulation Procedure. The completeness of the resulting computation is still exponentially close to one and its soundness is bounded away from one by an inverse-polynomial. The second stage of the third part then applies the Randomness Amplification Procedure to make the acceptance probability of no-instances arbitrarily close to 1/2, while keeping completeness exponentially close to one. Finally, the third stage of the third part proves that one can further decrease soundness error to be polynomially small using just one qubit via the One-Clean-Qubit Stability Checking Procedure, or to be exponentially small using just two qubits via the Two-Clean-Qubit Stability Checking Procedure, while keeping completeness exponentially close to one.

By considering the complement problem, the above argument can also prove the case of exponentially small soundness error in Theorem 1.5.

Acknowledgements. The authors are grateful to Richard Cleve, François Le Gall, Keiji Matsumoto, and Yasuhiro Takahashi for very useful discussions. Keisuke Fujii is supported by the Grant-in-Aid for Research Activity Start-up No. 25887034 of the Japan Society for the Promotion of Science. Hirotada Kobayashi and Harumichi Nishimura are supported by the Grant-in-Aid for Scientific Research (A) Nos. 24240001 and 16H01705 of the Japan Society for the Promotion of Science. Tomoyuki Morimae is supported by the Program to Disseminate Tenure Tracking System of the Ministry of Education, Culture, Sports, Science and Technology in Japan, the Grant-in-Aid for Scientific Research on Innovative Areas No. 15H00850 of the Ministry of Education, Culture, Sports, Science and Technology in Japan, and the Grant-in-Aid for Young Scientists (B) No. 26730003 of the Japan Society for the Promotion of Science. Harumichi Nishimura is also supported by the Grant-in-Aid for Scientific Research on Innovative Areas No. 24106009 of the Ministry of Education, Culture, Sports, Science and Technology in Japan, which Hirotada Kobayashi and Seiichiro Tani are

also grateful to. Harumichi Nishimura further acknowledges support from the Grant-in-Aid for Scientific Research (C) No. 25330012 of the Japan Society for the Promotion of Science. Part of the work of Shuhei Tamate was done while this author was at the RIKEN Center for Emergent Matter Science, Wako, Saitama, Japan.

References

1 Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A*, 461(2063):3473–3482, 2005. doi:10.1098/rspa.2005.1546.

2 Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. *Theory of Computing*, 9:143–252, 2013. doi:10.4086/toc.2013.v009a004.

3 Leonard M. Adleman, Jonathan DeMarrais, and Ming-Deh A. Huang. Quantum computability. *SIAM Journal on Computing*, 26(5):1524–1540, 1997. doi:10.1137/S0097539795293639.

4 Andris Ambainis and Rūsiņš Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *39th Annual Symposium on Foundations of Computer Science*, pages 332–341, 1998. doi:10.1109/SFCS.1998.743469.

5 Andris Ambainis, Leonard J. Schulman, and Umesh Vazirani. Computing with highly mixed states. *Journal of the ACM*, 53(3):507–531, 2006. doi:10.1145/1147954.1147962.

6 Elmar Böhler, Christian Glaßer, and Daniel Meister. Error-bounded probabilistic computations between MA and AM. *Journal of Computer and System Sciences*, 72(6):1043–1076, 2006. doi:10.1016/j.jcss.2006.05.001.

7 Michael J. Bremner, Richard Jozsa, and Dan J. Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. *Proceedings of the Royal Society A*, 467(2126):459–472, 2011. doi:10.1098/rspa.2010.0301.

8 Daniel J. Brod. The complexity of simulating constant-depth BosonSampling. *Physical Review A*, 91(4):article 042316, 2015. doi:10.1103/PhysRevA.91.042316.

9 Joseph Emerson, Yaakov S. Weinstein, Seth Lloyd, and D. G. Cory. Fidelity decay as an efficient indicator of quantum chaos. *Physical Review Letters*, 89(28), 2002. doi:10.1103/PhysRevLett.89.284102.

10 Bill Fefferman, Hirotada Kobayashi, Cedric Yen-Yu Lin, Tomoyuki Morimae, and Harumichi Nishimura. Space-efficient error reduction for unitary quantum computations. In *Automata, Languages, and Programming, 43rd International Colloquium, ICALP 2016, Proceedings*, Leibniz International Proceedings in Informatics, 2016.

11 Keisuke Fujii, Hirotada Kobayashi, Tomoyuki Morimae, Harumichi Nishimura, Shuhei Tamate, and Seiichiro Tani. Power of quantum computation with few clean qubits. arXiv.org e-Print archive, arXiv:1509.07276 [quant-ph], 2015. arXiv:1507.05592.

12 Stephen P. Jordan and Gorjan Alagic. Approximating the Turaev-Viro invariant of mapping tori is complete for one clean qubit. In Dave Bacon, Miguel Martin-Delgado, and Martin Roetteler, editors, *Theory of Quantum Computation, Communication, and Cryptography, 6th Conference, TQC 2011, Madrid, Spain, May 24–26, 2011, Revised Selected Papers*, volume 6745 of *Lecture Notes in Computer Science*, pages 53–72. Springer-Verlag, 2014. doi:10.1007/978-3-642-54429-3_5.

13 Stephen P. Jordan and Pawel Wocjan. Estimating Jones and HOMFLY polynomials with one clean qubit. *Quantum Information and Computation*, 9(3–4):0264–0289, 2009.

14 Richard Jozsa and Maarten Van den Nest. Classical simulation complexity of extended Clifford circuits. *Quantum Information and Computation*, 14(7–8):0633–0648, 2014.

15 E. Knill and R. Laflamme. Power of one bit of quantum information. *Physical Review Letters*, 81(25):5672–5675, 1998. doi:10.1103/PhysRevLett.81.5672.

16 Greg Kuperberg. How hard is it to approximate the Jones polynomial? *Theory of Computing*, 11:183–219 (article 6), 2015. doi:10.4086/toc.2015.v011a006.

17 Tomoyuki Morimae, Keisuke Fujii, and Joseph F. Fitzsimons. Hardness of classically simulating the one-clean-qubit model. *Physical Review Letters*, 112(13):article 130502, 2014. doi:10.1103/PhysRevLett.112.130502.

18 Xiaotong Ni and Maarten Van den Nest. Commuting quantum circuits: Efficient classical simulations versus hardness results. *Quantum Information and Computation*, 13(1–2):0054–0072, 2013.

19 David Poulin, Robin Blume-Kohout, Raymond Laflamme, and Harold Ollivier. Exponential speedup with a single bit of quantum information: Measuring the average fidelity decay. *Physical Review Letters*, 92(17), 2004. doi:10.1103/PhysRevLett.92.177906.

20 David Poulin, Raymond Laflamme, G. J. Milburn, and Juan Pablo Paz. Testing integrability with a single bit of quantum information. *Physical Review A*, 68(2):article 022302, 2003. doi:10.1103/PhysRevA.68.022302.

21 D. J. Shepherd. Computation with unitaries and one pure qubit. arXiv.org e-Print archive, arXiv:quant-ph/0608132, 2006. arXiv:quant-ph/0608132.

22 Daniel James Shepherd. *Quantum Complexity : restrictions on algorithms and architectures*. PhD thesis, Department of Computer Science, Faculty of Engineering, University of Bristol, 2009. arXiv.org e-Print archive, arXiv:1005.1425 [cs.CC]. arXiv:1005.1425.

23 Peter W. Shor and Stephen P. Jordan. Estimating Jones polynomials is a complete problem for one clean qubit. *Quantum Information and Computation*, 8(8–9):0681–0714, 2008.

24 Yasuhiro Takahashi, Seiichiro Tani, Takeshi Yamazaki, and Kazuyuki Tanaka. Commuting quantum circuits with few outputs are unlikely to be classically simulatable. In *Computing and Combinatorics, 21st International Conference, COCOON 2015*, volume 9198 of *Lecture Notes in Computer Science*, pages 223–234, 2015. doi:10.1007/978-3-319-21398-9_18.

25 Yasuhiro Takahashi, Takeshi Yamazaki, and Kazuyuki Tanaka. Hardness of classically simulating quantum circuits with unbounded Toffoli and fan-out gates. *Quantum Information and Computation*, 14(13–14):1149–1164, 2014.

26 Barbara M. Terhal and David P. DiVincenzo. Adaptive quantum computation, constant depth quantum circuits and Arthur-Merlin games. *Quantum Information and Computation*, 4(2):134–145, 2004.

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

28 John Watrous. Quantum simulations of classical random walks and undirected graph connectivity. *Journal of Computer and System Sciences*, 62(2):376–391, 2001. doi:10.1006/jcss.2000.1732.

29 Edward Witten. Quantum field theory and the Jones polynomial. *Communications in Mathematical Physics*, 121(3):351–399, 1989. doi:10.1007/BF01217730.

Space-Efficient Error Reduction for Unitary Quantum Computations*

Bill Fefferman[1], Hirotada Kobayashi[2], Cedric Yen-Yu Lin[3], Tomoyuki Morimae[4], and Harumichi Nishimura[5]

1 **Joint Center for Quantum Information and Computer Science, University of Maryland, College Park, USA**
2 **Principles of Informatics Research Division, National Institute of Informatics, Tokyo, Japan**
3 **Joint Center for Quantum Information and Computer Science, University of Maryland, College Park, USA**
4 **Advanced Scientific Research Leaders Development Unit, Gunma University, Kiryu, Gunma, Japan**
5 **Department of Computer Science and Mathematical Informatics, Graduate School of Information Science, Nagoya University, Nagoya, Aichi, Japan**

Abstract

This paper presents a general space-efficient method for error reduction for unitary quantum computation. Consider a polynomial-time quantum computation with completeness c and soundness s, either with or without a witness (corresponding to QMA and BQP, respectively). To convert this computation into a new computation with error at most 2^{-p}, the most space-efficient method known requires extra workspace of $O\left(p \log \frac{1}{c-s}\right)$ qubits. This space requirement is too large for scenarios like logarithmic-space quantum computations. This paper shows an error-reduction method for unitary quantum computations (i.e., computations without intermediate measurements) that requires extra workspace of just $O\left(\log \frac{p}{c-s}\right)$ qubits. This in particular gives the first method of strong amplification for logarithmic-space unitary quantum computations with two-sided bounded error. This also leads to a number of consequences in complexity theory, such as the uselessness of quantum witnesses in bounded-error logarithmic-space unitary quantum computations, the PSPACE upper bound for QMA with exponentially-small completeness-soundness gap, and strong amplification for matchgate computations.

1998 ACM Subject Classification F.1.2 Modes of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases space-bounded computation, quantum Merlin-Arthur proof systems, error reduction, quantum computing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.14

1 Introduction

1.1 Background

A very basic topic in various models of quantum computation is whether computation error can be efficiently reduced within a given model. For polynomial-time bounded error quantum computation, the most standard model of quantum computation, the computation error can

* A full version [3] of this paper is available at arXiv.org e-Print archive, arXiv:1604.08192 [quant-ph].

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 14; pp. 14:1–14:14

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

be made exponentially small via a simple repetition followed by a threshold-value decision. This justifies the choice of 2/3 and 1/3 for the completeness and soundness parameters in the definition of the corresponding complexity class BQP. This is also the case for quantum Merlin-Arthur (QMA) proof systems, another central model of quantum computation that models a quantum analogue of NP (more precisely, MA), and the resulting class QMA may again be defined with completeness and soundness parameters 2/3 and 1/3.

An undesirable feature of the simple repetition-based error reduction above is that the necessary workspace enlarges linearly with respect to the number of repetitions. More explicitly, for a given p, the number of repetitions necessary to achieve an error of 2^{-p} is $O\big(\frac{p}{(c-s)^2}\big)$, and thus both the workspace size and the witness size become $O\big(\frac{p}{(c-s)^2}\big)$ times larger. This implies that the simple repetition-based method is no longer useful when either the workspace size or the witness size is required to be logarithmically bounded.

Marriott and Watrous [13] developed a more sophisticated method of error reduction for QMA proof systems that does not increase the witness size at all. For a given p, their method still requires $O\big(\frac{p}{(c-s)^2}\big)$ calls of the original computation and its inverse to achieve the computation error 2^{-p}, but the method reuses both the workspace and the witness every time it calls the original computation and its inverse. Hence, the witness size never increases in their method. This is a strong property that allows them to show the uselessness of logarithmic-size quantum witnesses in QMA proof systems (i.e., $\mathrm{QMA}_{\log} = \mathrm{BQP}$, where $\mathrm{QMA}_{\log}$ is the class of problems having QMA proof systems with logarithmic-size quantum witnesses). Their method is also more efficient in workspace size than the simple repetition-based method, but still requires extra workspace of size $O\big(\frac{p}{(c-s)^2}\big)$, as it must record outcomes of all the calls of the original computation and its inverse.

Nagaj, Wocjan, and Zhang [15] succeeded in reducing to $O\big(\frac{p}{c-s}\big)$ the number of calls of the original computation and its inverse necessary to achieve the computation error 2^{-p} for a given p, while keeping the witness size unchanged. Their method makes use of the phase-estimation algorithm, an essential component of many quantum algorithms including the celebrated factoring algorithm. To achieve error 2^{-p} for a given p, their method must repeat $O(p)$ times the phase-estimation algorithm with precision of at least $O\big(\log \frac{1}{c-s}\big)$ bits and record all these estimated phases. Hence, this phase-estimation-based method uses extra workspace of size $O\big(p \log \frac{1}{c-s}\big)$.

As can be seen from above, both of the Marriott-Watrous method and the phase-estimation-based method are still insufficient for the case where the workspace size must be logarithmically bounded. No efficient error-reduction method is known that keeps the size of additionally necessary workspace logarithmically bounded. This is not limited to the case of QMA proof systems, and in fact almost no efficient error-reduction method is known even in the case of logarithmic-space quantum computations, and in the case of space-bounded quantum computations in general. The study of general space-bounded quantum computations was initiated by Watrous [21] based on quantum Turing machines. Several models of space-bounded quantum computations have been proposed and investigated since then in the literature [22, 23, 24, 9, 14, 18], some considering only logarithmic-space quantum computations and others treating general cases. It is not known whether any of these models are computationally equivalent. It is also not known whether error reduction is possible for logarithmic-space quantum computation defined according to any of these models, except the only known affirmative answer shown by Watrous [22] on computation of one-sided bounded error performed by logarithmic-space quantum Turing machines. As negative evidence in the case where computational resources are too limited, computation error cannot be reduced below a certain constant for one-way quantum finite state automata [1].

1.2 Main result and its consequences

This paper presents a general method of strong and space-efficient error reduction for *unitary* quantum computations. In particular, the method is applicable to logarithmic-space unitary quantum computations and logarithmic-space unitary QMA proof systems. All the results in this paper are model-independent and hold with any model of space-bounded quantum computations as long as it performs *unitary* quantum computations. The unitary model is not the most general in that it does not allow any intermediate measurements (notice that the standard technique of simulating intermediate measurements by unitary gates requires unallowably many ancilla qubits in the case of space-bounded computations), but is arguably one of the most reasonable models of space-bounded quantum computation.

Let $\mathbb{N}$ and $\mathbb{Z}^+$ denote the sets of positive and nonnegative integers, respectively. Let $\mathrm{QMA}_{\mathbf{U}}\mathrm{SPACE}[l_{\mathsf{V}}, l_{\mathsf{M}}](c, s)$ denote the class of problems having QMA proof systems with completeness c and soundness s, where the verifier performs a *unitary* quantum computation that has no time bound but is restricted to use $l_{\mathsf{V}}(n)$ private qubits and to receive a quantum witness of $l_{\mathsf{M}}(n)$ qubits on every input of length n. The main result of this paper is the following strong and space-efficient error-reduction for such QMA-type computations.

▶ **Theorem 1.1.** *For any functions $p, l_{\mathsf{V}}, l_{\mathsf{M}} \colon \mathbb{Z}^+ \to \mathbb{N}$ and for any functions $c, s \colon \mathbb{Z}^+ \to [0, 1]$ satisfying $c > s$, there exists a function $\delta \colon \mathbb{Z}^+ \to \mathbb{N}$ that is logarithmic with respect to $\frac{p}{c-s}$ such that*

$$\mathrm{QMA}_{\mathbf{U}}\mathrm{SPACE}[l_{\mathsf{V}}, l_{\mathsf{M}}](c, s) \subseteq \mathrm{QMA}_{\mathbf{U}}\mathrm{SPACE}[l_{\mathsf{V}} + \delta, l_{\mathsf{M}}](1 - 2^{-p}, 2^{-p}).$$

As will be found later, the proof is based on a reduction that is in space logarithmic and also in time polynomial with respect to $\frac{p}{c-s}$. Actually, the argument used in the reduction is remarkably simple. Nevertheless, the theorem is very powerful in that it fruitfully leads to many consequences that substantially deepen the understanding on the power of QMA proof systems and quantum computations in general, both in the space-bounded scenario and in the usual polynomial-time scenario. In what follows, a function $f \colon \mathbb{Z}^+ \to \mathbb{N}$ is *polynomially bounded* if f is polynomial-time computable and $f(n)$ is in $O(n^d)$ for some constant $d > 0$, and is *logarithmically bounded* if f is logarithmic-space computable and $f(n)$ is in $O(\log n)$.

Strong amplification for unitary BQL. The first consequence of Theorem 1.1 is a remarkably strong error-reducibility in logarithmic-space unitary quantum computations. Let $\mathrm{Q}_{\mathbf{U}}\mathrm{L}(c, s)$ denote the class of problems solvable by logarithmic-space unitary quantum computations with completeness c and soundness s. The following amplifiability is immediate from Theorem 1.1 by taking a function p to be logarithmic-space computable and polynomially bounded, functions c and s to be logarithmic-space computable and to satisfy $c - s \geq 1/q$ for some polynomially bounded function $q \colon \mathbb{Z}^+ \to \mathbb{N}$, a function l_{V} to be logarithmically bounded, and a function $l_{\mathsf{M}} = 0$.

▶ **Corollary 1.2.** *For any polynomially bounded function $p \colon \mathbb{Z}^+ \to \mathbb{N}$ that is logarithmic-space computable and for any logarithmic-space computable functions $c, s \colon \mathbb{Z}^+ \to [0, 1]$ satisfying $c - s \geq 1/q$ for some polynomially bounded function $q \colon \mathbb{Z}^+ \to \mathbb{N}$,*

$$\mathrm{Q}_{\mathbf{U}}\mathrm{L}(c, s) \subseteq \mathrm{Q}_{\mathbf{U}}\mathrm{L}(1 - 2^{-p}, 2^{-p}).$$

This in particular justifies defining the bounded-error class $\mathrm{BQ}_{\mathbf{U}}\mathrm{L}$ of logarithmic-space unitary quantum computations by $\mathrm{BQ}_{\mathbf{U}}\mathrm{L} = \mathrm{Q}_{\mathbf{U}}\mathrm{L}(2/3, 1/3)$, employing $2/3$ and $1/3$ for completeness and soundness parameters. Before this work, Watrous [22] showed a similar strong error-reducibility in the case of one-sided bounded error, and Corollary 1.2 extends this to the two-sided bounded error case.

Uselessness of quantum witnesses in logarithmic-space unitary QMA. Let $\mathrm{QMA}_{\mathbf{U}}\mathrm{L}(c,s)$ denote the class of problems having logarithmic-space unitary QMA proof systems (i.e., such systems in which a verifier performs a logarithmic-space unitary computation upon receiving a logarithmic-size quantum witness) with completeness c and soundness s. Similarly to Corollary 1.2, the following amplifiability is immediate from Theorem 1.1 by taking a function p to be logarithmic-space computable and polynomially bounded, functions c and s to be logarithmic-space computable and to satisfy $c - s \geq 1/q$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$, and functions l_{V} and l_{M} to be logarithmically bounded.

▶ **Corollary 1.3.** *For any polynomially bounded function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ that is logarithmic-space computable and for any logarithmic-space computable functions $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c - s \geq 1/q$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$,*

$$\mathrm{QMA}_{\mathbf{U}}\mathrm{L}(c,s) \subseteq \mathrm{QMA}_{\mathbf{U}}\mathrm{L}(1-2^{-p}, 2^{-p}).$$

Again this justifies defining the bounded-error class $\mathrm{QMA}_{\mathbf{U}}\mathrm{L}$ of logarithmic-space unitary QMA proof systems by $\mathrm{QMA}_{\mathbf{U}}\mathrm{L} = \mathrm{QMA}_{\mathbf{U}}\mathrm{L}(2/3, 1/3)$. By a standard technique of replacing a quantum witness by a totally mixed state as a self-prepared witness (to do this in a unitary computation, one can simply prepare sufficiently many EPR pairs and then take a qubit from each pair), Corollary 1.3 together with Corollary 1.2 further implies the equivalence of $\mathrm{QMA}_{\mathbf{U}}\mathrm{L}$ and $\mathrm{BQ}_{\mathbf{U}}\mathrm{L}$.

▶ **Corollary 1.4.** $\mathrm{QMA}_{\mathbf{U}}\mathrm{L} = \mathrm{BQ}_{\mathbf{U}}\mathrm{L}$.

As mentioned before, Marriott and Watrous [13] showed the equivalence $\mathrm{QMA}_{\log} = \mathrm{BQP}$, the uselessness of quantum witnesses of logarithmic size in the standard QMA proof systems with a polynomial-time verifier. In this respect, Corollary 1.4 states that quantum witnesses of logarithmic size do not increase the power of logarithmic-space unitary quantum computations at all, and indeed extends the result of Marriott and Watrous to logarithmic-space case.

Space-efficient amplification for QMA. Let $\mathrm{QMA}[l_{\mathsf{V}}, l_{\mathsf{M}}](c,s)$ be the time-efficient version of $\mathrm{QMA}_{\mathbf{U}}\mathrm{SPACE}[l_{\mathsf{V}}, l_{\mathsf{M}}](c,s)$, i.e., the class of problems having standard polynomial-time QMA proof systems with completeness c and soundness s in which a polynomial-time unitary quantum verifier receives a quantum witness of $l_{\mathsf{M}}(n)$ qubits and uses workspace of $l_{\mathsf{V}}(n)$ qubits on every input of length n. As the reduction is in time polynomial with respect to $\frac{p}{c-s}$ in the proof of Theorem 1.1, the following amplifiability is immediate from Theorem 1.1 by taking functions p, l_{V}, and l_{M} to be polynomially bounded, and functions c and s to be polynomial-time computable and to satisfy $c - s \geq 1/q$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$.

▶ **Corollary 1.5.** *For any polynomially bounded functions $p, l_{\mathsf{V}}, l_{\mathsf{M}}\colon \mathbb{Z}^+ \to \mathbb{N}$ and for any polynomial-time computable functions $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c - s \geq 1/q$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$, there exists a function $\delta\colon \mathbb{Z}^+ \to \mathbb{N}$ that is logarithmic with respect to $\frac{p}{c-s}$ such that*

$$\mathrm{QMA}[l_{\mathsf{V}}, l_{\mathsf{M}}](c,s) \subseteq \mathrm{QMA}[l_{\mathsf{V}} + \delta, l_{\mathsf{M}}](1-2^{-p}, 2^{-p}).$$

Recall that the Marriott-Watrous amplification [13] requires δ to be in $O\big(\frac{p}{(c-s)^2}\big)$ and the phase-estimation-based method by Nagaj, Wocjan, and Zhang [15] requires δ to be in $O\big(p \log \frac{1}{c-s}\big)$, instead of δ in $O\big(\log \frac{p}{c-s}\big)$ of Corollary 1.5. Hence, the method in this paper is most space-efficient among known error-reduction methods for standard QMA proof systems, and also among those for BQP.

Strong amplification for unitary QMAPSPACE. Let $\mathrm{Q_U PSPACE}(c,s)$ denote the class of problems solvable by polynomial-space unitary quantum computations with completeness c and soundness s, and let $\mathrm{QMA_U PSPACE}(c,s)$ denote the class of problems having polynomial-space unitary QMA proof systems (i.e., such systems in which a verifier performs a polynomial-space unitary computation upon receiving a polynomial-size quantum witness) with completeness c and soundness s. The following corollary states the scaled-up versions of Corollaries 1.2 and 1.3, and again is immediate from Theorem 1.1 by taking a function p to be polynomial-space computable and exponentially bounded, functions c and s to be polynomial-space computable and to satisfy $c - s \geq 2^{-q}$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$, and functions l_{V} and l_{M} to be polynomially bounded (or a function $l_{\mathsf{M}} = 0$ in the case of $\mathrm{Q_U PSPACE}(c,s)$).

▶ **Corollary 1.6.** *For any polynomially bounded function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ and for any polynomial-space computable functions $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c - s \geq 2^{-q}$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$,*

$$\mathrm{Q_U PSPACE}(c,s) \subseteq \mathrm{Q_U PSPACE}\big(1 - 2^{-2^p}, 2^{-2^p}\big), \text{ and}$$
$$\mathrm{QMA_U PSPACE}(c,s) \subseteq \mathrm{QMA_U PSPACE}\big(1 - 2^{-2^p}, 2^{-2^p}\big).$$

Again by a standard technique of replacing a quantum witness by a totally mixed state as a self-prepared witness, the following corollary follows from Corollary 1.6 together with the fact that RevPSPACE = PrQPSPACE = PSPACE [2, 21], where RevPSPACE and PrQPSPACE are the complexity classes corresponding to deterministic polynomial-space reversible computations and unbounded-error polynomial-space quantum computations, respectively.

▶ **Corollary 1.7.** *For any polynomial-space computable functions $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c - s \geq 2^{-q}$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$,*

$$\mathrm{QMA_U PSPACE}(c,s) = \mathrm{PSPACE}.$$

Now the PSPACE upper bound immediately follows for the class of problems having standard polynomial-time QMA proof systems with exponentially small completeness-soundness gap. More precisely, for the class $\mathrm{QMA}(c,s)$ of problems having standard polynomial-time QMA proof systems with completeness c and soundness s, the following corollary holds.

▶ **Corollary 1.8.** *For any polynomially bounded function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ and for any polynomial-time computable functions $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c - s \geq 2^{-q}$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$,*

$$\mathrm{QMA}(c,s) \subseteq \mathrm{PSPACE}.$$

For QMA proof systems with exponentially small completeness-soundness gap, the PSPACE upper bound was known previously only for the one-sided-error case (following from the result in Ref. [7]), and only the EXP upper bound was known for the two-sided-error case (following from the result in Ref. [10]). Natarajan and Wu [16] independently proved a statement equivalent to Corollary 1.8. In fact, statements equivalent to Corollary 1.8 were also proved with different proofs independently by the first and third authors of the present paper in Ref. [4] (see Ref. [5] also) and by the complement subset of the present authors. The first and third authors of the present paper further proved in Refs. [4, 5] that the converse of Corollary 1.8 also holds, i.e., PSPACE is characterized by QMA proof systems with exponentially small completeness-soundness gap.

Strong amplification for matchgate computations. A matchgate is defined to be a two-qubit gate of the form $G(A, B)$ corresponding to the four-by-four unitary matrix in which the four corner elements form A and the four inner-square elements form B for matrices A and B in SU(2), and all the other elements are 0. A matchgate circuit is a quantum circuit such that: (i) the input state is a computational basis state, (ii) all the gates of the circuit are matchgates which are applied to two neighbor qubits, and (iii) the output is a final measurement in the computational basis on any single qubit. Matchgate computations were introduced and proved classically simulable by Valiant [20]. Terhal and DiVincenzo [19] related them to noninteracting-fermion quantum circuits. Let $\mathrm{MG}(c, s)$ denote the class of problems solvable by polynomial-time matchgate computations with completeness c and soundness s. Using the equivalence of polynomial-time matchgate computations and logarithmic-space unitary computations shown by Jozsa, Kraus, Miyake, and Watrous [9, Corollary 3.3], the following is immediate from Corollary 1.2.

▶ **Corollary 1.9.** *For any polynomially bounded function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ that is logarithmic-space computable and for any logarithmic-space computable functions $c, s\colon \mathbb{Z}^+ \to [0, 1]$ satisfying $c - s \geq 1/q$ for some polynomially bounded function $q\colon \mathbb{Z}^+ \to \mathbb{N}$,*

$$\mathrm{MG}(c, s) \subseteq \mathrm{MG}(1 - 2^{-p}, 2^{-p}).$$

2 Overview of the proof of main theorem

We assume familiarity with basic quantum formalism (see Refs. [17, 11, 26], for instance). The main theorem can be proved with three different proofs. Due to space limitations, this version presents only one of the three proofs. The other two proofs, as well as precise definitions and technical proofs, are deferred to the full version [3].

Consider any unitary transformation V_x of the verifier on input x, and let p_{acc} be the maximum acceptance probability of it (and thus, $p_{\mathrm{acc}} \geq c(|x|)$ for yes instances, and $p_{\mathrm{acc}} \leq s(|x|)$ for no instances). Then the idea is to guess p_{acc} with *mild* precision $\delta = 2^{-l(|x|)}$, where $\frac{c-s}{2\sqrt{6q}} < 2^{-l} \leq \frac{c-s}{\sqrt{6q}}$ for some appropriately chosen function q and the (integer-valued) function l determined uniquely by given c, s, and q.

For each j in $\{1, \ldots, 2^{l(|x|)}\}$, let $r_j = j\delta$ be a possible guess of p_{acc}. Pick an integer k from $\{1, \ldots, 2^{l(|x|)}\}$ uniformly at random, and reject immediately if $r_k = k\delta < c(|x|)$ (so that no k can result in a good guess at p_{acc} for no instances). Otherwise r_k is used as a guess at p_{acc}. The point is that, for yes instances, there exists a choice of k such that $|r_k - p_{\mathrm{acc}}| < \delta \leq \frac{c(|x|)-s(|x|)}{\sqrt{6\,q(|x|)}}$, while for no instances, it holds that $|r_k - p_{\mathrm{acc}}| > c(|x|) - s(|x|)$ for any choice of k. Hence, by using the Reflection Procedure [12] combined with the additive adjustment of acceptance probability [8], the acceptance probability can be *mildly* amplified to at least $1 - \frac{(c(|x|)-s(|x|))^2}{6\,q(|x|)}$ in the yes-instance case, if the appropriate guess r_k is made. It is stressed that this mild amplification is the key for the efficiency in workspace. For no instances, the acceptance probability is at most $1 - \big(c(|x|) - s(|x|)\big)^2$ for any guess r_k.

Fix an index k of the guess r_k and let $V'_{x,k}$ be the unitary operator corresponding to the procedure constructed so far. Now repeat the following procedure $N(|x|)$ times for a function N defined by $N = \left\lceil \frac{q}{2(c-s)^2} \right\rceil$: One applies $V'_{x,k}$, and then increments a counter by 1 if the state corresponds to a rejection state of it. One further applies $(V'_{x,k})^\dagger$, and then increments a counter by 1 if any of the work qubits of $V'_{x,k}$ is in state $|1\rangle$. After the repetition, one accepts if and only if the counter value remains 0. In short, these repetitions essentially take the AND of the $N(|x|)$ attempts of $V'_{x,k}$ (with each initialization try by $(V'_{x,k})^\dagger$). The rigorous analysis shows that the initialization steps also contribute to taking AND, and this process is exactly

Additive Adjustment Procedure associated with (U, Δ, Π, l, k)

1. Prepare a single-qubit register B and an l-qubit register R, where all the qubits in B and R are initialized to state $|0\rangle$. Receive a quantum register Q that contains a state in the subspace corresponding to the projection Δ.
2. Apply the Hadamard transformation H to each qubit in (B, R), and apply U to Q.
3. Accept either if B contains 0 *and* the state in Q belongs to the subspace corresponding to Π or if B contains 1 *and* the content of R is greater than k (when viewed as an integer in $\{1, \ldots, 2^l\}$), and reject otherwise.

Figure 1 The Additive Adjustment Procedure.

equivalent to taking the AND of $2\,N(|x|)$ attempts of $V'_{x,k}$. The acceptance probability is at least $\frac{1}{2}$ for yes instances when the appropriate guess r_k at p_{acc} is made, while it is at most $e^{-\,q(|x|)} < 2^{-\,q(|x|)}$ for any guess r_k for no instances. Taking into account that the index k of r_k is chosen uniformly at random, this results in a unitary procedure V''_x with acceptance probability at least $2^{-\,l(|x|)} \cdot \frac{1}{2} > \frac{c(|x|)-s(|x|)}{4\sqrt{6\,q(|x|)}}$ for yes instances and at most $2^{-\,q(|x|)}$ for no instances.

Finally, by using a repetition similar to above based on V''_x that takes OR instead of AND, it is clear that the completeness acceptance probability becomes exponentially close to 1 with respect to q, while the soundness acceptance probability is still exponentially small with respect to q. To achieve error below 2^{-p} for a target p, one chooses q to be slightly larger than p when constructing V''_x (more precisely, one can choose a function $q = \lceil 2\big(p + \log \frac{6p}{c-s} + 1\big)\rceil$).

3 Basic procedures

Let $\Sigma = \{0, 1\}$ denote the binary alphabet set. For every positive integer n, let $\mathbb{C}(\Sigma^n)$ denote the 2^n-dimensional complex Hilbert space whose standard basis vectors are indexed by the elements in Σ^n. In this paper, all Hilbert spaces are complex and of dimension a power of two. A quantum register is a set of single or multiple qubits. For a quantum register R, let I_{R} denote the identity operator over the Hilbert space associated with R.

Let $\mathcal{H}$ be any Hilbert space of dimension a power of two. Given a unitary transformation U and two projections Δ and Π, all acting over $\mathcal{H}$, define the Hermitian operator M over $\mathcal{H}$ by

$$M = \Delta U^\dagger \Pi U \Delta,$$

which plays crucial roles in many well-known amplification methods in quantum computation, including the Grover search [6], the Marriott-Watrous amplification for QMA [13], and quantum rewinding for zero-knowledge proofs against quantum attacks [25].

Additive Adjustment Procedure. Consider the procedure described in Figure 1, called the Additive Adjustment Procedure, which uses the additive adjustment technique of acceptance probability proposed in Ref. [8].

The following properties hold with the Additive Adjustment Procedure.

▶ **Proposition 3.1.** *Let U be a unitary transformation and Δ and Π be projections, all acting over the same Hilbert space. Consider the Hermitian operator $M = \Delta U^\dagger \Pi U \Delta$. For any positive integer l and any integer k in $\{1, \ldots, 2^l\}$, the following two properties hold:*

(Completeness) *Suppose that M has an eigenstate $|\phi_\lambda\rangle$ with its associated eigenvalue λ. Then, the* Additive Adjustment Procedure *associated with (U, Δ, Π, l, k) results in acceptance with probability $\frac{1}{2} + \frac{1}{2}\big(\lambda - \frac{k}{2^l}\big)$ when $|\phi_\lambda\rangle$ is received in register Q in Step 1.*

Reflection Procedure associated with (U, Δ, Π)

1. Receive a quantum register Q that contains a state in the subspace corresponding to the projection Δ.
2. Apply U to Q.
3. Perform a phase-flip (i.e., multiply the phase by -1) if the state in Q belongs to the subspace corresponding to the projection Π. That is, apply the unitary transformation $I_{\mathsf{Q}} - 2\Pi$ to Q.
4. Apply $U^\dagger$ to Q.
5. Reject if the state in Q belongs to the subspace corresponding to Δ, and accept otherwise.

Figure 2 The Reflection Procedure.

(Soundness) *Suppose that all the eigenvalues of M are at most ε for some ε in $[0,1)$. Then, the* Additive Adjustment Procedure *associated with (U, Δ, Π, l, k) results in acceptance with probability at most $\frac{1}{2} + \frac{1}{2}\left(\varepsilon - \frac{k}{2^l}\right)$ regardless of the quantum state received in register Q in Step 1.*

Reflection Procedure. Now consider the procedure described in Figure 2, which is exactly the Reflection Procedure in a general form originally developed in Ref. [12].

The following proposition holds with the Reflection Procedure.

▶ **Proposition 3.2** ([12])**.** *Let U be a unitary transformation and Δ and Π be projections, all acting over the same Hilbert space. Consider the Hermitian operator $M = \Delta U^\dagger \Pi U \Delta$. The following two properties hold:*

(Completeness) *Suppose that M has an eigenstate $|\phi_\lambda\rangle$ with its associated eigenvalue λ. Then, the* Reflection Procedure *associated with (U, Δ, Π) results in acceptance with probability $4\lambda(1-\lambda)$ when $|\phi_\lambda\rangle$ is received in register Q in Step 1.*

(Soundness) *Suppose that none of the eigenvalues of M is in the interval $\left(\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon\right)$ for some ε in $\left(0, \frac{1}{2}\right]$. Then, the* Reflection Procedure *associated with (U, Δ, Π) results in acceptance with probability at most $1 - 4\varepsilon^2$ regardless of the quantum state received in register Q in Step 1.*

AND-Type and OR-Type Repetition Procedures. Given a unitary transformation U and two projections Δ and Π all acting over a Hilbert space, consider the process of applying U to a fixed initial state $|\phi\rangle$ in a quantum register Q that is in the subspace corresponding to Δ and then accepting if and only if the resulting state is projected onto the subspace corresponding to Π by the projective measurement $\{\Pi, I_{\mathsf{Q}} - \Pi\}$. Let p denote the acceptance probability of this process. By running N independent attempts of such a process, the probability clearly becomes p^N for the event that all the attempts result in acceptance, but which requires N copies of the initial state $|\phi\rangle$. When $|\phi\rangle$ is an eigenstate of the Hermitian operator $M = \Delta U^\dagger \Pi U \Delta$, the AND-Type Repetition Procedure described in Figure 3 essentially simulates such independent attempts with just one copy of $|\phi\rangle$.

The following proposition holds with the AND-Type Repetition Procedure.

▶ **Proposition 3.3.** *Let U be a unitary transformation and Δ and Π be projections, all acting over the same Hilbert space, and let N be a positive integer. Consider the Hermitian operator $M = \Delta U^\dagger \Pi U \Delta$. The following two properties hold:*

AND-Type Repetition Procedure associated with (U, Δ, Π, N)

1. Let $l = \lceil \log(2N+1) \rceil$, and prepare an l-qubit register C, where all the qubits in C are initialized to state $|0\rangle$. Receive a quantum register Q that contains a state in the subspace corresponding to the projection Δ.
2. For $j = 1$ to N, perform the following:
   - 2.1. Apply U to Q.
   - 2.2. If the state in Q belongs to the subspace corresponding to the projection $I_{\mathsf{Q}} - \Pi$, apply $U_{+1}(\mathbb{Z}_{2^l})$ to C, where $U_{+1}(\mathbb{Z}_{2^l})$ is the unitary transformation defined by
   $$U_{+1}(\mathbb{Z}_{2^l}) \colon |j\rangle \mapsto \left|(j+1) \bmod 2^l\right\rangle, \quad \forall j \in \mathbb{Z}_{2^l}.$$
   - 2.3. Apply $U^\dagger$ to Q.
   - 2.4. If the state in Q belongs to the subspace corresponding to the projection $I_{\mathsf{Q}} - \Delta$, apply $U_{+1}(\mathbb{Z}_{2^l})$ to C.
3. Accept if the content of C is 0 (i.e., all the qubits in C are in state $|0\rangle$), and reject otherwise.

Figure 3 The AND-Type Repetition Procedure.

(Completeness) *Suppose that M has an eigenstate $|\phi_\lambda\rangle$ with its associated eigenvalue λ. Then, the* AND-Type Repetition Procedure *associated with (U, Δ, Π, N) results in acceptance with probability λ^{2N} when $|\phi_\lambda\rangle$ is received in register Q in Step 1.*

(Soundness) *Suppose that all the eigenvalues of M are at most ε for some ε in $[0, 1)$. Then, the* AND-Type Repetition Procedure *associated with (U, Δ, Π, N) results in acceptance with probability at most ε^{2N} regardless of the quantum state received in register Q in Step 1.*

One can also construct a procedure that essentially simulates the process of taking OR of the N independent attempts mentioned before with just one copy of $|\phi\rangle$. One now applies $U_{+1}(\mathbb{Z}_{2^l})$ to C when the state in Q belongs to the subspace corresponding to the projection Π at Step 2.2, and *rejects* if and only if the content of C is 0 at Step 3. The resulting procedure is called the OR-Type Repetition Procedure, which has the following properties.

▶ **Proposition 3.4.** *Let U be a unitary transformation and Δ and Π be projections, all acting over the same Hilbert space, and let N be a positive integer. Consider the Hermitian operator $M = \Delta U^\dagger \Pi U \Delta$. The following two properties hold:*

(Completeness) *Suppose that M has an eigenstate $|\phi_\lambda\rangle$ with its associated eigenvalue λ. Then, the* OR-Type Repetition Procedure *associated with (U, Δ, Π, N) results in acceptance with probability $1 - (1-\lambda)^{2N}$ when $|\phi_\lambda\rangle$ is received in register Q in Step 1.*

(Soundness) *Suppose that all the eigenvalues of M are at most ε for some ε in $[0, 1)$. Then, the* OR-Type Repetition Procedure *associated with (U, Δ, Π, N) results in acceptance with probability at most $1 - (1-\varepsilon)^{2N}$ regardless of the quantum state received in register Q in Step 1.*

4 Guess-based amplification framework

Consider any QMA-type computation for a problem $A = (A_{\text{yes}}, A_{\text{no}})$ induced by a family $\{V_x\}_{x \in \Sigma^*}$ of a unitary transformation V_x of the verifier on input x in Σ^* that acts over a quantum register $\mathsf{Q} = (\mathsf{V}, \mathsf{M})$, where V is the quantum register consisting of all the private qubits of the verifier, and M is the one for storing a received quantum witness. Let Π_{init} be the projection onto the subspace spanned by the legal initial states of the QMA-type

Mild Completeness Amplification with Guess k associated with (V_x, p)

Define functions l and C by $l = \left\lceil \frac{1}{2} \log \frac{p}{(c-s)^2} \right\rceil$ and $C = \lceil 2^l c \rceil$. Let Π_{init} and Π_{acc} be the projections onto the subspaces spanned by the legal initial states and the accepting states, respectively, in the verification with V_x. Given an integer k in $\{1, \ldots, 2^{l(|x|)}\}$ as a guess, consider the Additive Adjustment Procedure associated with $(V_x, \Pi_{\text{init}}, \Pi_{\text{acc}}, l(|x|), k)$. Let $V'_{x,k}$ be the unitary transformation induced by it, let Π'_{init} be the projection onto the subspace spanned by the legal initial states of it, and let $\Pi'_{\text{acc},k}$ be the projection onto the subspace spanned by the accepting states of it.
Reject if $k < C(|x|)$, and continue otherwise by performing the Reflection Procedure associated with $\left(V'_{x,k}, \Pi'_{\text{init}}, \Pi'_{\text{acc},k}\right)$.

Figure 4 The Mild Completeness Amplification with Guess k.

computation induced by V_x (i.e., the subspace spanned by those in which all the qubits in V is in state $|0\rangle$) and let Π_{acc} be the projection onto the subspace spanned by the accepting states of the QMA-type computation associated with V_x (i.e., the subspace spanned by states for which the designated output qubit of V_x is in state $|0\rangle$).

The maximum eigenvalue of the Hermitian operator $M_x = \Pi_{\text{init}} V_x^\dagger \Pi_{\text{acc}} V_x \Pi_{\text{init}}$ exactly corresponds to the maximum acceptance probability of the verifier on input x over all possible quantum witnesses received in M. Hence, M_x has an eigenvalue at least $c(|x|)$ if x is in A_{yes}, while all eigenvalues of M_x are at most $s(|x|)$ if x is in A_{no}, where $c, s \colon \mathbb{Z}^+ \to [0,1]$ are functions that provide completeness and soundness conditions of the QMA-type computation induced by $\{V_x\}_{x \in \Sigma^*}$, respectively.

Mild completeness amplification with a guess. Fix arbitrarily a function $p \colon \mathbb{Z}^+ \to \mathbb{N}$ and functions $c, s \colon \mathbb{Z}^+ \to [0,1]$ satisfying $c > s$, and let $l, C \colon \mathbb{Z}^+ \to \mathbb{N}$ be functions defined by $l = \left\lceil \frac{1}{2} \log \frac{p}{(c-s)^2} \right\rceil$ and $C = \lceil 2^l c \rceil$. Fix an input x and an integer k in $\{1, \ldots, 2^{l(|x|)}\}$. Given the triplet $(V_x, \Pi_{\text{init}}, \Pi_{\text{acc}})$ and an integer k, one first constructs the Additive Adjustment Procedure associated with $\left(V_x, \Pi_{\text{init}}, \Pi_{\text{acc}}, l(|x|), k\right)$, if k is at least $C(|x|)$ (and automatically rejects otherwise so that no k can result in a good guess at the acceptance probability when the actual value of it is unallowably small). Let $V'_{x,k}$ be the unitary transformation induced by it, let Π'_{init} be the projection onto the subspace spanned by the legal initial states of it, and let $\Pi'_{\text{acc},k}$ be the projection onto the subspace spanned by the accepting states of it. Next, from the triplet $\left(V'_{x,k}, \Pi'_{\text{init}}, \Pi'_{\text{acc},k}\right)$, one constructs the Reflection Procedure associated with $\left(V'_{x,k}, \Pi'_{\text{init}}, \Pi'_{\text{acc},k}\right)$, and performs it. The resulting procedure is called the Mild Completeness Amplification with Guess k, and is summarized in Figure 4.

From the properties of the Additive Adjustment Procedure and the Reflection Procedure (Propositions 3.1 and 3.2), one can show the following lemma.

▶ **Lemma 4.1.** *Given functions $l_{\mathsf{V}}, l_{\mathsf{M}} \colon \mathbb{Z}^+ \to \mathbb{N}$ and $c, s \colon \mathbb{Z}^+ \to [0,1]$ satisfying $c > s$, let $A = (A_{\text{yes}}, A_{\text{no}})$ be a problem in $\mathrm{QMA}_{\mathbf{U}}\mathrm{SPACE}[l_{\mathsf{V}}, l_{\mathsf{M}}](c,s)$, and let $V = \{V_x\}_{x \in \Sigma^*}$ be the $(l_{\mathsf{V}}, l_{\mathsf{M}})$-space-bounded quantum verifier witnessing this membership. Then, for any function $p \colon \mathbb{Z}^+ \to \mathbb{N}$ and for every x in Σ^*, letting $l = \left\lceil \frac{1}{2} \log \frac{p}{(c-s)^2} \right\rceil$,*

(Completeness) *if x is in A_{yes}, there exists an integer k in $\{1, \ldots, 2^{l(|x|)}\}$ as a guess such that the Mild Completeness Amplification with Guess k associated with (V_x, p) results in acceptance with probability at least $1 - \frac{(c(|x|) - s(|x|))^2}{p(|x|)}$, and*

(Soundness) *if x is in A_{no}, for any integer k in $\{1, \ldots, 2^{l(|x|)}\}$ as a guess, the Mild Completeness Amplification with Guess k associated with (V_x, p) results in acceptance with probability at most $1 - \left(c(|x|) - s(|x|)\right)^2$.*

Soundness Error Reduction with Guess k associated with (V_x, p)

Define functions l and N by $l = \left\lceil \frac{1}{2} \log \frac{6p}{(c-s)^2} \right\rceil$ and $N = \left\lceil \frac{p}{2(c-s)^2} \right\rceil$. Given an integer k in $\{1, \ldots, 2^{l(|x|)}\}$, consider the Mild Completeness Amplification with Guess k associated with $(V_x, 6p)$. Let $V'_{x,k}$ be the unitary transformation induced by it, let Π'_{init} be the projection onto the subspace spanned by the legal initial states of it, and let $\Pi'_{\text{acc},k}$ be the projection onto the subspace spanned by the accepting states of it.
Perform the AND-Type Repetition Procedure associated with $\left(V'_{x,k}, \Pi'_{\text{init}}, \Pi'_{\text{acc},k}, N(|x|)\right)$.

Figure 5 The Soundness Error Reduction with Guess k.

Soundness error reduction with a guess. Again fix arbitrarily a function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ and functions $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c > s$, and let $l, N\colon \mathbb{Z}^+ \to \mathbb{N}$ be functions defined by $l = \left\lceil \frac{1}{2} \log \frac{6p}{(c-s)^2} \right\rceil$ and $N = \left\lceil \frac{p}{2(c-s)^2} \right\rceil$. Fix an input x and an integer k in $\{1, \ldots, 2^{l(|x|)}\}$. Given the pair (V_x, p) and the integer k, consider the Mild Completeness Amplification with Guess k associated with $(V_x, 6p)$. As before, let $V'_{x,k}$ be the unitary transformation induced by it, let Π'_{init} be the projection onto the subspace spanned by the legal initial states of it, and let $\Pi'_{\text{acc},k}$ be the projection onto the subspace spanned by the accepting states of it. From the triplet $\left(V'_{x,k}, \Pi'_{\text{init}}, \Pi'_{\text{acc},k}\right)$ and a positive integer $N(|x|)$, one constructs the AND-Type Repetition Procedure associated with $\left(V'_{x,k}, \Pi'_{\text{init}}, \Pi'_{\text{acc},k}, N(|x|)\right)$, and performs it. The resulting procedure is called the Soundness Error Reduction with Guess k, and is summarized in Figure 5.

From the properties of the AND-Type Repetition Procedure and the Mild Completeness Amplification with Guess k (Proposition 3.3 and Lemma 4.1), one can show the following lemma.

▶ **Lemma 4.2.** *Given functions $l_{\mathsf{V}}, l_{\mathsf{M}}\colon \mathbb{Z}^+ \to \mathbb{N}$ and $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c > s$, let $A = (A_{\text{yes}}, A_{\text{no}})$ be a problem in $\mathrm{QMA}_{\mathbf{U}}\mathrm{SPACE}[l_{\mathsf{V}}, l_{\mathsf{M}}](c,s)$, and let $V = \{V_x\}_{x \in \Sigma^*}$ be the $(l_{\mathsf{V}}, l_{\mathsf{M}})$-space-bounded quantum verifier witnessing this membership. Then, for any function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ and for every x in Σ^*, letting $l = \left\lceil \frac{1}{2} \log \frac{6p}{(c-s)^2} \right\rceil$,*

(Completeness) *if x is in A_{yes}, there exists an integer k in $\{1, \ldots, 2^{l(|x|)}\}$ as a guess such that the Soundness Error Reduction with Guess k associated with (V_x, p) results in acceptance with probability at least $\frac{1}{2}$, and*

(Soundness) *if x is in A_{no}, for any integer k in $\{1, \ldots, 2^{l(|x|)}\}$ as a guess, the Soundness Error Reduction with Guess k associated with (V_x, p) results in acceptance with probability at most $2^{-p(|x|)}$.*

Soundness error reduction with a random guess. Again fix arbitrarily a function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ and functions $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c > s$, and let $l\colon \mathbb{Z}^+ \to \mathbb{N}$ be a function defined by $l = \left\lceil \frac{1}{2} \log \frac{6p}{(c-s)^2} \right\rceil$. Fix an input x. Given the pair (V_x, p), consider choosing an integer k from $\{1, \ldots, 2^{l(|x|)}\}$ uniformly at random, and performing the Soundness Error Reduction with Guess k associated with (V_x, p). The resulting procedure is called the Soundness Error Reduction with Random Guess and is summarized in Figure 6.

Lemma 4.3 below follows from the Soundness Error Reduction with Random Guess together with the properties of the Soundness Error Reduction with Guess k stated in Lemma 4.2.

▶ **Lemma 4.3.** *For any functions $p, l_{\mathsf{V}}, l_{\mathsf{M}}\colon \mathbb{Z}^+ \to \mathbb{N}$ and any functions $c, s\colon \mathbb{Z}^+ \to [0,1]$ satisfying $c > s$ and $\frac{c-s}{4\sqrt{6p}} > 2^{-p}$, there exists a function $\delta\colon \mathbb{Z}^+ \to \mathbb{N}$ that is logarithmic with*

Soundness Error Reduction with Random Guess associated with (V_x, p)

Define a function l by $l = \left\lceil \frac{1}{2} \log \frac{6p}{(c-s)^2} \right\rceil$.
Pick an integer k from $\{1, \ldots, 2^{l(|x|)}\}$ uniformly at random and perform the Soundness Error Reduction with Guess k associated with (V_x, p).

Figure 6 The Soundness Error Reduction with Random Guess.

Space-Efficient Amplification Based on Random Guess associated with (V_x, p)

Define functions q and N by $q = \left\lceil 2\left(p + \log \frac{6p}{c-s} + 1\right) \right\rceil$ and $N = \left\lceil \frac{2\sqrt{6q}}{c-s} \cdot p \right\rceil$. Consider the Soundness Error Reduction with Random Guess associated with (V_x, q). Let V'_x be the unitary transformation induced by it, let Π'_{init} be the projection onto the subspace spanned by the legal initial states of it, and let Π'_{acc} be the projection onto the subspace spanned by the accepting states of it.
Perform the OR-Type Repetition Procedure associated with $\left(V'_x, \Pi'_{\text{init}}, \Pi'_{\text{acc}}, N(|x|)\right)$.

Figure 7 The Space-Efficient Amplification Based on Random Guess.

respect to $\frac{p}{c-s}$ *such that*

$$\mathrm{QMA}_{\mathbf{U}}\mathrm{SPACE}[l_{\mathsf{V}}, l_{\mathsf{M}}](c, s) \subseteq \mathrm{QMA}_{\mathbf{U}}\mathrm{SPACE}[l_{\mathsf{V}} + \delta, l_{\mathsf{M}}]\left(\frac{c-s}{4\sqrt{6p}}, 2^{-p}\right).$$

Space-efficient amplification based on a random guess. Again fix a function $p\colon \mathbb{Z}^+ \to \mathbb{N}$ and functions $c, s\colon \mathbb{Z}^+ \to [0, 1]$ satisfying $c > s$ arbitrarily. Let $q, N\colon \mathbb{Z}^+ \to \mathbb{N}$ be functions defined by $q = \left\lceil 2\left(p + \log \frac{6p}{c-s} + 1\right) \right\rceil$ and $N = \left\lceil \frac{2\sqrt{6q}}{c-s} \cdot p \right\rceil$. Fix an input x. Given the pair (V_x, p), consider the Soundness Error Reduction with Random Guess associated with (V_x, q). Let V'_x be the unitary transformation induced by it, let Π'_{init} be the projection onto the subspace spanned by the legal initial states of it, and let Π'_{acc} be the projection onto the subspace spanned by the accepting states of it. From the triplet $\left(V'_x, \Pi'_{\text{init}}, \Pi'_{\text{acc}}\right)$ and a positive integer $N(|x|)$, one constructs the OR-Type Repetition Procedure associated with $\left(V'_x, \Pi'_{\text{init}}, \Pi'_{\text{acc}}, N(|x|)\right)$, and performs it. The resulting procedure is called the Space-Efficient Amplification Based on Random Guess and is summarized in Figure 7.

Now Theorem 1.1, the main theorem of this paper, is proved by using the Space-Efficient Amplification Based on Random Guess combined with the properties of the Soundness Error Reduction with Random Guess and the OR-Type Repetition Procedure stated in Lemma 4.3 and Proposition 3.4, respectively.

Acknowledgements. BF and CYL are supported by the Department of Defense. HK and HN are supported by the Grant-in-Aid for Scientific Research (A) Nos. 24240001 and 16H01705 of the Japan Society for the Promotion of Science. TM is supported by the Program to Disseminate Tenure Tracking System of the Ministry of Education, Culture, Sports, Science and Technology in Japan, the Grant-in-Aid for Scientific Research on Innovative Areas No. 15H00850 of the Ministry of Education, Culture, Sports, Science and Technology in Japan, and the Grant-in-Aid for Young Scientists (B) No. 26730003 of the Japan Society for the Promotion of Science. HN is also supported by the Grant-in-Aid for Scientific Research on Innovative Areas No. 24106009 of the Ministry of Education, Culture, Sports, Science and Technology in Japan, which HK is also grateful to. HN further acknowledges support from the Grant-in-Aid for Scientific Research (C) No. 25330012 of the Japan Society for the Promotion of Science.

References

1 Andris Ambainis and Rūsiņš Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *39th Annual Symposium on Foundations of Computer Science*, pages 332–341, 1998. doi:10.1109/SFCS.1998.743469.

2 Charles H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989. doi:10.1137/0218053.

3 Bill Fefferman, Hirotada Kobayashi, Cedric Yen-Yu Lin, Tomoyuki Morimae, and Harumichi Nishimura. Space-efficient error reduction for unitary quantum computations. arXiv.org e-Print archive, arXiv:1604.08192 [quant-ph], 2016. arXiv:1604.08192.

4 Bill Fefferman and Cedric Lin. Quantum Merlin Arthur with exponentially small gap. arXiv.org e-Print archive, arXiv:1601.01975 [quant-ph], 2016. arXiv:1601.01975.

5 Bill Fefferman and Cedric Yen-Yu Lin. A complete characterization of unitary quantum space. arXiv.org e-Print archive, arXiv:1604.01384 [quant-ph], 2016. arXiv:1604.01384.

6 Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 212–219, 1996. doi:10.1145/237814.237866.

7 Tsuyoshi Ito, Hirotada Kobayashi, and John Watrous. Quantum interactive proofs with weak error bounds. In *ITCS'12, Proceedings of the 2012 ACM Conference on Innovations in Theoretical Computer Science*, pages 266–275, 2012. doi:10.1145/2090236.2090259.

8 Stephen P. Jordan, Hirotada Kobayashi, Daniel Nagaj, and Harumichi Nishimura. Achieving perfect completeness in classical-witness quantum Merlin-Arthur proof systems. *Quantum Information and Computation*, 12(5–6):0461–0471, 2012.

9 Richard Jozsa, Barbara Kraus, Akimasa Miyake, and John Watrous. Matchgate and space-bounded quantum computations are equivalent. *Proceedings of the Royal Society A*, 466(2115):809–830, 2010. doi:10.1098/rspa.2009.0433.

10 Alexei Kitaev and John Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 608–617, 2000. doi:10.1145/335305.335387.

11 Alexei Yu. Kitaev, Alexander H. Shen, and Mikhail N. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002. doi:10.1090/gsm/047.

12 Hirotada Kobayashi, François Le Gall, and Harumichi Nishimura. Stronger methods of making quantum interactive proofs perfectly complete. *SIAM Journal on Computing*, 44(2):243–289, 2015. doi:10.1137/140971944.

13 Chris Marriott and John Watrous. Quantum Arthur-Merlin games. *Computational Complexity*, 14(2):122–152, 2005. doi:10.1007/s00037-005-0194-x.

14 Dieter van Melkebeek and Thomas Watson. Time-space efficient simulations of quantum computations. *Theory of Computing*, 8:1–51 (Article 1), 2012. doi:10.4086/toc.2012.v008a001.

15 Daniel Nagaj, Pawel Wocjan, and Yong Zhang. Fast amplification of QMA. *Quantum Information and Computation*, 9(11–12):1053–1068, 2009.

16 Anand Natarajan and Xiaodi Wu. Private communication, January 2016.

17 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

18 Amnon Ta-Shma. Inverting well conditioned matrices in quantum logspace. In *STOC'13, Proceedings of the 2013 ACM Symposium on Theory of Computing*, pages 881–890, 2013. doi:10.1145/2488608.2488720.

19 Barbara M. Terhal and David P. DiVincenzo. Classical simulation of noninteracting-fermion quantum circuits. *Physical Review A*, 65:article 032325, 2002. doi:10.1103/PhysRevA.65.032325.

20 Leslie G. Valiant. Quantum circuits that can be simulated classically in polynomial time. *SIAM Journal on Computing*, 31(4):1229–1254, 2002. doi:10.1137/S0097539700377025.

21 John Watrous. Space-bounded quantum complexity. *Journal of Computer and System Sciences*, 59(2):281–326, 1999. doi:10.1006/jcss.1999.1655.

22 John Watrous. Quantum simulations of classical random walks and undirected graph connectivity. *Journal of Computer and System Sciences*, 62(2):376–391, 2001. doi:10.1006/jcss.2000.1732.

23 John Watrous. On the complexity of simulating space-bounded quantum computations. *Computational Complexity*, 12(1–2):48–84, 2003. doi:10.1007/s00037-003-0177-8.

24 John Watrous. Quantum computational complexity. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 7174–7201. Springer New York, 2009. doi:10.1007/978-0-387-30440-3_428.

25 John Watrous. Zero-knowledge against quantum attacks. *SIAM Journal on Computing*, 39(1):25–58, 2009. doi:10.1137/060670997.

26 Mark M. Wilde. *Quantum Information Theory*. Cambridge University Press, 2013. doi:10.1017/CBO9781139525343.

Linear Time Algorithm for Quantum 2SAT

Itai Arad[1], Miklos Santha[2], Aarthi Sundaram[3], and Shengyu Zhang[4]

1 **Centre for Quantum Technologies, National University of Singapore, Singapore**
arad.itai@fastmail.com
2 **Centre for Quantum Technologies, National University of Singapore, Singapore; and**
CNRS, IRIF, Université Paris Diderot 75205 Paris, France
miklos.santha@gmail.com
3 **Centre for Quantum Technologies, National University of Singapore, Singapore**
aarthims@gmail.com
4 **Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong**
syzhang@cse.cuhk.edu.hk

Abstract

A canonical result about satisfiability theory is that the 2-SAT problem can be solved in linear time, despite the NP-hardness of the 3-SAT problem. In the quantum 2-SAT problem, we are given a family of 2-qubit projectors Q_{ij} on a system of n qubits, and the task is to decide whether the Hamiltonian $H = \sum Q_{ij}$ has a 0-eigenvalue, or it is larger than $1/n^c$ for some $c = O(1)$. The problem is not only a natural extension of the classical 2-SAT problem to the quantum case, but is also equivalent to the problem of finding the ground state of 2-local frustration-free Hamiltonians of spin 1/2, a well-studied model believed to capture certain key properties in modern condensed matter physics. While Bravyi has shown that the quantum 2-SAT problem has a classical polynomial-time algorithm, the running time of his algorithm is $O(n^4)$. In this paper we give a classical algorithm with linear running time in the number of local projectors, therefore achieving the best possible complexity.

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

Keywords and phrases Quantum SAT, Davis-Putnam Procedure, Linear Time Algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.15

1 Introduction

Various formulations of the satisfiability problem of Boolean formulae arguably constitute the center piece of classical complexity theory. In particular, a great amount of attention has been paid to the SAT problem, in which we are given a formula in the form of a conjunction of *clauses*, where each clause is a disjunction of *literals* (variables or negated variables), and the task is to find a satisfying assignment if there is one, or prove that none exists when the formula is unsatisfiable. In the case of the k-SAT problem, where k is a positive integer, in each clause the number of literals is at most k. While k-SAT is an NP-complete problem [4, 12, 17] when $k \geq 3$, the 2-SAT problem is well-known to be efficiently solvable.

Polynomial time algorithms for 2-SAT come in various flavors. Let us suppose that the input formula has n variables and m clauses. The algorithm of Krom [15] based on the resolution principle and on transitive closure computation decides if the formula is satisfiable in time $O(n^3)$ and finds a satisfying assignment in time $O(n^4)$. The limited backtracking

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 15; pp. 15:1–15:14


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

technique of Even, Itai and Shamir [9] has linear time complexity in m, as well as the elegant procedure of Aspvall, Plass and Tarjan [1] based on computing strongly connected components in a graph. A particularly simple randomized procedure of complexity $O(n^2)$ is described by Papadimitriou [18].

For our purposes the Davis-Putnam procedure [6] is of singular importance. This is a resolution-principle based general SAT solving algorithm, which with its refinement due to Davis, Putnam, Logemann and Loveland [5], forms even today the basis for the most efficient SAT solvers. While on general SAT instances it works in exponential time, on 2-SAT formulae it is of polynomial complexity.

The high level description of the procedure for 2-SAT is relatively simple. Let us suppose that our formula ϕ contains only clauses with two literals. Pick an arbitrary unassigned variable x_i and assign $x_i = 0$. The formula is simplified: a clause $(\bar{x}_i \vee x_j)$ becomes true and therefore can be removed, and a clause $(x_i \vee x_j)$ forces $x_j = 1$. This can be, in turn, propagated to other clauses to further simplify the formula until a contradiction is found or no more propagation is possible. If no contradiction is found and the propagation stops with the simplified formula ϕ_0, then we recurse on the satisfiabilty of ϕ_0. Otherwise, when a contradiction is found, that is at some point the propagation assigns two different values to the same variable, we reverse the choice made for x_i, and propagate the new choice $x_i = 1$. If this also leads to contradiction we declare ϕ unsatisfiable, otherwise we recurse on the result of this propagation, the simplified formula ϕ_1.

There is a deep and profound link between k-SAT formulas and k-local Hamiltonians, the central objects of condensed matter physics. A k-local Hamiltonian on n qubits is a Hermitian operator of the form $H = \sum_{i=1}^{m} h_i$, where each h_i is by itself a Hermitian operator acting non-trivially on at most k qubits. Local Hamiltonians model the local interactions between quantum spins. Of central importance is the minimal eigenstate of the Hamiltonian, known as the *ground state*, and its associated eigenvalue, known as the *ground energy*. The ground state governs much of the low temperature physics of the system, such as quantum phase transitions and collective quantum phenomena [19, 20]. Finding the ground state of a local Hamiltonian shares important similarities with the k-SAT problem: in both problems we are trying to find a global minimum of a set of local constraints. This connection with complexity theory is of physical significance. Indeed, with the advent of quantum information theory and quantum complexity theories, it has become clear that the complexity of finding the ground state and its energy is intimately related to its entanglement structure. In recent years, much attention has been devoted into understanding this structure, revealing a rich an intricate behaviour such as area laws [8] and topological order [13].

The connection between classical k-SAT and quantum local Hamiltonian was formalized by Kitaev [14] who introduced the k-local Hamiltonian problem: one is given a k-local Hamiltonian H, along with two constants $a < b$ such that $b - a > 1/n^{\alpha}$ for some constant α. It is promised that the ground energy of H is at most a (the YES case) or is at least b (the NO case), and the task is to decide which case holds. Broadly speaking, given a quantum state $|\psi\rangle$, the energy of a local term $\langle\psi|h_i|\psi\rangle$ is a measure of how much $|\psi\rangle$ "violates" h_i, hence the ground energy is the quantum analog of the minimal number of violations in a classical k-SAT. Therefore, in spirit, the k-local Hamiltonian problem corresponds to MAX-k-SAT, and indeed Kitaev has shown [14] that the 5-local Hamiltonian problem is QMA-complete, where the complexity class QMA is the quantum analogue of classical class MA, the probabilistic version of NP.

The problem *quantum* k-SAT, the quantum analogue of k-SAT, is a close relative of the k-local Hamiltonian problem. Here we are given a k-local Hamiltonian that is made of k-local

projectors, $H = \sum_{i=1}^{m} Q_i$, and we are asked whether the ground energy is 0 or it is larger than $b = 1/n^{\alpha}$ for some constant α. Notice that in the YES case, the energy of all projectors at the ground state is necessarily 0, since by definition, projectors are non-negative operators. Classically, this corresponds to a perfectly satisfiable formula. Physically, this is an example of a *frustration-free* Hamiltonian, in which the global ground state is also a ground state of every local term. Bravyi [2] has shown that quantum k-SAT was QMA_1-complete for $k \geq 4$, where QMA_1 stands for QMA with one-sided error (that is on YES instances the verifier accepts with probability 1). The QMA_1-completeness of quantum 3-SAT was recently proven by Nagaj [10].

This paper is concerned with the quantum 2-SAT problem, which we will also denote simply by 2QSAT. One major result concerning this problem is due to Bravyi [2], who has proven that it belongs to the complexity class P. More precisely, he has proven that 2QSAT can be decided by a deterministic algorithm in time $O(n^4)$, together with a ground state that has a polynomial classical description. In the case of 2QSAT, the Hamiltonian is given as a sum of 2-qubits projectors; each projector is defined on a 4-dimensional Hilbert space and can therefore be of rank 1, 2 or 3. In this paper, we give an algorithm for 2QSAT of *linear* complexity.

▶ **Theorem 1.** *There is a deterministic algorithm for* 2QSAT *whose running time is* $O(n+m)$ *where* n *is the number of variables and* m *is the number of local terms in the Hamiltonian.*

Our algorithm shares the same trial and error approach of the Davis-Putnam procedure for classical 2SAT, but handles many difficulties arising in the quantum setting. First, a ground state of 2QSAT input may be entangled, a distinctive feature that classical 2SAT does not have. Thus the idea of setting some qubit to a certain state and propagating from there does not have a foundation in the first place. Indeed, if a rank-3 projector forces the only allowed state to be entangled, then any ground state is entangled in those two qubits. We overcome this by showing a *product-state theorem*, which asserts that for any frustration-free 2QSAT instance H that contains only rank-1 and rank-2 projectors, there always exists a ground state in the form of a tensor product of *single-qubit* states.

This structural theorem grants us the following approach: We try some candidate solution $|\psi\rangle_i$ on a qubit i, and propagate this along the graph. If no contradiction is found, it turns out that we can detach the explored part and recurse on the rest of the graph. If a contradiction is found, then we can identify two candidates $(i, |\psi\rangle_i)$ and $(j, |\phi\rangle_j)$ such that either assigning $|\psi\rangle_i$ to qubit i or assigning $|\phi\rangle_j$ to qubit j is correct, if there exists a solution at all. More details follow next.

To illustrate the main idea of our algorithm, let us suppose that the input contains only projectors of rank at most two. Such a system can be further simplified to a system consisting only of rank-1 projectors, by writing every rank-2 projector as a sum of two rank-1 projectors. Consider, for example, qubits 1 and 2 and a rank-1 projector $\Pi_{12} = |\psi\rangle\langle\psi|$ over these qubits. The product-state theorem implies that it suffices to search for a product ground state. Thus on the first two qubits, we are looking for states $|\alpha\rangle, |\beta\rangle$ such that $(\langle\alpha| \otimes \langle\beta|)\Pi_{12}(|\alpha\rangle \otimes |\beta\rangle) = 0$, which is equivalent to $(\langle\alpha| \otimes \langle\beta|) \cdot |\psi\rangle = 0$. In other words, we look for a product state $|\alpha\rangle \otimes |\beta\rangle$ that is perpendicular to $|\psi\rangle$. Assume that we have assigned qubit 1 with the state $|\alpha\rangle$ and we are looking for a state $|\beta\rangle$ for qubit 2. The crucial point, which enables us to solve 2QSAT efficiently, is that just like in the classical case, there are only two possibilities: (i) for any $|\beta\rangle$, the state $|\alpha\rangle \otimes |\beta\rangle$ is perpendicular to $|\psi\rangle$, or (ii) there is only one state $|\beta\rangle$ (up to an overall complex phase), for which $(\langle\alpha| \otimes \langle\beta|) \cdot |\psi\rangle = 0$. The first case happens if and only if $|\psi\rangle$ is by itself a product state of the form $|\psi\rangle = |\alpha^{\perp}\rangle \otimes |\xi\rangle$,

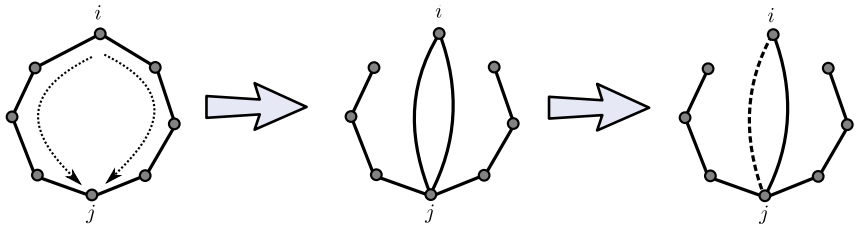


Figure 1 Handling a contradicting cycle: we slide edges that touch i along the two paths to j until we get a double edge with a 'tail'. We then use a structure lemma to deduce that at least one of these edges can be written as a product projector.

where $|\alpha^{\perp}\rangle$ is perpendicular to $|\alpha\rangle$ and $|\xi\rangle$ is arbitrary. If the second case happens, we say that state $|\alpha\rangle$ is propagated to state $|\beta\rangle$ by the constraint state $|\psi\rangle$.

This dichotomy enables us to propagate a product state $|s\rangle$ on part of the system until we reach a contradiction, or find that no further propagation is possible and we are left with a smaller Hamiltonian H_s. This smaller Hamiltonian consists of a subset of the original projectors, *without introducing new projectors*. It turns out that once an edge is checked for potential propagation, then no matter whether a propagation happens along the edge or not, the edge can be safely removed without changing the satisfiability. Thus the satisfiability of the original Hamiltonian H is the same as that of the smaller Hamiltonian H_s.

We still need to specify how the state $|\alpha\rangle$ is chosen to initialize the propagation. An idea is to begin with projectors $|\psi\rangle\langle\psi|$ for which $|\psi\rangle$ is a product state $|\alpha\rangle \otimes |\beta\rangle$. In such cases a product state solution must either have $|\alpha^{\perp}\rangle$ at the first qubit or $|\beta^{\perp}\rangle$ at the second. To maintain a linear running time, we propagate these two choices simultaneously until one of the propagations stops without contradiction, in which case the corresponding qubit assignment is made final. If both propagations end with contradiction, the input is rejected.

The more interesting case of the algorithm happens when we have only entangled rank-1 projectors. What should our initial state be then? We make an arbitrary assignment (say, $|0\rangle$) to any of the still unassigned qubits and propagate this choice. If the propagation ends without contradiction, we recurse. If a contradiction is found then we confront a challenging problem. In the classical case we could reverse our choice, say $x_0 = 0$, and try the other possibility, $x_i = 1$. But in the quantum case we have an infinite number of potential assignment choices. The solution is found by the following observation: Whenever a contradiction is reached, it can be attributed to a cycle of entangled projectors in which the assignment has propagated from qubit i along the cycle and returned to it with another value. Then using the techniques of 'sliding', which was introduced in Ref. [11], one can show that this cycle is equivalent to a system of one double edge and a 'tail' (see Fig. 1). Using a simple structure lemma, we are guaranteed that at least one of the projectors of the double edge can be turned into a product state projector, which, as in the previous stage, gives us two possible free choices.

Let us state here that our algorithm works in the algebraic model of computation: we suppose that every arithmetic operation on complex numbers can be done in unit time.

Classically, Davis-Putnam [6] and DPLL algorithms [5] are widely-used heuristics, forming the basis of today's most efficient solvers for general SAT. For quantum k-SAT, it could also be a good heuristic if we try to find product-state solutions, and in that respect our algorithm makes the first-step exploration.

Simultaneously and independently from our work and approximately at the same time, de Beaudrap and Gharibian [7] have also presented a linear time algorithm for quantum 2SAT.

The main difference between the two algorithms is how they deal with instances with only entangled rank-1 projectors. Contrarily to us, [7] handles these instances by using transfer matrix techniques to find discretizing cycles [16]. Most proofs are omitted from this version of the paper owing to space constraints.

2 Preliminaries

2.1 Notation

We will use the notation $[n] = \{1, \ldots, n\}$. For a graph $G = (V, E)$, and for a subset $U \subseteq V$ of the vertices, we denote by $G(U)$ the subgraph induced by U. Our Hilbert space is defined over n qubits, and is written as $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_n$, where $\mathcal{H}_i$ is the two-dimensional Hilbert space of the i^{th} qubit. We shall often write $|\alpha\rangle_i$ to emphasize that the 1-qubit state $|\alpha\rangle$ lives in $\mathcal{H}_i$. Similarly, $|\psi\rangle_{ij}$ denotes a 2-qubit state that lives in $\mathcal{H}_i \otimes \mathcal{H}_j$. For a 1-qubit state $|\alpha\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, we define its *perpendicular state* as $|\alpha^\perp\rangle = \alpha_1|0\rangle - \alpha_0|1\rangle$.

We shall denote local projectors either by Π_{ij}, or by Π_e, where $e = (i, j)$. When $i < j$, Π_{ij} is a 2-local projector on the qubits i, j; it can be written as $\Pi_{ij} = \hat{\Pi}_{ij} \otimes \mathbb{I}_{rest}$, where $\hat{\Pi}_{ij}$ is a projector working on $\mathcal{H}_i \otimes \mathcal{H}_j$ and $\mathbb{I}_{rest}$ is the identity operator on the rest of the system. Similarly, when $i = j$, $\Pi_{ii} = \hat{\Pi}_{ii} \otimes \mathbb{I}_{rest}$, where $\hat{\Pi}_{ii}$ is a projector defined in $\mathcal{H}_i$. Often, in order not to overload the notation, we shall use Π_{ij} instead of $\hat{\Pi}_{ij}$, even when acting on states in $\mathcal{H}_i \otimes \mathcal{H}_j$. Similarly, with a slight abuse the notation, we define the *rank* of a projector Π_e to be the dimension of the subspace that its local projector $\hat{\Pi}_e$ projects to, and it will be denoted by $\texttt{rank}(\Pi_e)$. We call a rank-1 projector $\Pi_e = |\psi\rangle\langle\psi|$, *entangled* if $|\psi\rangle$ is an entangled state, and *product* if $|\psi\rangle$ is a product state.

2.2 The 2QSAT problem

A quantum 2-SAT Hamiltonian on an n-qubit system is a Hermitian operator $H = \sum_{e \in I} \Pi_e$, for some $I \subseteq \{(i, j) \in [n] \times [n] : 1 \leq i \leq j \leq n\}$. We suppose that $\texttt{rank}(\Pi_{ii}) = 1$, for all $(i, i) \in I$, and $0 < \texttt{rank}(\Pi_{ij}) < 4$, for all $(i, j) \in I$ when $i < j$. The single-qubit projectors of H as well as its 2-qubit projectors of rank-3 are called *maximal* rank.

The *ground energy* of a Hamiltonian $H = \sum_{e \in I} \Pi_e$ is its smallest eigenvalue, and a *ground state* of H is an eigenvector corresponding to the smallest eigenvalue. The subspace of the ground states is called the *ground space*. A Hamiltonian is *frustration-free* if it has a ground state that is also simultaneously the ground state of all local terms. As explained in the introduction, if the Hamiltonian is made of local projectors, it is frustration-free if and only if there is a state that is a mutual zero eigenstate of all projectors, which happens if and only if the ground energy is 0. Therefore, if $|\Gamma\rangle$ is a ground state of a frustration-free quantum 2-SAT Hamiltonian, $\Pi_e|\Gamma\rangle = 0$ for all $e \in I$. We can also view each local projector as a *constraint* on at most two qubits, then a ground state satisfies every constraint.

It turns out that for the representation of the 2QSAT Hamiltonian, it will be helpful to eliminate the rank-2 projectors by decomposing each one of them into a sum of two rank-1 projectors. For every $(i, j) \in I$ such that $\texttt{rank}(\Pi_{ij}) = 2$, let $\Pi_{ij} = \Pi_{ij,1} + \Pi_{ij,2}$, where $\Pi_{ij,1}$ and $\Pi_{ij,2}$ are rank-1 projectors. Such projectors can be found in constant time. We therefore suppose without loss of generality that H is specified by

$$H = \sum_{\texttt{rank}(\Pi_{ij}) \neq 2} \Pi_{ij} + \sum_{\texttt{rank}(\Pi_{ij}) = 2} (\Pi_{ij,1} + \Pi_{ij,2}),$$

which we call the *rank*-1 *decomposition* of H.

To the rank-1 decomposition we associate a weighted, directed multigraph with self-loops $G(H) = (V, E, w)$, the *constraint graph* of H. By definition $V = \{i \in [n] : \exists j \in [n]$ such that $(i,j) \in I$ or $(j,i) \in I\}$, For every rank-3 and rank-1 projector acting on two qubits, there is an edge in each direction between the two nodes representing them. For every projector acting on a single qubit, there is a self-loop. Finally, for every rank-2 projector, there are two parallel edges in each direction between nodes representing its qubits. Because of the parallel edges, E is not a subset of $V \times V$. Formally, $E = E_1 \cup E_2$ where respectively

$$E_1 = \{(i,j) \in [n] \times [n] : (i,j) \in I \text{ and } \mathtt{rank}(\Pi_{ij}) \in \{1,3\}, \text{ or } (j,i) \in I \text{ and } \mathtt{rank}(\Pi_{ji}) \in \{1,3\}\},$$

and

$$E_2 = \{(i,j,b) \in [n] \times [n] \times [2] : (i,j) \in I \text{ and } \mathtt{rank}(\Pi_{ij}) = 2, \text{ or } (j,i) \in I \text{ and } \mathtt{rank}(\Pi_{ji}) = 2\}.$$

We say that an edge $e \in E$ *goes from* i *to* j if $e \in \{(i,j),(i,j,1),(i,j,2)\}$. For a projector Π acting on two qubits, we define its *reverse* projector Π^{rev} by $\Pi^{rev}|\alpha\rangle|\beta\rangle = \Pi|\beta\rangle|\alpha\rangle$, and for $i \leq j$ and $b \in [2]$, we set $\Pi_{ji} = \Pi_{ij}^{rev}$ and $\Pi_{jib} = \Pi_{ijb}^{rev}$. Then for an edge (i,j), its *weight* is defined as $w(i,j) = \Pi_{ij}$, and analogously for an edge (i,j,b), we set $w(i,j,b) = \Pi_{ijb}$.

We will suppose that the input to our problem is the constraint graph $G(H)$ of the Hamiltonian, given in the standard adjacency list representation of weighted graphs, naturally modified for dealing with the parallel edges. In this representation there is a linked list of size at most n containing one element for each vertex, and the element i in this list is also pointing towards a linked list containing an element for every edge (i,j) or (i,j,b). For an edge (i,j), this element contains j, the projector Π_{ij} and a pointer towards the next element in the list, for an edge (i,j,b) it also contains the value b. The problem 2QSAT is defined formally as follows.

2QSAT
Input: The constraint graph $G(H)$ of a 2-local Hamiltonian H, given in the adjacency list representation.
Output: A solution if H is frustration free, "H is unsatisfiable" if it is not.

2.3 Simple ground states

Our algorithm is based crucially on the following *product state theorem*, which says that any frustration-free 2QSAT Hamiltonian composed has a ground state which is a product state of single qubit and two-qubit states, where the latter only appear in the support of rank-3 projectors. A slightly weaker claim of that form has already appeared in Theorem 2 of Ref. [3]. The difference here is that we specifically attribute the 2-qubits states in the product state to rank-3 projectors. Just as in Ref. [3], our derivation begins with Theorem 1 of Ref. [3], which we give below. It relies on the notion of a *genuinely entangled* state in an n-qubit system, which is a pure state that is not a product state with respect to any bi-partition of the system. Then Theorem 1 in [3] states

▶ **Proposition 2.** *A 2-local frustration-free Hamiltonian on n qubits which has a genuinely entangled ground state always has a product ground state, whenever $n \geq 3$.*

We will also need the following simple fact about 2-dimensional subspaces in $\mathbb{C}^2 \otimes \mathbb{C}^2$.

▶ **Fact 3.** *Any 2-dimensional subspace V of the 2-qubit space $\mathbb{C}^2 \otimes \mathbb{C}^2$ contains at least one product state, which can be found in constant time.*

The proofs of Fact 3 and our Product State Theorem are omitted owing to space constraints.

▶ **Theorem 4** (Product State Theorem). *Any frustration-free* 2QSAT *Hamiltonian has a ground state which is a tensor product of one qubit and two-qubit states, where two-qubit states only appear in the support of rank-3 projectors.*

2.4 Assignments

Let $H = \sum_{e \in I}$ be a 2-local Hamiltonian. By Theorem 4, if H is frustration free then it has a ground state which is the tensor product of 1-qubit and 2-qubit entangled states, where the latter only appear in pairs of qubits corresponding to rank-3 projectors. To build up a ground state of such form, our algorithm will use partial assignments, or shortly assignments. An *assignment* s is a mapping from $[n]$. For every $i \in [n]$, the value $s(i)$ is either a 1-qubit state $|\alpha\rangle$, or a 2-qubit entangled state $|\gamma\rangle_{ij}$ for some $j \neq i$, or a symbol from the set $\{\ominus, X\}$. If $s(i) = |\alpha\rangle$ or $s(i) = |\gamma\rangle_{ij}$, then this value is assigned to qubit variable i, and in the latter case the entangled state is shared with variable j. The symbol $\ominus$ is used for unassigned variables, and the symbol X is used when several values are assigned to some variable.

We define the *support* of s by $\mathsf{supp}(s) = \{i \in [n] : s(i) \neq \ominus\}$. The assignment s is *empty* if $\mathsf{supp}(s) = \emptyset$. When there is no danger of confusion, we will denote the empty assignment also by $\ominus$. We say that an assignment is *coherent* if for every i, we have $s(i) \neq X$, and whenever $s(i) = |\gamma\rangle_{ij}$, we also have $s(j) = |\gamma\rangle_{ji}$. For coherent assignments s and s', we say that s' is an *extension* of s, if for every i, such that $s(i) \neq \ominus$, we have $s'(i) = s(i)$. A coherent assignment is *total* if $s(i) \neq \ominus$, for all i. Clearly, a coherent assignments defines a product state of 1-qubit and 2-qubits states on qubits in its support. We denote this state by $|s\rangle$. We say that a coherent assignment s *satisfies* a projector Π_e, or simply that it satisfies the edge e, if for any total extension s' of s we have $\Pi_e|s'\rangle = 0$.

For $H = \sum_{e \in I} \Pi_e$ given in rank-1 decomposition, and a coherent s, we define the *reduced Hamiltonian* H_s of s as $H_s = H - \sum_{s \text{ satisfies } e} \Pi_e$. We will denote the constraint graph $G(H_s)$ of the reduced Hamiltonian H_s by $G_s = (V_s, E_s)$. We call a coherent assignment s a *pre-solution* if it has a total extension s' satisfying every constraint in H, and we call s is a *solution* if s itself satisfies every constraint in H. Obviously, an assignment is a solution if and only if G_s is the empty graph. A coherent assignment s is *closed* if $\mathsf{supp}(s) \cap V_s = \emptyset$.

3 Propagation

The crucial building block of our algorithm is the propagation of values by rank-1 projectors. This is the quantum analog of the classical propagation process when for example the clause $x_i \vee x_j$ propagates the value $x_i = 0$ to the value $x_j = 1$ in the sense that given $x_i = 0$, the choice $x_j = 1$ is the only possibility to make the clause true. In the quantum case this notion has already appeared in Ref. [16], and can in fact be traced back also to Bravyi's original work. Here, we shall adopt the following definition

▶ **Definition 5** (Propagation). Let $\Pi_e = |\psi\rangle\langle\psi|$ be a rank-1 projector acting on variables i, j, and let $|\alpha\rangle$ be either a 1-qubit state assigned to variable i, or a 2-qubit entangled state assigned to variables k, i for some $k \neq j$. We say that Π_e *propagates* $|\alpha\rangle$ if, up to a phase, there exists a unique 1-qubit state $|\beta\rangle$ such that $\Pi_e|\alpha\rangle_i \otimes |\beta\rangle_j = 0$. In such a case we say that $|\alpha\rangle$ is propagated to $|\beta\rangle$ along Π_e, or that Π_e propagated $|\alpha\rangle$ to $|\beta\rangle$.

We establish a sequence of Lemmas on propagation whose proofs are not discussed here owing to space contraints. The first lemma shows how the propagation properties of $\Pi_e = |\psi\rangle\langle\psi|$ are determined by entanglement in $|\psi\rangle$.

▶ **Lemma 6.** *Consider the rank-1 projector $\Pi_e = |\psi\rangle\langle\psi|$, defined on qubits i, j. If $|\psi\rangle$ is entangled, it propagates* every 1*-qubit state $|\alpha\rangle_i$ to a state $|\beta(\alpha)\rangle_j$ such that if $|\alpha\rangle_i \neq |\alpha'\rangle_i$ then $|\beta(\alpha)\rangle_j \neq |\beta(\alpha')\rangle_j$. This propagation can be calculated in constant time. When $|\psi\rangle$ is a product state $|\psi\rangle = |x\rangle_i \otimes |y\rangle_j$, the projector Π_e does not propagate states that are proportional to $|x^\perp\rangle_i$, while all other states are propagated to $|y^\perp\rangle_j$.*

We now present two lemmas that describe the structure of the *global* ground state of the system, if we know that part of it is in a tensor product of 1-qubit or 2-qubits states, which are then propagated by some Π_e.

▶ **Lemma 7** (Single qubit propagation)**.** *Consider a frustration-free* 2QSAT *system $H = \sum_{e\in I} \Pi_e$ with a rank-1 projector $\Pi_e = |\psi\rangle\langle\psi|$ between qubits i, j, and assume that H has a ground state of the form $|\Gamma\rangle = |\alpha\rangle_i \otimes |rest\rangle$. Then:*

1. *If Π_e propagates $|\alpha\rangle_i$ to $|\beta\rangle_j$ then necessarily $|rest\rangle = |\beta\rangle_j \otimes |rest'\rangle$.*
2. *$|\Gamma\rangle$ is also a ground state of the* 2QSAT *Hamiltonian $H - \Pi_e$.*

▶ **Lemma 8** (Entangled 2-qubits propagation)**.** *Consider a frustration-free* 2QSAT *system H with a rank-1 projector $\Pi_e = |\psi\rangle\langle\psi|$ between qubits i, j. Assume that H has a ground state of the form $|\Gamma\rangle = |\phi\rangle_{ik} \otimes |rest\rangle$, where $|\phi\rangle$ is an entangled state on qubits i, k with $k \neq j$. Then:*

1. *$|\psi\rangle$ is a product state $|\psi\rangle = |x\rangle|y\rangle$.*
2. *Π_e propagates $|\phi\rangle$ to $|y^\perp\rangle$ and necessarily $|rest\rangle = |y^\perp\rangle_j \otimes |rest'\rangle$.*
3. *$|\Gamma\rangle$ is also a ground state of the* 2QSAT *Hamiltonian $H - \Pi_e$.*

Let H be a 2QSAT Hamiltonian in rank-1 decomposition, let s be a coherent assignment, and let $G_s = (V_s, E_s)$ be the constraint graph of the reduced Hamiltonian H_s. We would like to describe in G_s the result of the *iterated propagation* process when a value given to variable i is first propagated along all possible projectors, followed by these propagated values being propagated on their turn, and so on until no more values assigned during this process can be propagated further. The propagation can start when the initial value is already assigned by s, that is, when $s(i) = |\delta\rangle$ for $|\delta\rangle \in \{|\alpha\rangle, |\gamma\rangle_{ij}\}$, where $|\alpha\rangle$ is some 1-qubit state and $|\gamma\rangle_{ij}$ some 2-qubit state, or it can get started when $s(i) = \ominus$, in which case we shall explicitly choose a 1-qubit state $|\alpha\rangle$ and assign it to i.

Now, let s, i and $|\delta\rangle$ be such that $s(i) \in \{\ominus, |\delta\rangle\}$. We say that an edge $e \in E_s$, in the constraint graph G_s, from i to j *propagates* $|\delta\rangle$ if Π_e propagates it, and we denote by $\mathsf{prop}(s, e, |\delta\rangle)$ the state $|\delta\rangle$ is propagated to. We further generalize the notion of propagation in G_s from edges to paths. Let $i = i_0, i_1, \ldots i_k$ be vertices in V_s, and let e_j be an edge from i_j to i_{j+1}, for $j = 0, \ldots, k-1$. Let $s(i) \in \{\ominus, |\delta\rangle\}$, and set $|\alpha_0\rangle = |\delta\rangle$. Let $|\alpha_1\rangle, \ldots, |\alpha_k\rangle$ be states such that the propagation of $|\alpha_j\rangle$ along Π_{e_j} is $|\alpha_{j+1}\rangle$, for $j = 0, \ldots, k-1$. Then we say that the path $p = (e_0, \ldots, e_{k-1})$ from i_0 to i_k *propagates* $|\delta\rangle$, and we set $\mathsf{prop}(s, p, |\delta\rangle) = |\alpha_k\rangle$. We say that a vertex $j \in V_s$ is *accessible* by propagating $|\delta\rangle$ from i if either $j = i$ or there is a path from i to j that propagates $|\delta\rangle$. We denote by $V_s^{\mathsf{prop}}(i, |\delta\rangle)$ the set of such vertices, and by $\mathsf{ext}_s^{\mathsf{prop}}(i, |\delta\rangle)$ the extension of s by the values given to the vertices in $V_s^{\mathsf{prop}}(i, |\delta\rangle)$ by iterated propagation.

Let us suppose that $s' = \mathsf{ext}_s^{\mathsf{prop}}(i, |\delta\rangle)$ is also coherent. The set $V_s^{\mathsf{prop}}(i, |\delta\rangle)$ divides the edges E_s into three disjoint subsets: the edges E_1 of the induced subgraph $G(V_s^{\mathsf{prop}}(i, |\delta\rangle))$, the edges E_2 between the induced subgraphs $G(V_s^{\mathsf{prop}}(i, |\delta\rangle))$ and $G(V_s \setminus V_s^{\mathsf{prop}}(i, |\delta\rangle))$, and the

Algorithm 1 Propagation$(s, G_s, i, |\delta\rangle)$

$s(i) := |\delta\rangle$
create a list L and a queue Q, and **put** i into Q
while s is coherent and Q is not empty **do**
  remove the head j of Q
  for all edge e from j to k **do**
    remove e from E_s
    if e propagates $s(j)$ **then**
      $s(k) := \begin{cases} \mathsf{prop}(s, e, s(j)) & \text{if } s(k) = \ominus \\ X & \text{if } s(k) \notin \{\ominus, \mathsf{prop}(s, e, s(j))\} \end{cases}$
      enqueue k
    if e is not propagating and k is not in L **then put** k into L
  remove j from V_s
if s is not coherent **return** "unsuccessful"
for all k in L **do**
  for all edges e from k to ℓ **do**
    if ℓ was removed from V_s **then remove** e
  if all edges outgoing from k were removed **then remove** k from V_s

edges E_3 of the induced subgraph $G(V_s \setminus V_s^{\mathsf{prop}}(i, |\delta\rangle))$. While the edges in $E_1 \cup E_2$ are satisfied by s', none of the edges in E_3 is satisfied. Therefore $G_{s'}$ is nothing but $G(V_s \setminus V_s^{\mathsf{prop}}(i, |\delta\rangle))$ without the isolated vertices, and it can be constructed by the following process. Given s and i, the edges in $E_1 \cup E_2$ can be traversed via a breadth first search rooted at i. The levels of the tree are decided dynamically: at any level the next level is composed of those vertices whose value is propagated from the current level. The leaves of the tree are vertices in $V_s \setminus V_s^{\mathsf{prop}}(i, |\delta\rangle)$. The algorithm Propagation uses a temporary queue Q to implement this process.

▶ **Lemma 9** (Propagation Lemma). *Let Propagation$(s, G_s, i, |\delta\rangle)$ be called when H_s doesn't have rank-3 constraints, and $s(i) \in \{\ominus, |\delta\rangle\}$. Let s' and $G' = (V', E')$ be the outcome of the procedure. Then:*

1. *If Propagation$(s, G_s, i, |\delta\rangle)$ doesn't return "unsuccessful" then $s' = \mathsf{ext}_s^{\mathsf{prop}}(i, |\delta\rangle)$ and $G' = G_{s'}$. Moreover, if s is a pre-solution then s' is a pre-solution, and if s is closed then s' is also closed.*
2. *If Propagation(s, G_s, i) returns "unsuccessful" then there is no solution z which is an extension of s and for which $z(i) = |\delta\rangle$.*
3. *The complexity of the procedure is $O(|E_s| - |E_{s'}|)$.*

4 The main algorithm

4.1 Description of the algorithm

We now give in broad strokes the description of our algorithm called 2QSATSolver. It takes as input the adjacency list representation of the constraint graph $G(H)$ of a 2-local Hamiltonian H in rank-1 decomposition. The algorithm uses four global variables: assignments s_0 and s_1 initialized to $\ominus$, and graphs G_0 and G_1 in the adjacency list representation, initialized to $G(H)$. The algorithm consists of four phases, and except the first one, each phase consists of

Algorithm 2 2QSATSolver($G(H)$)

$s_0 = s_1 := \ominus, \quad G_0 = G_1 := G(H)$ ▷ Initialize global variables

MaxRankRemoval() ▷ Remove maximal rank constraints

while there exist $i \in V_0$ such that $s(i) \neq \ominus$ **do** ▷ Propagate all assigned values
  PROPAGATE$(s_0, G_0, i, s_0(i))$ for some vertex i in G_0 such that $s_0(i) \neq \ominus$
  if the propagation returns "unsuccessful" **output** "H is unsatisfiable"
  $s_1 := s_0, G_1 := G_0$

while there exists in G_0 a product edge with constraint $|\alpha_0^\perp\rangle_{i_0} \otimes |\alpha_1^\perp\rangle_{i_1}\langle\alpha_0^\perp|_{i_0} \otimes \langle\alpha_1^\perp|_{i_1}$ **do**
  ParallelPropagation$(i_0, |\alpha_0\rangle, i_1, |\alpha_1\rangle)$ ▷ Remove product constraints

while G_0 is not empty **do** ▷ Remove entangled constraints
  ProbePropagation(i) for some vertex i

output $|s\rangle$ for any total extension s of s_0.

several stages, where essentially one stage corresponds to one Propagation process. In the case of an unsatisfiable *Hamiltonian* the algorithm at some point outputs "H is unsatisfiable" and stops. This happens when either the maximal rank constraints are already unsatisfiable, or at some later point when several values are assigned to the same variable during a necessary propagation process.

In the case of a frustration-free Hamiltonian, at the beginning and end of each stage, we will have $s_0 = s_1$, and $G_0 = G_1 = G_{s_0}$. In the first two phases only (s_0, G_0) develops, and is copied to (s_1, G_1) at the end of the phase. In the last two phases, (s_0, G_0) and (s_1, G_1) develop independently, but only the result of one of the two processes is retained and is copied into the other variable at the end of the phase. This parallel development of the two processes is necessary for complexity considerations, ensuring that the useless work done is proportional to the useful work.

In the first phase the procedure MaxRankRemoval satisfies, if possible, all constraints of maximal rank. In the second phase all these assignments are propagated, which, if successful, result in a closed assignment s such that H_s has only rank-1 constraints. In the third phase the procedure ParallelPropagation satisfies the product constraints one by one and propagates the assigned values. To satisfy a product constraint, the only two possible choices are tried and propagated in parallel. In the fourth phase the remaining entangled constraints are taken care of, again, one by one. To satisfy a constraint, an arbitrary value is tried and propagated. In case of an unsuccessful propagation we are able to efficiently find a product constraint implied by the entangled constraints considered during the propagation, and therefore it becomes possible to proceed as in phase three. In case of success we are left with a satisfying assignment and the empty constraint graph. Theorem 1 is an immediate consequence of the following result.

▶ **Theorem 10.** *Let $G(H) = (V, E)$ be the constraint graph of a 2-local Hamiltonian. Then:*

1. *If H is frustration-free, the algorithm 2QSATSolver($G(H)$) outputs a ground state $|s\rangle$.*
2. *If H is not frustration-free, the algorithm 2QSATSolver($G(H)$) outputs "H is unsatisfiable".*
3. *The running time of the algorithm is $O(|V| + |E|)$.*

The proofs of the series of lemmas in the following sections is omitted. Theorem 10 will be proven in Section 4.5.

Algorithm 3 ParallelPropagation$(i_0, |\alpha_0\rangle, i_1, |\alpha_1\rangle)$

Run in parallel Propagation$(s_0, G_0, i_0, |\alpha_0\rangle)$ and Propagation$(s_1, G_1, i_1, |\alpha_1\rangle)$
until one of them terminates successfully **or** both terminate unsuccessfully

if both propagations terminate unsuccessfully **then**
  output "H is unsatisfiable"
else let Propagation$(s_0, G_0, i_0, |\alpha_0\rangle)$ terminate first (the other case is symmetric)
  undo Propagation$(s_1, G_1, i_1, |\alpha_1\rangle)$
  $s_1 := s_0,\ G_1 := G_0$

4.2 Max rank removal

The MaxRankRemoval procedure is conceptually very simple. Since every maximal rank constraint has a unique solution (up to a global phase), it makes this assignment for each constraint, and then checks if this is globally consistent. The description of the procedure and the proof of Lemma 11 are omitted.

▶ **Lemma 11.** *Let s_0, G_0, s_1, G_1 be the outcome of MaxRankRemoval. Then:*
1. *If MaxRankRemoval doesn't output "H is unsatisfiable" then s_0 is coherent, it satisfies every maximal rank constraint, $G_0 = G(H_{s_0})$ and $s_0 = s_1, G_0 = G_1$. Moreover, if H is satisfiable then s_0 is a pre-solution.*
2. *If MaxRankRemoval outputs "H is unsatisfiable" then indeed H is unsatisfiable.*
3. *The complexity of the procedure is $O(|V| + |E|)|$.*

4.3 Algorithm ParallelPropagation

The procedure ParallelPropagation is called when s_0 is a closed assignment, and in G_{s_0} there is a product edge. Since there are only two ways to satisfy a product constraint, these are tried and propagated in parallel. If one of these propagations terminates successfully, the other is stopped, which ensures that the overall work done is proportional to the progress made.

▶ **Lemma 12.** *Let ParallelPropagation be called when s_0 is closed, H_{s_0} doesn't have rank-3 constraints, $G_0 = G_{s_0}$, in G_0 there exists a product edge from i_0 to i_1 with constraint $|\alpha_0^\perp\rangle \otimes |\alpha_1^\perp\rangle$, and $s_1 = s_0, G_1 = G_0$. Let s_0', s_1', G_0', G_1' be the outcome of the procedure. Then:*
1. *If ParallelPropagation doesn't output "H is unsatisfiable" then s_0' is a proper closed extension of s_0, $G_0' = G_{s_0'}$, and $s_1' = s_0'$, $G_1' = G_0'$. Moreover, if s is a pre-solution then s_0' is a pre-solution.*
2. *If ParallelPropagation outputs "H is unsatisfiable" then indeed H is unsatisfiable.*
3. *The complexity of the procedure is $O(|E_{s_0}| - |E_{s_0'}|)$.*

4.4 Algorithm ProbePropagation

The procedure ProbePropagation is evoked when s_0 is a closed assignment, and in G_{s_0} there are only entangled constraints. It picks an arbitrary vertex in $i \in V_s$, assigns $|0\rangle$ (an arbitrary value) to it, and propagates this choice. In the lucky case of successful propagation this is repeated. Otherwise, we reach a contradiction: there is some $j \in V_s$, such that two propagating paths assign different values to it. We prove below the Sliding Lemma which already appeared in Ref. [11]. It implies that when $i_0 \to i_1 \to \ldots \to i_k$ is a propagating path of

Algorithm 4 ProbePropagation(i)

Propagation$(s_0, G_0, i, |0\rangle)$.
if the propagation is successful **then** $s_1 := s_0$, $G_1 := G_0$
else
Let j such that $|s_0(j)| > 1$
find two paths p_1 and p_2 in G_0 from i to j such that $\mathsf{prop}(s_0, p_1, |0\rangle) \neq \mathsf{prop}(s_0, p_2, |0\rangle)$
find a product state $|\alpha^\perp\rangle_i \otimes |\beta^\perp\rangle_j$ in the two dimensional space $\mathsf{span}\{\mathsf{slide}(p_1), \mathsf{slide}(p_2)\}$
undo Propagation$(s_0, G_0, i, |0\rangle)$
ParallelPropagation$(i, |\alpha\rangle, j, |\beta\rangle)$

entangled rank-1 projectors, the ground space of the Hamiltonian $\Pi_{i_0,i_1} + \Pi_{i_1,i_2} + \ldots + \Pi_{i_{k-1},i_k}$ is equal to the ground state of the Hamiltonian $\Pi_{i_0,i_k} + \Pi_{i_1,i_2} + \ldots + \Pi_{i_{k-1},i_k}$, where Π_{i_0,i_k} is a new projector defined on the qubits (i_0, i_k) that replaces the projector Π_{i_0,i_1}. Graphically, this can be viewed as if we are sliding the (i_0, i_1) edge on the path $i_1 \to \ldots \to i_k$. Therefore, if we have two propagating paths starting at i and ending at j, they define two projectors on qubits (i, j). As we shall see, if these two paths are contradicting then necessarily the two projectors are different, which by Lemma 3 implies the existence of a product constraint between (i_0, i_k) variables. In this case, we can proceed by calling the procedure ProbePropagation.

▶ **Lemma 13** (Sliding Lemma)**.** *Consider a system on 3 qubits i, j and k. Suppose that we have a two rank-1 constraints $\Pi_1 = |\psi_1\rangle\langle\psi_1|_{ij}$ on qubits (i, j) and $\Pi_2 = |\psi_2\rangle\langle\psi_2|_{jk}$ on qubits (j, k). If $|\psi_2\rangle$ is entangled, there is another rank-1 constraint $\Pi_3 = |\psi_3\rangle\langle\psi_3|_{ik}$ on qubits (i, k) such that the ground space of $\Pi_1 + \Pi_2$ is identical to the ground space of $\Pi_2 + \Pi_3$. In addition, if a single qubit state $|\alpha\rangle_i$ is propagated by $\Pi_1 + \Pi_2$ to $|\beta\rangle_k$, then it is also propagated to $|\beta\rangle_k$ directly via Π_3.*

Applying Lemma 13 iteratively, we reach the following corollary

▶ **Corollary 14.** *Let $H = \sum_{e \in I} H_e$ be a 2-local Hamiltonian in rank-1 decomposition. Let $i_0, i_1, \ldots i_k$ be vertices in V, and let e_j be an edge from i_j to i_{j+1}, for $j = 0, \ldots, k-1$ such that the rank-1 constraints Π_{e_j} are entangled. Then there exists a 2-qubit entangled state $|\gamma\rangle$ between i_0 and i_k such that the ground space of $\sum_{j=0}^{k-1} \Pi_{e_j}$ is identical to the ground space of $\sum_{j=1}^{k-1} \Pi_{e_j} + |\gamma\rangle\langle\gamma|_{i_0,i_k}$. Moreover, if $|\alpha\rangle_{i_0}$ is propagated to $|\beta\rangle_{i_k}$ along the path, then it is also propagated directly by $|\gamma\rangle\langle\gamma|_{i_0,i_k}$.*

We will denote the state $|\gamma\rangle$ in the conclusion of the corollary by $\mathsf{slide}(p)$.

▶ **Lemma 15.** *Let ProbePropagation be called when s_0 is closed, H_{s_0} has only rank-1 entangled constraints, $G_0 = G_{s_0}$, and $s_1 = s_0, G_1 = G_0$. Let s'_0, s'_1, G'_0, G'_1 be the outcome of the procedure. Then:*

1. *If ProbePropagation doesn't output "H is unsatisfiable" then s'_0 is a proper closed extension of s_0, $G'_0 = G_{s'_0}$, and $s'_1 = s'_0$, $G'_1 = G'_0$. Moreover, if s is a pre-solution then s'_0 is a pre-solution.*
2. *If ParallelPropagation outputs "H is unsatisfiable" then indeed H is unsatisfiable.*
3. *The complexity of the procedure is $O(|E_{s_0}| - |E_{s'_0}|)$.*

4.5 Analysis of the algorithm

Proof of Theorem 10. If H is frustration free then by Lemma 11 MaxRankRemoval outputs a pre-solution s_0 that satisfies every maximal rank constraint. By the Propagation Lemma, at

the end of Phase two, in addition s_0 is a closed assignment. By Lemma 12 ParallelPropagation outputs s_0 such that there are only entangled constraints in H_s. By Lemma 15 at the end of the algorithm in addition H_s is empty, and therefore s is a solution.

If the algorithm doesn't output "H is unsatisfiable" then by Lemma 11, by the Propagation Lemma, and by Lemmas 12 and 15 it outputs a coherent assignment s such that G_s is the empty graph, and therefore s is a solution.

The complexity of MaxRankRemoval by Lemma 11 is $O(|E|)$. After the second phase, the propagation of the assigned values during MaxRankRemoval, the copying of s_0 and G_0 into respectively s_1 and G_1 can be done by executing the same propagation steps this time with s_1 and G_1. The complexity of the rest of the algorithm by the Propagation Lemma, and Lemmas 12 and 15 is a telescopic sum which sums up to also $O(|E|)$. ◀

References

1. B. Aspvall, M. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979. Erratum: Information Processing Letters 14(4): 195 (1982). doi:10.1016/0020-0190(79)90002-4.
2. S. Bravyi. Efficient algorithm for a quantum analogue of 2-SAT. In K. Mahdavi, D. Koslover, and L. L. Brown, editors, *Contemporary Mathematics*, volume 536. American Mathematical Society, 2011. URL: http://arxiv.org/abs/quant-ph/0602108.
3. J. Chen, X. Chen, R. Duan, Z. Ji, and B. Zeng. No-go theorem for one-way quantum computing on naturally occurring two-level systems. *Physical Review A*, 83(5):050301, 2011.
4. S. Cook. The complexity of theorem proving procedures. In *Proceedings of the Third Annual ACM Symposium*, pages 151–158, New York, 1971. ACM.
5. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, July 1962. doi:10.1145/368273.368557.
6. M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, July 1960. doi:10.1145/321033.321034.
7. N. de Beaudrap and S. Gharibian. A linear time algorithm for quantum 2-SAT. *CoRR*, abs/1508.07338, 2015. To appear in 31st Conference on Computational Complexity. URL: http://arxiv.org/abs/1508.07338.
8. J. Eisert, M. Cramer, and M. Plenio. Area laws for the entanglement entropy – a review. *Reviews of Modern Physics*, 82(277), 2010.
9. S. Even, A. Itai, and A. Shamir. On the complexity of timetable and multicommodity flow problems. *SIAM J. Comput.*, 5(4):691–703, 1976. doi:10.1137/0205048.
10. D. Gosset and D. Nagaj. Quantum 3-sat is qma1-complete. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 0:756–765, 2013. doi:10.1109/FOCS.2013.86.
11. Z. Ji, Z. Wei, and B. Zeng. Complete characterization of the ground-space structure of two-body frustration-free hamiltonians for qubits. *Physical Review A*, 84:042338, 2011.
12. R. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. URL: http://www.cs.berkeley.edu/~luca/cs172/karp.pdf.
13. A. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):2–30, 2003. doi:10.1016/S0003-4916(02)00018-0.
14. A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, Boston, MA, USA, 2002.

15 M. Krom. The decision problem for a class of first-order formulas in which all disjunctions are binary. *Mathematical Logic Quarterly*, 13(1-2):15–20, 1967. doi:10.1002/malq.19670130104.

16 C. Laumann, R. Moessner, A. Scardicchio, and S. Sondhi. Phase transitions and random quantum satisfiability. *Quantum Information & Computation*, 10(1), 2010.

17 L. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

18 C. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 163–169, 1991. doi:10.1109/SFCS.1991.185365.

19 S. Sachdev. *Quantum phase transitions*. Wiley Online Library, 2007.

20 G. Vidal, J.-I. Latorre, E. Rico, and A. Kitaev. Entanglement in quantum critical phenomena. *Phys. Rev. Lett.*, 90:227902, Jun 2003. doi:10.1103/PhysRevLett.90.227902.

Optimal Quantum Algorithm for Polynomial Interpolation*

Andrew M. Childs[†1], Shih-Han Hung[2], Wim van Dam[‡3], and Igor E. Shparlinski[§4]

1 **Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, USA; and Joint Center for Quantum Information and Computer Science, University of Maryland, College Park, USA**
2 **Joint Center for Quantum Information and Computer Science, University of Maryland, College Park, USA**
3 **Departments of Computer Science and Physics, University of California, Santa Barbara, USA**
4 **Department of Pure Mathematics, University of New South Wales, Sydney, Australia**

Abstract

We consider the number of quantum queries required to determine the coefficients of a degree-d polynomial over $\mathbb{F}_q$. A lower bound shown independently by Kane and Kutin and by Meyer and Pommersheim shows that $d/2 + 1/2$ quantum queries are needed to solve this problem with bounded error, whereas an algorithm of Boneh and Zhandry shows that d quantum queries are sufficient. We show that the lower bound is achievable: $d/2 + 1/2$ quantum queries suffice to determine the polynomial with bounded error. Furthermore, we show that $d/2+1$ queries suffice to achieve probability approaching 1 for large q. These upper bounds improve results of Boneh and Zhandry on the insecurity of cryptographic protocols against quantum attacks. We also show that our algorithm's success probability as a function of the number of queries is precisely optimal. Furthermore, the algorithm can be implemented with gate complexity $\mathrm{poly}(\log q)$ with negligible decrease in the success probability. We end with a conjecture about the quantum query complexity of multivariate polynomial interpolation.

1998 ACM Subject Classification F.1.1 Models of Computation, F.2.1 Numerical Algorithms and Problems

Keywords and phrases Quantum algorithms, query complexity, polynomial interpolation, finite fields

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.16

1 Introduction

Let $f(X) = c_d X^d + \cdots + c_1 X + c_0 \in \mathbb{F}_q[X]$ be an unknown polynomial of degree d, specified by its coefficient vector $c \in \mathbb{F}_q^{d+1}$. Suppose q and d are known and we are given a black box that evaluates f on any desired $x \in \mathbb{F}_q$. (We assume $q > d$ so that different coefficients

* The extended version of this article is available at http://arxiv.org/abs/1509.09271
† AMC acknowledges support from ARO (grant W911NF-12-1-0482), CIFAR, IARPA, NSF (grant 1526380), and NRO.
‡ WvD acknowledges support from NSF (grant 1314969).
§ IES acknowledges support from ARC (grant DP140100118).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 16; pp. 16:1–16:13


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

correspond to distinct functions $f\colon \mathbb{F}_q \to \mathbb{F}_q$.) In the *polynomial interpolation problem*, our goal is to learn f – that is, to determine the vector c – by querying this black box. We would like to determine how many queries are required to solve this problem.

The classical query complexity of polynomial interpolation is well known: $d+1$ queries to f are clearly sufficient and are also necessary to determine the polynomial, even with bounded error. Shamir [17] used this fact to construct a cryptographic protocol that divides a secret into $d+1$ parts such that knowledge of all the parts can be used to infer the secret, but any d parts give no information about the secret. The security of this protocol relies on the fact that if f is chosen uniformly at random, and if we only know d function values $f(x_1), \ldots, f(x_d)$, then we cannot guess the value $f(x_{d+1})$ for a point $x_{d+1} \notin \{x_1, \ldots, x_d\}$ with probability greater than $1/q$ (that is, there is no advantage over random guessing). This example motivates understanding the query complexity of polynomial interpolation precisely, since a single query can dramatically increase the amount of information that can be extracted.

The quantum query complexity of polynomial interpolation has also been studied previously. Kane and Kutin [9] and Meyer and Pommersheim [12] independently showed that $d/2+1/2$ quantum queries are needed to solve the problem with bounded error. Furthermore, Kane and Kutin conjectured that $d+1$ quantum queries might be necessary. This was refuted by Boneh and Zhandry, who showed that d quantum queries suffice to solve the problem with probability $1 - O(1/q)$ [3]. (While the notation $O(\cdot)$ only indicates an asymptotic upper bound on the absolute value, we sometimes write $1 - O(\cdot)$ to indicate a bound on a quantity that is at most 1.) To show this, they described a 1-query quantum algorithm that determines a linear polynomial with probability $1 - O(1/q)$. The result for general d follows because $d-1$ classical queries can be used to reduce the case of a degree-d polynomial to that of a linear polynomial. However, this work left a substantial gap between the lower and upper bounds.

Here we present an improved quantum algorithm for polynomial interpolation. We show that the aforementioned lower bounds are tight: with d fixed, $k = d/2 + 1/2$ queries suffice to solve the problem with constant success probability. While the success probability at this value of k has a q-independent lower bound, it decreases rapidly with k, scaling like $1/k!$. This raises the question of how the success probability increases as we make more queries. We show that there is a sharp transition as k is increased: in particular, with $k = d/2 + 1$ queries, the algorithm succeeds with a probability that approaches 1 for large q.

Our algorithm is motivated by the pretty good measurement (PGM) approach to the hidden subgroup problem (HSP) [1]. In this approach, one queries the black box on uniform superpositions to create *coset states* and then makes entangled measurements on several coset states to infer the hidden subgroup. As in the PGM approach (and in other approaches to the HSP using the so-called standard method), our algorithm makes nonadaptive queries to the black box and performs collective postprocessing. Also, similarly to previous analysis of the PGM approach, we can express our success probability in terms of the number of solutions of a system of polynomial equations.

However, our approach to polynomial interpolation also has significant differences from the PGM approach to the HSP. In particular, we introduce a different way to query the black box that simplifies both the algorithm and its analysis. In the PGM approach, we query the black box on a uniform superposition and then uncompute uniform superpositions over certain sets. For polynomial interpolation, we instead query a carefully-chosen non-uniform superposition of inputs so that the subsequent uncomputation is classical. Furthermore, the success probability of our method is higher, and its analysis is more straightforward, than

if we used a direct analog of the PGM approach. We hope that these techniques will prove useful for other quantum algorithms, perhaps for the hidden subgroup problem or for other applications of the PGM approach [5, 7].

We also show that our strategy is precisely optimal: for any number of queries k, we describe a k-query algorithm with the highest possible success probability. We give a simple algebraic characterization of this success probability, as follows.

▶ **Theorem 1.** *The maximum success probability of any k-query quantum algorithm for interpolating a polynomial of degree d over $\mathbb{F}_q$ is $|R_k|/q^{d+1}$, where $R_k := Z(\mathbb{F}_q^k \times \mathbb{F}_q^k)$ is the range of the function $Z\colon \mathbb{F}_q^k \times \mathbb{F}_q^k \to \mathbb{F}_q^{d+1}$ defined by $Z(x,y)_j := \sum_{i=1}^k y_i x_i^j$ for $j \in \{0,1,\ldots,d\}$.*

We present an explicit quantum algorithm that achieves this success probability, and we show that no algorithm can do better. We establish optimality with an argument based on the dimension of the space spanned by the possible output states, which appears to be distinct from arguments using the two main approaches to proving limitations on quantum algorithms, the polynomial and adversary methods. Instead, our approach is closely related to a linear-algebraic lower bound technique of Radhakrishnan, Sen, and Venkatesh [16] and to the "rank method" of Boneh and Zhandry [3].

We characterize the query complexity by proving bounds on $|R_k|$, as follows.

▶ **Theorem 2.** *For any fixed positive integer d, the success probability of Theorem 1 is*
(i) *$|R_k|/q^{d+1} = \frac{1}{k!}(1 - O(1/q))$ if d is odd and $k = \frac{d}{2} + \frac{1}{2}$, and*
(ii) *$|R_k|/q^{d+1} = 1 - O(1/q)$ if d is even and $k = \frac{d}{2} + 1$.*

To show the former bound, we explicitly characterize the possible $(x,y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$ such that $Z(x,y)$ takes a particular value. We prove the latter bound in a completely different way, using a second moment argument.

Theorem 2 shows that the success probability has a sharp transition as a function of k, from subconstant for $k < d/2 + 1/2$ (by known lower bounds [9, 12]), to a (d-dependent) constant for $k = d/2 + 1/2$, to $1 - o(1)$ for $k = d/2 + 1$. Note that since k must be an integer, the success probability varies differently with k depending on whether d is odd or even. For fixed even d, $k = d/2 + 1$ queries give success probability $1 - o(1)$, whereas $k = d/2$ queries give success probability $o(1)$. For fixed odd d, the success probability is $o(1)$ for $k = d/2 - 1/2$ and constant for $k = d/2 + 1/2$. To achieve higher success probability, we can make $k = d/2 + 3/2$ queries and treat f as a polynomial of degree $d + 1$ with $c_{d+1} = 0$, giving success probability $1 - o(1)$.

In light of these results, polynomial interpolation is reminiscent of the task of computing the parity of n bits, where the classical query complexity is n (even for bounded error) and the quantum query complexity is $n/2$ [2, 8]. More generally, a similar factor-of-two improvement is possible for the oracle interrogation problem, where the goal is to learn the entire n-bit string encoded by a black box [18]. However, polynomial interpolation is qualitatively different in that the oracle returns values over $\mathbb{F}_q$ rather than $\mathbb{F}_2$. Note that for the oracle interrogation problem over $\mathbb{F}_q$, one can only achieve speedup by a factor of about $1 - 1/q$ [3]*Section 4, which is negligible for large q.

Our algorithm improves results of Boneh and Zhandry giving quantum attacks on certain cryptographic protocols [3]. For a version of the Shamir secret sharing scheme [17] where the shares can be quantum superpositions, their d-query interpolation algorithm shows that a subset of only d parties can recover the secret. Our algorithm considerably strengthens this, showing that a subset of $d/2 + 1/2$ parties can recover the secret with constant probability,

and $d/2+1$ can recover it with probability $1-O(1/q)$. Boneh and Zhandry also formulate a model of quantum message-authentication codes (MACs), where the goal is to tag messages to authenticate the sender. Informally, a MAC is called d-time if, given the ability to create d valid message-tag pairs, an attacker cannot forge another valid message-tag pair. Boneh and Zhandry show that there are $(d+1)$-wise independent functions that are not d-time quantum MACs. Our result improves this to show that there are $(d+1)$-wise independent functions that are not $(d/2+1/2)$-time quantum MACs.

Finally, we consider the gate complexity of polynomial interpolation. We call an algorithm *gate-efficient* if it can be implemented with a number of 2-qubit gates that is only larger than its query complexity by a factor of $\mathrm{poly}(\log q)$. We construct a gate-efficient variant of our algorithm that achieves almost the same success probability. (Note that while our algorithm for $k=d/2+1/2$ has gate complexity polynomial in both $\log q$ and d, the algorithm for $k=d/2+1$ has gate complexity $k!\,\mathrm{poly}(\log q)$. Improving the dependence on d is a natural open question.)

▶ **Theorem 3.** *For any fixed positive integer d, there is a gate-efficient quantum algorithm for interpolating a polynomial of degree d over $\mathbb{F}_q$ using*

(i) $k=\frac{d}{2}+\frac{1}{2}$ *queries, succeeding with probability* $\frac{1}{k!}(1-O(1/q))$*, if d is odd; and*

(ii) $k=\frac{d}{2}+1$ *queries, succeeding with probability* $1-o(1)$*, if d is even.*

The main step in implementing the algorithm is to invert the function Z described in the statement of Theorem 1, i.e., to find some $x,y\in\mathbb{F}_q^k$ so that $Z(x,y)$ takes a given value. We achieve this by characterizing the solutions in terms of a polynomial equation and a system of linear equations.

In Section 5 we discuss the more general case where $f\in\mathbb{F}_q[X_1,\dots,X_n]$ is a multivariate polynomial of degree d. While our algorithm generalizes straightforwardly, the analysis of its success probability is more complicated. We conjecture that the quantum query complexity of this problem is smaller than the classical query complexity by a factor of $n+1$.

The remainder of the paper is organized as follows. After introducing some definitions in Section 2.1, we describe our k-query algorithm in Section 2.2. We analyze the success probability of this algorithm for $k=d/2+1/2$ in Section 2.3, and for $k=d/2+1$ in Section 2.4. We also show in Section 2.5 that essentially the same performance can be achieved using k independent queries to the oracle, each on a uniform superposition of inputs (which might make some cryptographic attacks easier, depending on the model). We establish optimality of our algorithm in Section 3. In Section 4, we describe the gate-efficient version of our algorithm. Finally, we conclude in Section 5 with a brief discussion of some open questions.

2 Quantum algorithm for polynomial interpolation

2.1 Preliminaries

Let $f(X)=c_dX^d+\cdots+c_1X+c_0\in\mathbb{F}_q[X]$ be an unknown polynomial of degree d that is specified by the vector of coefficients $c\in\mathbb{F}_q^{d+1}$, where $q=p^r$ a power of a prime p. Access to f is provided by a black box acting as $|x,y\rangle\mapsto|x,y+f(x)\rangle$ for all $x,y\in\mathbb{F}_q$.

Let $e\colon\mathbb{F}_q\to\mathbb{C}$ be the exponential function $e(z)=\mathrm{e}^{2\pi\mathrm{i}\,\mathrm{Tr}(z)/p}$, where the trace function $\mathrm{Tr}\colon\mathbb{F}_q\to\mathbb{F}_p$ is defined by $\mathrm{Tr}(z)=z+z^p+z^{p^2}+\cdots+z^{p^{r-1}}$. The Fourier transform over $\mathbb{F}_q$ is the unitary transformation acting as $|x\rangle\mapsto\frac{1}{\sqrt{q}}\sum_{y\in\mathbb{F}_q}e(xy)|y\rangle$ for all $x\in\mathbb{F}_q$.

We can compute the value of f into the phase by Fourier transforming the second query register. If we apply the inverse Fourier transform, perform a query, and then apply the

Fourier transform, we have the transformation

$$|x,y\rangle \mapsto \frac{1}{\sqrt{q}} \sum_{z\in\mathbb{F}_q} e(-yz)|x,z\rangle \mapsto \frac{1}{\sqrt{q}} \sum_{z\in\mathbb{F}_q} e(-yz)|x,z+f(x)\rangle \tag{1}$$

$$\mapsto \frac{1}{q} \sum_{z,w\in\mathbb{F}_q} e(-yz+(z+f(x))w)|x,w\rangle = e(yf(x))|x,y\rangle \tag{2}$$

for any $x,y\in\mathbb{F}_q$, where we used the fact that $\sum_{z\in\mathbb{F}_q} e(zv) = q\delta_{z,v}$. We call the transformation $|x,y\rangle \mapsto e(yf(x))|x,y\rangle$ a *phase query*. Since a phase query can be implemented with a single standard query and vice versa, the query complexity of a problem does not depend on which type of query we use.

For vectors $x,y\in\mathbb{F}_q^k$, we denote the inner product over $\mathbb{F}_q$ by $x\cdot y := \sum_{i=1}^k x_i y_i$. The k-fold Fourier transform (i.e., the Fourier transform acting independently on each register) acts as $|x\rangle \mapsto \frac{1}{\sqrt{q^k}} \sum_{y\in\mathbb{F}_q^k} e(x\cdot y)|y\rangle$ for any $x\in\mathbb{F}_q^k$.

2.2 The algorithm

We now describe our algorithm for polynomial interpolation. An ideal algorithm would produce the Fourier transform of the coefficient vector $c\in\mathbb{F}_q^{d+1}$, that is, the state

$$|\hat{c}\rangle = \frac{1}{\sqrt{q^{d+1}}} \sum_{z\in\mathbb{F}_q^{d+1}} e(c\cdot z)|z\rangle. \tag{3}$$

Instead we use k quantum queries to create the approximate state

$$|\hat{c}_{R_k}\rangle := \frac{1}{\sqrt{|R_k|}} \sum_{z\in R_k} e(c\cdot z)|z\rangle \tag{4}$$

for some set $R_k\subseteq\mathbb{F}_q^{d+1}$. A measurement of this state in the Fourier basis gives c with probability $|\langle\hat{c}_{R_k}|\hat{c}\rangle|^2 = |R_k|/q^{d+1}$.

Our algorithm performs k phase queries in parallel, each acting on a separate register. On input $|x,y\rangle$ for $x,y\in\mathbb{F}_q^k$, these k queries introduce the phase $e(\sum_{i=1}^k y_i f(x_i))$. To define the set R_k, recall the function $Z\colon \mathbb{F}_q^k\times\mathbb{F}_q^k\to\mathbb{F}_q^{d+1}$ defined by

$$Z(x,y)_j := \sum_{i=1}^k y_i x_i^j \text{ for } j\in\{0,1,\ldots,d\}. \tag{5}$$

Then we have $\sum_{i=1}^k y_i f(x_i) = \sum_{i=1}^k \sum_{j=0}^d y_i c_j x_i^j = c\cdot Z(x,y)$ for all $x,y\in\mathbb{F}_q^k$. The range $R_k := Z(\mathbb{F}_q^k\times\mathbb{F}_q^k)$ of the function Z is the set

$$R_k = \{Z(x,y) : (x,y)\in\mathbb{F}_q^k\times\mathbb{F}_q^k\}\subseteq\mathbb{F}_q^{d+1}. \tag{6}$$

For each $z\in R_k$ we choose a unique $(x,y)\in\mathbb{F}_q^k\times\mathbb{F}_q^k$ such that $Z(x,y)=z$. Let $T_k\subseteq\mathbb{F}_q^k\times\mathbb{F}_q^k$ be the set of these representatives. Clearly, $Z\colon T_k\to R_k$ is a bijection.

To create the state $|\hat{c}_{R_k}\rangle$, we prepare a uniform superposition over T_k, perform k phase queries, and compute Z in place (i.e., perform the unitary transformation $|x,y\rangle\mapsto|Z(x,y)\rangle$), giving

$$\frac{1}{\sqrt{|T_k|}} \sum_{(x,y)\in T_k} |x,y\rangle \mapsto \frac{1}{\sqrt{|T_k|}} \sum_{(x,y)\in T_k} e(c\cdot Z(x,y))|x,y\rangle \mapsto \frac{1}{\sqrt{|R_k|}} \sum_{z\in R_k} e(c\cdot z)|z\rangle. \tag{7}$$

The above procedure is a k-query algorithm for polynomial interpolation that succeeds with probability $|R_k|/q^{d+1}$, establishing the lower bound on the success probability stated in Theorem 1. To analyze the algorithm, it remains to lower bound $|R_k|$ as a function of k.

2.3 Performance using d/2 + 1/2 queries

We now consider the performance of the above algorithm using $k = d/2 + 1/2$ queries. Let

$$Z^{-1}(z) = \{(x, y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k : Z(x, y) = z\} \tag{8}$$

be the set of those $(x, y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$ corresponding to a particular $z \in \mathbb{F}_q^{d+1}$. Clearly $|R_k|$ is the number of values of z such that $Z^{-1}(z)$ is nonempty. To analyze this, we focus on "good" values of (x, y). Define $X_k^{\text{good}} := \{x \in \mathbb{F}_q^k : x_i \neq x_j \; \forall\, i \neq j\}$ and $Y_k^{\text{good}} := (\mathbb{F}_q^\times)^k$ and let $Z^{-1}(z)^{\text{good}} := Z^{-1}(z) \cap (X_k^{\text{good}} \times Y_k^{\text{good}})$. We claim the following:

▶ **Lemma 4.** *If $k = d/2 + 1/2$, then for all $z \in \mathbb{F}_q^{d+1}$, either $|Z^{-1}(z)^{\text{good}}| = 0$ or $|Z^{-1}(z)^{\text{good}}| = k!$.*

Proof. See the proof in the extended version [6] of this article. ◀

Using Lemma 4, we can show that $k = d/2 + 1/2$ queries suffice to perform polynomial interpolation with probability that is independent of q, but that decreases with d.

Proof of Theorem 2(i): $k = d/2 + 1/2$**.** We have $|X_k^{\text{good}}| = q!/(q-k)!$ and $|Y_k^{\text{good}}| = (q-1)^k$, so

$$\sum_{z \in \mathbb{F}_q^{d+1}} |Z^{-1}(z)^{\text{good}}| = |X_k^{\text{good}}| \cdot |Y_k^{\text{good}}| = \frac{q!}{(q-k)!}(q-1)^k = q^{2k}(1 - O(1/q)). \tag{9}$$

Thus, invoking Lemma 4, the number of values of z for which $|Z^{-1}(z)^{\text{good}}| = k!$ is at least $\frac{q^{2k}}{k!}(1 - O(1/q))$. Since $k = d/2 + 1/2$, it follows that $|R_k|/q^{d+1}$ is at least $\frac{1}{k!}(1 - O(1/q))$, as claimed. ◀

2.4 Performance using d/2 + 1 queries

Next we show that with more than $d/2 + 1/2$ queries, the success probability approaches 1 for large q.

Proof of Theorem 2(ii): $k = d/2 + 1$**.** Under the uniform distribution on $z \in \mathbb{F}_q^{d+1}$, we have

$$|R_k|/q^{d+1} = 1 - \Pr[|Z^{-1}(z)| = 0]. \tag{10}$$

We use a second moment argument to upper bound the number of $z \in \mathbb{F}_q^{d+1}$ for which $|Z^{-1}(z)| = 0$. The mean of $|Z^{-1}(z)|$ is $\mu := q^{-(d+1)} \sum_{z \in \mathbb{F}_q^{d+1}} |Z^{-1}(z)| = q^{2k-(d+1)}$. Let $\delta[\mathcal{P}]$ be 1 if $\mathcal{P}$ is true and 0 if $\mathcal{P}$ is false. For the second moment, we compute

$$\sum_{z \in \mathbb{F}_q^{d+1}} |Z^{-1}(z)|^2 = \sum_{u,v,x,y \in \mathbb{F}_q^k} \delta[Z(u, v) = Z(x, y)] \tag{11}$$

$$= \sum_{u,v,x,y \in \mathbb{F}_q^k} \frac{1}{q^{d+1}} \sum_{\lambda \in \mathbb{F}_q^{d+1}} e(\lambda \cdot (Z(u, v) - Z(x, y))) \tag{12}$$

$$= \frac{q^{4k}}{q^{d+1}} + \frac{1}{q^{d+1}} \sum_{\lambda \in \mathbb{F}_q^{d+1} \setminus (0,\ldots,0)} \left(\sum_{x,y \in \mathbb{F}_q} e\left(y \sum_{j=0}^{d} \lambda_j x^j \right) \right)^{2k} \tag{13}$$

$$= q^{4k-(d+1)} + \frac{1}{q^{d+1}} \sum_{\lambda \in \mathbb{F}_q^{d+1} \setminus (0,\ldots,0)} \left(q \sum_{x \in \mathbb{F}_q} \delta\left[\sum_{j=0}^{d} \lambda_j x^j = 0 \right] \right)^{2k} \tag{14}$$

$$\leq q^{4k-(d+1)} + (qd)^{2k}. \tag{15}$$

Thus for the variance, we have

$$\sigma^2 := \frac{1}{q^{d+1}} \sum_{z \in \mathbb{F}_q^{d+1}} |Z^{-1}(z)|^2 - \mu^2 \leq \frac{(qd)^{2k}}{q^{d+1}}. \tag{16}$$

(note that $\sigma^2 \geq 0$ by the Cauchy inequality). Applying the Chebyshev inequality, we find

$$\Pr[Z^{-1}(z) = 0] \leq \frac{\sigma^2}{\mu^2} \leq \frac{(qd)^{2k}/q^{d+1}}{q^{4k-2(d+1)}} = d^{2k} q^{d+1-2k}. \tag{17}$$

Therefore $|R_k|/q^{d+1} = 1 - \Pr[Z^{-1}(z) = 0] \geq 1 - d^{2k} q^{d+1-2k}$. With $k = d/2 + 1$, we have

$$|R_k|/q^{d+1} \geq 1 - d^{2k}/q = 1 - O(1/q) \tag{18}$$

as claimed. ◀

Note that one can improve the dependence on d in (16) using results on the distribution of zeros in random polynomials [10].

2.5 An alternative algorithm

The algorithm described above queries the oracle nonadaptively, that is, all k queries can be performed in parallel. However, the input state to these queries is correlated across all k copies. In this section, we describe an alternative algorithm that queries the black box on a state that is independent and identical for each of the k queries, namely, a uniform superposition over all inputs. This algorithm is suboptimal, but its performance is not significantly worse than that of the optimal algorithm described in Section 2.2.

Analogous to the so-called standard method for the hidden subgroup problem, querying f on a uniform superposition gives the state $\frac{1}{\sqrt{q}} \sum_{x \in \mathbb{F}_q^k} |x, f(x)\rangle$. If we use k queries to prepare k copies of this state and then perform the Fourier transform on the second register (or equivalently, perform k independent phase queries), we obtain the state

$$\frac{1}{q^k} \sum_{x,y \in \mathbb{F}_q^k} e(c \cdot Z(x,y)) |x, y\rangle = \frac{1}{q^k} \sum_{z \in \mathbb{F}_q^{d+1}} e(c \cdot z) \sqrt{|Z^{-1}(z)|}\, |Z^{-1}(z)\rangle \tag{19}$$

where $|Z^{-1}(z)\rangle := \sum_{(x,y) \in Z^{-1}(z)} |x, y\rangle / |Z^{-1}(z)|^{1/2}$. Motivated by the PGM approach to the hidden subgroup problem [1], suppose we perform the transformation $|Z^{-1}(z)\rangle \mapsto |z\rangle$, giving the state

$$|\phi_k^c\rangle := \frac{1}{q^k} \sum_{z \in \mathbb{F}_q^{d+1}} e(c \cdot z) \sqrt{|Z^{-1}(z)|}\, |z\rangle. \tag{20}$$

Measuring this state in the Fourier basis gives the outcome c with probability

$$|\langle \phi_k^c | \hat{c} \rangle|^2 = \frac{1}{q^{2k+d+1}} \left(\sum_{z \in \mathbb{F}_q^{d+1}} \sqrt{|Z^{-1}(z)|} \right)^2. \tag{21}$$

If $k = d/2 + 1/2$, we claim that this algorithm succeeds with constant probability. From the proof of Theorem 2 for $k = d/2 + 1/2$, we have that $|Z^{-1}(z)| \geq k!$ for at least $\frac{q^{2k}}{k!}(1 - O(1/q))$ values of z. Therefore the success probability is at least $\frac{1}{k!}(1 - O(1/q))$.

If $k = d/2 + 1$, then this algorithm succeeds with probability that approaches 1 for large q. To see this, recall from the proof of Theorem 2 for $k = d/2 + 1$ that, under a uniform

distribution over $z \in \mathbb{F}_q^{d+1}$, the quantity $Z^{-1}(z)$ has mean $\mu = q$ and standard deviation $\sigma = \sqrt{q}d^k$. Thus, by the Chebyshev inequality, we have

$$\Pr\left[|Z^{-1}(z)| \leq q - \alpha\sqrt{q}d^k\right] \leq \frac{1}{\alpha^2}. \tag{22}$$

It follows that $|\langle\phi_k^c|\hat{c}\rangle|^2 \geq (1 - \frac{\alpha d^k}{\sqrt{q}})(1 - \frac{1}{\alpha^2})^2$ Choosing $\alpha = \Theta(q^{1/6})$, this gives a success probability of $|\langle\phi_k^c|\hat{c}\rangle|^2 = 1 - O(q^{-1/3})$, which approaches 1 for large q.

3 Optimality

In this section, we show that the query complexity of our algorithm is precisely optimal: no k-query algorithm can succeed with a probability larger than $|R_k|/q^{d+1}$. We begin with a basic result showing that m states spanning an n-dimensional subspace can be distinguished with probability at most n/m.

▶ **Lemma 5.** *Suppose we are given a state $|\psi_c\rangle$ with $c \in C$ chosen uniformly at random. Then the probability of correctly determining c with some orthogonal measurement is at most* $\dim \operatorname{span}\{|\psi_c\rangle : c \in C\}/|C|$.

Proof. Consider a measurement with orthogonal projectors E_c, and let Π denote the projection onto $\operatorname{span}\{|\psi_c\rangle : c \in C\}$. Then we have that Pr[success] equals

$$\frac{1}{|C|}\sum_{c\in C}\langle\psi_c|E_c|\psi_c\rangle \leq \frac{1}{|C|}\sum_{c\in C}\operatorname{tr}(E_c\Pi) = \frac{\operatorname{tr}(\Pi)}{|C|} = \frac{\dim \operatorname{span}\{|\psi_c\rangle : c \in C\}}{|C|} \tag{23}$$

as claimed. ◀

We apply this lemma where $|\psi_c\rangle$ is the final state of a given quantum query algorithm when the black box contains $c \in \mathbb{F}_q^{d+1}$. There is no loss of generality in considering an orthogonal measurement at the end of the algorithm since we allow the use of an arbitrary-sized ancilla.

▶ **Lemma 6.** *Let $|\psi_c\rangle$ be the state of any quantum polynomial interpolation algorithm after k queries, where the black box contains $c \in \mathbb{F}_q^{d+1}$. Then* $\dim \operatorname{span}\{|\psi_c\rangle : c \in \mathbb{F}_q^{d+1}\} \leq |R_k|$.

Proof. See the proof in the extended version [6] of this article. ◀

We can now prove our upper bound on the success probability of quantum algorithms for polynomial interpolation.

Proof of Theorem 1 (upper bound on success probability). By combining Lemma 5 with Lemma 6, we see that if the coefficients $c \in \mathbb{F}_q^{d+1}$ are chosen uniformly at random, no algorithm can succeed with probability greater than $|R_k|/q^{d+1}$. Since the minimum cannot be larger than the average, this implies a lower bound on the success probability in the worst case of $|R_k|/q^{d+1}$. ◀

This result also shows that the exact quantum query complexity of polynomial interpolation is maximal.

▶ **Corollary 7.** *The exact quantum query complexity of interpolating a degree-d polynomial is $d + 1$.*

Proof. See the proof in the extended version [6] of this article. ◀

4 Gate complexity

In Section 2, we analyzed the query complexity of our polynomial interpolation algorithm. Here we describe a $(d/2+1/2)$-query algorithm whose gate complexity is $\mathrm{poly}(\log q)$, and whose success probability is close to that of the best algorithm using this number of queries (in particular, for fixed d it still succeeds with constant probability). We also give an algorithm for the case $k = d/2+1$ whose gate complexity is larger by a factor of $\mathrm{poly}(\log q)$, but with an additional factor of $k!$.

4.1 Algorithm for k = d/2 + 1/2 queries

To simplify the computation of unique representatives of values $z \in R_k$, we restrict attention to the "good" case considered in Section 2.3. Let

$$R_k^{\mathrm{good}} := \{Z(x,y) : x \in X_k^{\mathrm{good}},\, y \in Y_k^{\mathrm{good}}\}. \tag{24}$$

For any $z \in R_k^{\mathrm{good}}$, we show how to efficiently compute representative values $x \in X_k^{\mathrm{good}}$ and $y \in Y_k^{\mathrm{good}}$ with $Z(x,y) = z$, defining a set of representatives T_k^{good}. Then we consider an algorithm as described in Section 2.2, but with R_k replaced by R_k^{good} and T_k replaced by T_k^{good}. Clearly the success probability of this algorithm is $|R_k^{\mathrm{good}}|/q^{d+1}$. Our lower bound on $|R_k|$ in Section 2.3 was actually a bound on $|R_k^{\mathrm{good}}|$, so this algorithm still succeeds with probability $\frac{1}{k!}(1+O(1/q))$.

To give a gate-efficient algorithm, it suffices to show how to efficiently compute the function $Z^{-1}\colon R_k^{\mathrm{good}} \to T_k^{\mathrm{good}}$ (that is, to compute this function using $\mathrm{poly}(\log q)$ gates).

▶ **Lemma 8.** *Suppose there is an efficient algorithm to compute $Z^{-1}\colon R_k^{\mathrm{good}} \to T_k^{\mathrm{good}}$. Then the algorithm of Section 2.2 can be made gate-efficient (with R_k replaced by R_k^{good} and T_k by T_k^{good}).*

Proof. It is trivial to compute $Z\colon T_k^{\mathrm{good}} \to R_k^{\mathrm{good}}$ efficiently. Given an efficient procedure for computing $Z^{-1}\colon R_k^{\mathrm{good}} \to T_k^{\mathrm{good}}$, this gives us the ability to efficiently compute Z in place (that is, to perform the transformation $|x,y\rangle \mapsto |z\rangle$ as required by the algorithm). To do this, we first compute z in an ancilla register by evaluating Z (which only requires arithmetic over $\mathbb{F}_q$) and then uncompute (x,y) by applying the circuit for Z^{-1} in reverse.

It remains to prepare the initial uniform superposition over T_k^{good}. This can also be done using the ability to compute Z^{-1}. Suppose we create a uniform superposition over all of $z \in \mathbb{F}_q^{d+1}$ and then attempt to compute Z^{-1}. If $z \notin R_k^{\mathrm{good}}$, this is detected, and we can set a flag qubit indicating failure. Thus we can prepare a state of the form

$$\frac{1}{\sqrt{q^{d+1}}}\left(\sum_{(x,y)\in T_k^{\mathrm{good}}} |Z(x,y),0,x,y\rangle + \sum_{z\in\mathbb{F}_q^{d+1}\setminus R_k^{\mathrm{good}}} |z,1,0,0\rangle\right). \tag{25}$$

A measurement of the flag qubit gives the outcome 0 with probability $|R_k^{\mathrm{good}}|/q^{d+1}$. Since this is our lower bound on the success probability of the overall algorithm, we do not have to repeat this process too many times before we successfully prepare the initial state (and by sufficiently many repetitions, we can make the error probability arbitrarily small). When the measurement succeeds, we can uncompute the first register to obtain the state $\sum_{(x,y)\in T_k^{\mathrm{good}}} |x,y\rangle/|T_k^{\mathrm{good}}|^{1/2}$ as desired. ◀

In the remainder of this section, we describe how to efficiently compute $Z^{-1}(z)$ for $z \in R_k^{\mathrm{good}}$. Our approach appeals to "Prony's method" [14] (a precursor to Fourier analysis)

and the theory of linear recurrences. We start with the following technical result, where e_j denotes the jth elementary symmetric polynomial in k variables, i.e.,

$$e_j(x_1, \ldots, x_k) = \sum_{1 \le i_1 < i_2 < \cdots < i_j \le k} x_{i_1} x_{i_2} \cdots x_{i_j}. \tag{26}$$

▶ **Lemma 9.** *We have* $x_i^k = -\sum_{j=1}^{k} x_i^{k-j}(-1)^j e_j(x_1, \ldots, x_k)$ *for all* $i \in \{1, \ldots, k\}$.

Proof. See the proof in the extended version [6] of this article. ◀

Using this fact, we can show that each component of $Z(x, y)$ satisfies a kth-order linear recurrence.

▶ **Lemma 10.** *If* $z_j = \sum_{i=1}^{k} y_i x_i^j$ *for all nonnegative integers* j*, then we have for all nonnegative integers* n *that* $z_{n+k} = -\sum_{j=0}^{k-1}(-1)^{k-j} e_{k-j}(x_1, \ldots, x_k) z_{n+j}$.

Proof. See the proof in the extended version [6] of this article. ◀

We are now ready to describe the gate-efficient algorithm for polynomial interpolation.

Proof of Theorem 3(i): $k = d/2 + 1/2$**.** By Lemma 8, it suffices to give an efficient algorithm for computing a representative $(x, y) \in Z^{-1}(z)^{\text{good}}$ for any given $z \in R_k^{\text{good}}$. See the proof in the extended version [6] of this article. ◀

4.2 Algorithm for k = d/2 + 1 queries

We now present a similar algorithm for the case $k = d/2 + 1$ that also has gate complexity $\text{poly}(\log q)$, although it has more overhead as a function of d.

To apply the approach of Section 4.1, we again focus on solutions of $Z(x, y) = z$ with $(x, y) \in X^{\text{good}} \times Y^{\text{good}}$. However, recall that our lower bound on the success probability for $k = d/2 + 1$ in Section 2.4 used all solutions $(x, y) \in \mathbb{F}_q^k \times \mathbb{F}_q^k$. Thus we begin by showing that the success probability of the algorithm remains close to 1 even when restricted to good solutions.

▶ **Lemma 11.** *If* $k = d/2 + 1$*, then* $|R_k^{\text{good}}|/q^{d+1} = 1 - O(1/q)$.

Proof. See the proof in the extended version [6] of this article. ◀

Now consider the problem of computing a value $(x, y) \in X^{\text{good}} \times Y^{\text{good}}$ such that $Z(x, y) = z$ for some given $z \in R_k^{\text{good}}$. We can approach this task using the strategy outlined in the proofs of the lemmas of Section 4.1, which can be found in the extended version [6] of this article.

We claim that choosing a random $z_{d+1} \in \mathbb{F}_q$ gives a solution with probability nearly $1/k!$.

▶ **Lemma 12.** *Suppose* $z = (z_0, \ldots, z_d)$ *is chosen uniformly at random from* $\mathbb{F}_q^{d+1}$*. Then with probability* $1 - o(1)$ *(over the choice of* z*), choosing* z_{d+1} *uniformly at random from* $\mathbb{F}_q$ *and solving for* $(x, y) \in Z^{-1}(z)^{\text{good}}$ *as in the proof of Theorem 3(ii) gives a solution with probability* $(1 - o(1))/k!$ *(over the choice of* z_{d+1}*).*

Proof. See the proof in the extended version [6] of this article. ◀

Lemma 12 gives a method for computing a representative $(x, y) \in X^{\text{good}} \times Y^{\text{good}}$ such that $Z(x, y) = z$: simply choose $z_{d+1} \in \mathbb{F}_q$ at random until we find a solution. Repeating this process $O(k!)$ times suffices to find a solution with constant probability (for almost all z). However, since this approach constructs a random $(x, y) \in Z^{-1}(z)^{\text{good}}$ rather than a unique representative, it does not define a set T_k^{good}, and it cannot be directly applied to our quantum algorithm as described so far. Instead, we construct an equivalent algorithm that represents the sets $Z^{-1}(z)^{\text{good}}$ using quantum superpositions.

▶ **Lemma 13.** *Suppose there is an efficient algorithm to generate the quantum state*

$$|Z^{-1}(z)^{\text{good}}\rangle := \frac{1}{\sqrt{|Z^{-1}(z)^{\text{good}}|}} \sum_{(x,y)\in Z^{-1}(z)^{\text{good}}} |x, y\rangle \tag{27}$$

for any given $z \in R_k^{\text{good}}$. Then there is a gate-efficient k-query quantum algorithm for the polynomial interpolation problem, succeeding with probability $|R_k^{\text{good}}|/q^{d+1}$.

Proof. We essentially replace $(x, y) \in T_k^{\text{good}}$ by $|Z^{-1}(Z(x, y))^{\text{good}}\rangle$ throughout the algorithm. More concretely, we proceed as follows.

Observe that the ability to perform the given state generation map $|z\rangle \mapsto |z\rangle|Z^{-1}(z)^{\text{good}}\rangle$ implies the ability to perform the in-place transformation

$$|z\rangle \mapsto |Z^{-1}(z)^{\text{good}}\rangle. \tag{28}$$

After applying the state generation map, we simply uncompute the map Z to erase the register $|z\rangle$.

The algorithm begins by creating a uniform superposition over all of $z \in \mathbb{F}_q^{d+1}$ and applying the map (28). As in the proof of Lemma 8, we can detect whether $z \notin R_k^{\text{good}}$, and we can postselect on the outcomes for which $z \in R_k^{\text{good}}$ with reasonable overhead, giving the state $\sum_{z\in R_k^{\text{good}}} |Z^{-1}(z)^{\text{good}}\rangle/|R_k^{\text{good}}|^{1/2}$. Then perform k phase queries and apply the inverse of the transformation (28), giving the state

$$\frac{1}{\sqrt{|R_k^{\text{good}}|}} \sum_{z\in R_k^{\text{good}}} e(c \cdot z)|Z^{-1}(z)^{\text{good}}\rangle \mapsto \frac{1}{\sqrt{|R_k^{\text{good}}|}} \sum_{z\in R_k^{\text{good}}} e(c \cdot z)|z\rangle. \tag{29}$$

As discussed in Section 2.2, measuring this state gives c with probability $|R_k^{\text{good}}|/q^{d+1}$. ◀

Finally, we show how to prepare $|Z^{-1}(z)^{\text{good}}\rangle$ and thereby give a gate-efficient quantum algorithm for polynomial interpolation with $k = d/2 + 1$ queries.

Proof of Theorem 3(ii): $k = d/2 + 1$. We use $|Z^{-1}(z)^{\text{good}}\rangle$ as a quantum representative of the set of solutions $Z^{-1}(z)^{\text{good}}$ as described in Lemma 13. We claim that we can efficiently perform the transformation $|z\rangle \mapsto |Z^{-1}(z)^{\text{good}}\rangle$ for a fraction $1 - o(1)$ of those $z \in R_k^{\text{good}}$, which in turn are a fraction $1 - o(1)$ of all $z \in \mathbb{F}_q^{d+1}$ (by Lemma 11), giving the claimed success probability.

To prepare $|Z^{-1}(z)^{\text{good}}\rangle$, we first prepare a uniform superposition over $z_{d+1} \in \mathbb{F}_q$ and use the procedure of Section 4.1 to compute the corresponding (x, y), if it exists. Lemma 12 shows that a fraction $(1 - o(1))/k!$ of the values of z_{d+1} correspond to a valid (x, y), so this process can be boosted to prepare a state close to $|Z^{-1}(z)^{\text{good}}\rangle$ with overhead $O(k!)$ (or with amplitude amplification, $O(\sqrt{k!})$), which in particular is independent of q. We can easily uncompute z_{d+1} given (x, y), giving the desired transformation. ◀

5 Open problems

In this paper, we have precisely characterized the quantum query complexity of polynomial interpolation. We conclude by briefly discussing some possible directions for future work.

In Section 4, we gave an algorithm for the case $k = d/2+1$ whose gate complexity is larger than its query complexity by a factor of $k!\operatorname{poly}(\log q)$. This gate complexity is polynomial in $\log(q)$ but superexponential in d. Is it possible to give an algorithm with gate complexity only $\operatorname{poly}(d, \log q)$?

A natural extension of our results would be to consider the problem of learning a multivariate polynomial $f \in \mathbb{F}_q[X_1, \ldots, X_n]$ of degree at most d. Montanaro gave asymptotically optimal bounds for this problem assuming f is multilinear [13], but it is also natural to consider the more general case where f is not necessarily multilinear. The quantum algorithm described in Section 2.2 can be extended to the multivariate case in a fairly straightforward manner, and we conjecture that it performs as follows.

▶ **Conjecture 14.** *For any fixed positive integers d and n, there exists a k-query quantum algorithm for interpolating a degree-d multivariate polynomial in n variables that, as q grows, has success probability $1 - o(1)$ provided $k > \binom{n+d}{d}/(n+1)$.*

Note that classically one needs $\binom{n+d}{d}$ queries to solve the same problem, so our conjecture states that the quantum query complexity is smaller by a factor of $n+1$. We now discuss why computing the success probability of the quantum algorithm appears to be a difficult problem in algebraic geometry.

Let $f \in \mathbb{F}_q[X_1, \ldots, X_n]$ be of degree at most d. For $j \in \mathbb{N}^n$ and $x \in \mathbb{F}_q^n$, we let $x^j := \prod_{t=1}^n x_t^{j_t}$. To define the set of possible polynomials, we use the set of allowed exponents $\mathcal{J}$ with size

$$J := |\mathcal{J}| := |\{j \in \mathbb{N}^n : j_1 + \cdots + j_n \le d\}| = \binom{n+d}{d}. \tag{30}$$

We now define the function $Z\colon (\mathbb{F}_q^n)^k \times \mathbb{F}_q^k \to \mathbb{F}_q^J$ by $Z(x,y)_j = \sum_{i=1}^k y_i x_i^j$ and consider its range $\mathcal{R}_k := Z((\mathbb{F}_q^n)^k \times \mathbb{F}_q^k) \subseteq \mathbb{F}_q^J$. A straightforward generalization of the univariate interpolation algorithm described in Section 2.2 gives a multivariate interpolation algorithm with success probability $|\mathcal{R}_k|/q^J$. We expect that this algorithm solves the interpolation problem with probability $1 - o(1)$ using $\lfloor J/(n+1) \rfloor + 1$ queries. This would be implied by the following:

▶ **Conjecture 15.** *With $J := \binom{n+d}{d}$ and $\mathcal{R}_k$ as above, we have $|\mathcal{R}_k| = q^J(1 - o(1))$ provided $k > J/(n+1)$.*

Note that this holds for $n = 1$ (according to Lemma 11) and also for $d = 1$. Unfortunately, the approach via exponential sums used in the proof of Lemma 11 only works if $k > J/2$. Thus, while it gives a tight result for $n = 1$, it appears to be inefficient for $n > 1$.

Another way to approach Conjecture 15 is to consider the affine variety $\mathcal{V}_k\colon Z(x,y) = z$ in $kn + k + J$ variables $x \in (\mathbb{F}_q^n)^k$, $y \in \mathbb{F}_q^k$, $z \in \mathbb{F}_q^J$. Clearly $|\mathcal{V}_k(\mathbb{F}_q)| = q^{kn+k}$. It is not hard to show that $\mathcal{V}_k$ is a complete intersection and has only one absolutely irreducible component. Thus it suffices to show that for almost all specializations of $z \in \mathbb{F}_q^J$, the corresponding variety $\mathcal{V}_k(\boldsymbol{z})$ is absolutely irreducible; then provided $k(n+1) > J$, a version of the Lang-Weil bound [11] applies and gives the desired result. Although results of this type are known (see [4, 15] and references therein), unfortunately none of them seems to imply the desired statement. Nevertheless, since a generic variety is absolutely irreducible, the conjecture appears plausible.

References

1 Dave Bacon, Childs, Andrew M., and Wim van Dam. From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups. In *Proceedings of the 46th IEEE Symposium on Foundations of Computer Science*, pages 469–478, 2005. arXiv:arXiv:quant-ph/0504083.

2 Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. arXiv:arXiv:quant-ph/9802049.

3 Dan Boneh and Mark Zhandry. Quantum-secure message authentication codes. In *Proceedings of Eurocrypt*, pages 592–608, 2013.

4 François Charles and Bjorn Poonen. Bertini irreducibility theorems over finite fields. *Journal of the American Mathematical Society*, 29:81–94, 2016.

5 Andrew M. Childs and Wim van Dam. Quantum algorithm for a generalized hidden shift problem. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, pages 1225–1234, 2007. arXiv:arXiv:quant-ph/0507190.

6 Andrew M. Childs, Wim van Dam, Shih-Han Hung, and Igor E. Shparlinski. Optimal quantum algorithm for polynomial interpolation, 2015. arXiv:arXiv:1509.09271v2.

7 Thomas Decker, Jan Draisma, and Pawel Wocjan. Efficient quantum algorithm for identifying hidden polynomials. *Quantum Information and Computation*, 9(3):215–230, 2009. arXiv:arXiv:0706.1219.

8 Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Limit on the speed of quantum computation in determining parity. *Physical Review Letters*, 81(24):5442–5444, 1998. arXiv:arXiv:quant-ph/9802045.

9 Daniel M. Kane and Samuel A. Kutin. Quantum interpolation of polynomials. *Quantum Information and Computation*, 11(1):95–103, 2011. arXiv:arXiv:0909.5683.

10 Arnold Knopfmacher and John Knopfmacher. Counting polynomials with a given number of zeros in a finite field. *Linear and Multilinear Algebra*, 26(4):287–292, 1990.

11 Serge Lang and André Weil. Number of points of varieties in finite fields. *American Journal of Mathematics*, 76:819–827, 1954.

12 David A. Meyer and James Pommersheim. On the uselessness of quantum queries. *Theoretical Computer Science*, 412(51):7068–7074, 2011. arXiv:arXiv:1004.1434.

13 Ashley Montanaro. The quantum query complexity of learning multilinear polynomials. *Information Processing Letters*, 112(11):438–442, 2012. arXiv:arXiv:1105.3310.

14 Gerald M. Pitstick, João R. Cruz, and Robert J. Mulholland. A novel interpretation of Prony's method. *Proceedings of the IEEE*, 76(8):1052–1053, 1988.

15 Bjorn Poonen. Bertini theorems over finite fields. *Annals of Mathematics*, 160:1099–1127, 2004.

16 Jaikumar Radhakrishnan, Pranab Sen, and S. Venkatesh. The quantum complexity of set membership. *Algorithmica*, pages 462–479, 2002. arXiv:arXiv:quant-ph/0007021.

17 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

18 Wim van Dam. Quantum oracle interrogation: Getting all information for almost half the price. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 362–367, 1998. arXiv:arXiv:quant-ph/9805006.

Lower Bounds for the Approximate Degree of Block-Composed Functions*

Justin Thaler

Yahoo! Research, New York, NY, USA

Abstract

We describe a new hardness amplification result for point-wise approximation of Boolean functions by low-degree polynomials. Specifically, for any function f on N bits, define

$$F(x_1, \ldots, x_M) = \mathrm{OMB}(f(x_1), \ldots, f(x_M))$$

to be the function on $M \cdot N$ bits obtained by block-composing f with a function known as ODD-MAX-BIT. We show that, if f requires large degree to approximate to error $2/3$ in a certain one-sided sense (captured by a complexity measure known as *positive one-sided approximate degree*), then F requires large degree to approximate even to error $1 - 2^{-M}$. This generalizes a result of Beigel (Computational Complexity, 1994), who proved an identical result for the special case $f = \mathrm{OR}$.

Unlike related prior work, our result implies strong approximate degree lower bounds even for many functions F that have low *threshold degree*. Our proof is constructive: we exhibit a solution to the dual of an appropriate linear program capturing the approximate degree of any function. We describe several applications, including improved separations between the complexity classes $\mathbf{P}^{\mathbf{NP}}$ and $\mathbf{PP}$ in both the query and communication complexity settings. Our separations improve on work of Beigel (1994) and Buhrman, Vereshchagin, and de Wolf (CCC, 2007).

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases approximate degree, one-sided approximate degree, polynomial approximations, threshold degree, communication complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.17

1 Introduction

Approximate degree and threshold degree are two measures of Boolean function complexity that capture the difficulty of point-wise approximation by low-degree polynomials. The ε-approximate degree of a function $f\colon \{-1,1\}^n \to \{-1,1\}$, denoted $\widetilde{\deg}_\varepsilon(f)$, is the least degree of a polynomial that point-wise approximates f to error ε. The threshold degree, denoted $\deg_\pm(f)$, is the least degree of a real polynomial that agrees in sign with f point-wise.

Approximate degree and threshold degree have found a diverse array of algorithmic and complexity-theoretic applications. On the complexity side, approximate degree lower bounds underlie many tight lower bounds on quantum query complexity [2, 3, 25, 6, 41], and have proven instrumental in resolving a host of long-standing open problems in communication and circuit complexity [40, 39, 35, 42, 16, 44, 34, 41, 14, 12, 13, 27, 8]. On the algorithms side, upper bounds on these complexity measures underlie the fastest known learning algorithms in a number of important models, including the PAC, agnostic, and mistake-bounded models [23, 24, 20, 36]. They also yield fast algorithms for private data release [49, 11].

* The full version of this paper is available at `http://eccc.hpi-web.de/report/2014/150/`.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 17; pp. 17:1–17:15
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Despite these applications, our understanding of approximate and threshold degree remains limited. While tight upper and lower bounds are known for some specific functions, including symmetric functions [32, 38, 15] and certain read-once formulae, few general results are known, and characterizing the approximate and threshold degrees of many simple functions remains open. However, a handful of recent works has established various forms of "hardness amplification" for approximate degree [43, 9, 10, 46, 45, 26, 47]. Roughly speaking, these results show how to take a function f which is hard to approximate by low-degree polynomials in a weak sense, and turn f into a related function F that is hard to approximate by low-degree polynomials in a much stronger sense.

Our Contributions. We extend this recent line of work by establishing a new, generic form of hardness amplification for approximate degree. Unlike prior work, our result implies strong lower bounds even for many functions F that have low threshold degree (e.g., halfspaces). In contrast, analogous hardness amplification results [43, 9, 10, 46, 45, 26, 47] apply only to functions with polynomially large threshold degree. We describe several applications of our result, including an improved separation between the complexity classes $\mathbf{P^{NP}}$ and $\mathbf{PP}$ in both the query and communication complexity settings (see Section 1.3 for details).

We prove our results by constructing explicit *dual polynomials*, which are dual solutions to an appropriate linear program capturing the approximate degree of any function. This "method of dual polynomials" has proven to be a powerful technique for establishing lower bounds on approximate degree. Our construction departs qualitatively from earlier applications of the method, and we believe it to be of interest in its own right. In addition to implying approximate degree lower bounds, dual polynomials have been used to resolve several long-standing open problems in communication complexity, and they yield explicit distributions under which various communication problems are hard [40, 42, 16, 44, 34, 41, 14].

1.1 Overview of Our Results

Let $f : \{-1,1\}^n \to \{-1,1\}$ be a Boolean function. Our hardness amplification method relies heavily on a complexity measure known as *one-sided approximate degree*, or, more precisely, its "positive" and "negative" variants, denoted $\widetilde{\deg}_{+,\varepsilon}(f)$ and $\widetilde{\deg}_{-,\varepsilon}(f)$ respectively. These are intermediate complexity measures that lie between ε-approximate degree and threshold degree, and they have played a central role in recent prior work on hardness amplification for approximate degree [46, 10, 9, 43].[1] Unlike the latter two complexity measures, $\widetilde{\deg}_{+,\varepsilon}(f)$ and $\widetilde{\deg}_{-,\varepsilon}(f)$ treat inputs in $f^{-1}(+1)$ and inputs in $f^{-1}(-1)$ asymmetrically.

In more detail, a polynomial p is said to be a positive one-sided ε-approximation for a Boolean function f if $|p(x) - f(x)| \leq \varepsilon$ for all $x \in f^{-1}(-1)$, and $p(x) \geq 1 - \varepsilon$ for all $x \in f^{-1}(+1)$. The positive one-sided ε-approximate degree of f is the least degree of a positive one-sided ε-approximation for f. Negative one-sided ε-approximate degree is defined analogously. Notice that $\widetilde{\deg}_{+,\varepsilon}(f)$ and $\widetilde{\deg}_{-,\varepsilon}(f)$ are always at most $\widetilde{\deg}_{\varepsilon}(f)$, but can be much smaller. Similarly, $\widetilde{\deg}_{+,\varepsilon}(f)$ and $\widetilde{\deg}_{-,\varepsilon}(f)$ are always at least $\deg_{\pm}(f)$, but can be much larger.

[1] Strictly speaking, the terms positive and negative one-sided approximate degree were introduced by Kanade and Thaler [21], who gave applications of these complexity measures to learning theory. Earlier works on hardness amplification for pointwise approximation by polynomials only used negative one-sided approximate degree, and referred to this complexity measure without qualification as one-sided approximate degree [10, 46]. For our purposes, the distinction between positive and negative one-sided approximate degree is crucial.

Let $\mathrm{OMB} : \{-1,1\}^n \to \{-1,1\}$ denote a specific polynomial size DNF formula known as ODD-MAX-BIT, defined as follows. On input $x = (x_1, \ldots, x_n)$, let i^* denote the largest index such that $x_{i^*} = -1$, and let $i^* = 0$ if no such index exists. We define $\mathrm{OMB}(x_1, \ldots, x_n) = -1$ if i^* is odd, and $\mathrm{OMB}(x_1, \ldots, x_n) = 1$ otherwise. When appropriate, we also use subscripts after function symbols to indicate the number of variables over which the function is defined. Thus, OMB_M denotes the OMB function on M inputs.

For any function $f\colon \{-1,1\}^N \to \{-1,1\}$, define $F : \left(\{-1,1\}^N\right)^M \to \{-1,1\}$ to be the block-composition of OMB_M with f, i.e., $F = \mathrm{OMB}_M(f, \ldots, f)$. Our hardness amplification result establishes that if $\widetilde{\deg}_{+,\varepsilon}(f)$ is large for some ε bounded away from 1, then $\widetilde{\deg}_{+,\varepsilon}(F)$ is large even for ε exponentially close to 1.

▶ **Theorem 1.** *Fix an $f\colon \{-1,1\}^N \to \{-1,1\}$, and let $F = \mathrm{OMB}_M(f, \ldots, f)$. If $\widetilde{deg}_{+,2/3}(f) \geq d$, then $\widetilde{deg}_{+,\varepsilon}(F) \geq d$ for $\varepsilon = 1 - 2^{-M}$.*

A Matching Upper Bound for Theorem 1. To understand the intuition underlying Theorem 1, it is instructive to consider (matching) upper bounds. We begin by giving the well-known sign-representing polynomial for OMB_M itself. Define $p\colon \{-1,1\}^M \to \mathbb{R}$ via

$$p(x_1, \ldots, x_M) := 1 + \sum_{i=1}^{M} (-2)^i \cdot (1 - x_i)/2.$$

It is easy to see that $\mathrm{OMB}_M(x) = \mathrm{sgn}\,(p(x))$, and in fact $2^{-M-1} \cdot p(x)$ approximates OMB_M to error $\varepsilon = 1 - 2^{-M-1}$.

We now turn to constructing approximants for $\mathrm{OMB}_M(f, \ldots, f)$, for an arbitrary inner function f. Fix a $W \geq 2$, and let $q\colon \{-1,1\}^N \to \mathbb{R}$ be any degree d polynomial satisfying the following two properties.

$$q(x) = 0 \quad \text{for all } x \in f^{-1}(+1). \tag{1}$$

$$1 \leq q(x) \leq W - 1 \quad \text{for all } x \in f^{-1}(-1). \tag{2}$$

Denoting an $(M \cdot N)$-bit input as $(x_1, \ldots, x_M) \in \left(\{-1,1\}^N\right)^M$, it is easy to check that

$$F(x_1, \ldots, x_M) = \mathrm{sgn}(h(x_1, \ldots, x_M)), \text{ where } h(x_1, \ldots, x_M) = 1 + \sum_{i=1}^{M} (-W)^i \cdot q(x_i).$$

In fact, $W^{-M-1} \cdot h(x)$ approximates F to error $1 - W^{-M-1}$, and has degree equal to that of q. If $W = O(1)$, then this construction shows that F can be approximated to error $1 - 2^{-O(M)}$ by a degree d polynomial, which matches the error bound of Theorem 1 up to a constant factor in the exponent.

▶ **Observation 2.** *If there exists a polynomial q of degree d satisfying Eq. (1) and Eq. (1) with $W = O(1)$, then $\widetilde{deg}_{\varepsilon}(F) \leq d$ for some $\varepsilon = 1 - 2^{-O(M)}$.*

A few words are in order regarding the relationship between the hypothesis of the upper bound (Observation 2), and the hypothesis of the lower bound (Theorem 1) that $\widetilde{\deg}_{+,2/3}(f) \geq d$. Conditions 1 and 1 together imply that $r(x) := \frac{1}{2W} \cdot (1 - 2q(x))$ is a positive one-sided approximation to f for error parameter $\varepsilon = 1 - \frac{1}{2W}$. Moreover, r has the additional (crucial) property that this approximant is *constant* on inputs in $f^{-1}(+1)$. Observe that the smaller W is, the smaller the error of the one-sided approximant $r(x)$ for $f(x)$, and the smaller the error of the derived approximant $W^{-M-1} \cdot h(x)$ that we constructed for F.

In general, requiring that r be constant on inputs in $f^{-1}(+1)$ is a very stringent condition, which will not be satisfied by all one-sided approximations for f. However, Bun and Thaler [10, Theorem 2] have identified a large class of functions for which *any* one-sided approximation for f can be transformed into one that is constant on inputs in $f^{-1}(+1)$, without increasing its degree. This class includes important functions such as $f = \mathrm{OR}$ (see Section 1.2.2), and $f = \overline{\mathrm{ED}}$, where ED is the well-studied Element Distinctness function that we use in our applications to communication and query complexity. For such functions, Observation 2 implies that Theorem 1 is tight.

Can the Hypothesis in Theorem 1 be Weakened? There are two natural ways to weaken the hypothesis of Theorem 1, and it is natural to wonder whether Theorem 1 would continue to hold under these hypotheses. Specifically, we can ask:

- Does Theorem 1 hold if we replace the outer function OMB_M function with the simpler function OR_M, as in previous hardness amplification results for approximate degree [46, 10, 9, 43]?[2]
- Is a one-sided hardness assumption really essential for Theorem 1 to hold? That is, does OMB_M still amplify the hardness of f if we replace the assumption that $\widetilde{\deg}_{+,2/3}(f) \geq d$ with the weaker assumption that $\widetilde{\deg}_{2/3}(f) \geq d$?

The answer to the first question is no. A counterexample is given by $f = \mathrm{OR}_N$. It is known that $\widetilde{\deg}_{+,2/3}(\mathrm{OR}_N) = \Omega(N^{1/2})$ (see, e.g., [30, 10, 17]), yet $\mathrm{OR}_M(\mathrm{OR}_N, \ldots, \mathrm{OR}_N) = \mathrm{OR}_{N \cdot M}$ can be approximated to error $1 - 1/(MN) \ll 1 - 2^{-M}$ by a polynomial of degree 1. Thus, the use of OMB_M as the "hardness amplifier" is essential to Theorem 1.

The answer to the second question, unfortunately, remains unknown. Formally, we leave the resolution of the following conjecture as an open problem.

▶ **Conjecture 3.** *Suppose that $f\colon \{-1,1\}^N \to \{-1,1\}$ satisfies $\widetilde{deg}_{2/3}(f) \geq d$. Then letting $F = \mathrm{OMB}_M(f, \ldots, f)$, it holds that $\widetilde{deg}_{\varepsilon}(\mathrm{OMB}_M(f, \ldots, f)) \geq d$, for some $\varepsilon = 1 - 2^{-\Omega(M)}$.*

1.2 Technical Comparison to Prior Work

1.2.1 The Method of Dual Polynomials

A dual witness to the statement $\widetilde{\deg}_{\varepsilon}(f) \geq d$ is a non-zero function $\psi : \{-1,1\}^N \to \mathbb{R}$ satisfying two conditions: (a) $\sum_{x \in \{-1,1\}^N} \psi(x) \cdot f(x) \geq \varepsilon \cdot \|\psi\|_1$, where $\|\psi\|_1 = \sum_{x \in \{-1,1\}^N} |\psi(x)|$, and (b) ψ has zero correlation with all polynomials of degree at most d. We refer to Property (a) by saying that ψ is *ε-correlated* with f. We refer to Property (b) by saying that ψ has *pure high degree d*. We refer to ψ as a *dual polynomial* for f.

A dual witness to the statement that $\widetilde{\deg}_{+,\varepsilon}(f) \geq d$ must satisfy an additional correlation condition, namely: (c) $\phi(x)$ agrees in sign with $f(x)$ for all $x \in f^{-1}(+1)$. We refer to Property (c) by saying that ϕ has *positive one-sided error*. (Due to space constraints, we defer further discussion of the duality theory to the full version of the paper.)

We prove Theorem 1 by showing the following: given a dual polynomial ψ_{in} witnessing the assumed $\widetilde{\deg}_{+,2/3}$ lower bound on the inner function f, one can construct an explicit

[2] One may also ask about replacing OMB_M with AND_M in the statement of Theorem 1. Analyses from prior works [10, 46] apply in this case, but show that the resulting function in fact has high threshold degree, and hence is not suitable for our applications to query and communication complexity. We discuss this point in detail in the next section (see Theorem 5, Footnote 5, and the surrounding discussion).

dual polynomial ψ_{comb} witnessing the claimed lower bound on the composed function $F = \text{OMB}(f, \ldots, f)$.

1.2.2 Prior Work on the Approximate Degree of OMB

Beigel [7] proved that for any $d > 0$, there is an $\varepsilon \in 1 - 2^{-\Omega(n/d^2)}$ such that $\widetilde{\deg}_{\varepsilon}(\text{OMB}_n) \geq d$, and used this result[3] to give an oracle separating the (Turing Machine) complexity class **PP** from $\mathbf{P}^{\mathbf{NP}}$. Note that $\text{OMB}_M(\text{OR}_N, \ldots, \text{OR}_N)$ is a sub-function of $\text{OMB}_{M \cdot (2N)}$. As mentioned in Section 1.1, it is known that $\widetilde{\deg}_{+,2/3}(\text{OR}_N) = \Omega(N^{1/2})$. Hence, Theorem 1 can be viewed as a substantial strengthening of Beigel's result: we recover Beigel's lower bound as a special case of Theorem 1 by letting $f = \text{OR}_{d^2}$. Unlike Beigel's proof, which used a non-constructive symmetrization technique, our proof of Theorem 1 constructs an explicit dual polynomial witnessing the lower bound.

For any $\varepsilon > 0$, Klivans and Servedio [24] gave an optimal ε-approximating polynomial for the function OMB, showing that Beigel's lower bound (and hence also our Theorem 1 in the case $f = \text{OR}_N$) is asymptotically tight for all $d > 0$.[4]

1.2.3 Earlier Constructions of Dual Polynomials

Given functions g_M, f_N, Sherstov [45] and Lee [26] independently described a powerful method for constructing a dual polynomial for the composed function $F = g_M(f_N, \ldots, f_N) : \{-1,1\}^{M \cdot N} \to \{-1,1\}$. This method takes a dual polynomial ψ_{in} for f_N, and a dual polynomial ψ_{out} for g, and combines them to obtain a dual polynomial ψ_{comb} for the composed function F. Specifically, denoting an $(M \cdot N)$-bit input as $(x_1, \ldots, x_M) \in \left(\{-1,1\}^N\right)^M$, Sherstov and Lee defined

$$\psi_{\text{comb}}(x_1, \ldots, x_M) = \psi_{\text{out}}\left(\widetilde{\text{sgn}}\left(\psi_{\text{in}}\left(x_1\right)\right), \ldots, \widetilde{\text{sgn}}\left(\psi_{\text{in}}\left(x_M\right)\right)\right) \cdot \prod_{i=1}^{M} |\psi_{\text{in}}(x_i)|. \tag{3}$$

Here, $\widetilde{\text{sgn}} : \mathbb{R} \to \{-1, 0, 1\}$ denotes the function satisfying $\widetilde{\text{sgn}}(t) = 1$ if $t > 0$, $\widetilde{\text{sgn}}(t) = -1$ if $t < 0$, and $\widetilde{\text{sgn}}(0) = 0$.

Recall that for ψ_{comb} to witness a good lower bound for the approximate degree of F, it must be well-correlated with F (Property (a) of Section 1.2.1), and it must have large pure high degree (Property (b) of Section 1.2.1). Sherstov and Lee showed that the pure high degree of ψ_{comb} is multiplicative in the pure high degrees of ψ_{in} and ψ_{out}. That is, if ψ_{in} has pure high degree d_1, and ψ_{out} has pure high degree d_2, then ψ_{comb} has pure high degree $d_1 \cdot d_2$. And while ψ_{comb} is not *in general* well-correlated with the composed function F, several important examples have been identified in which this is the case, as we now explain.

Sherstov [43] and independently Bun and Thaler [9] used the combining technique of Eq. (3) to resolve the (1/3)-approximate degree of the two-level AND-OR tree. Subsequent work by Bun and Thaler [10] used Eq. (3) to establish a hardness amplification result that looks similar to our Theorem 1. Specifically, Bun and Thaler proved:

▶ **Theorem 4** (Bun and Thaler [10]). *Suppose* $\widetilde{deg}_{-,2/3}(f) \geq d$. *Then* $\widetilde{deg}_{-,\varepsilon}(OR_M(f, \ldots, f)) \geq d$, *for* $\varepsilon = 1 - 2^{-M}$.

[3] Beigel describes his result as a lower bound on the *degree-d threshold weight* of OMB_n. However, his argument is easily seen to establish the claimed approximate degree lower bound.

[4] Like Beigel, Klivans and Servedio state their results in terms of degree-d threshold weight. However, their construction is easily seen to imply the claimed upper bound on the approximate degree of OMB_n.

Theorem 4 is identical to our Theorem 1, but for two differences: first, in our Theorem 1, the outer function in the composition is OMB, while in Theorem 4 it is OR. Second, the hypothesis in Theorem 1 is that the inner function f satisfies $\widetilde{\deg}_{+,2/3}(f) \geq d$, while the assumption in Theorem 4 is that $\widetilde{\deg}_{-,2/3}(f) \geq d$. *Both* of these differences are crucial for obtaining a hardness amplification result that applies to functions with low threshold degree (which is essential for our applications to the communication and query complexity described in Section 1.3 below). Indeed, subsequent work by Sherstov refined Theorem 4 to yield a threshold degree lower bound, rather than a $\widetilde{\deg}_{-,\varepsilon}$ lower bound [46].

▶ **Theorem 5** (Sherstov [46]). *Suppose* $\widetilde{deg}_{-,2/3}(f) \geq d$. *Then* $\deg_{\pm}(OR_M(f,\ldots,f)) \geq \min\{d, cM\}$ *for some constant* $c > 0$.[5]

Sherstov gives several proofs of Theorem 5. One proof draws heavily on Eq. (3): he constructs a dual witness of the form $\psi_{\text{comb}} + \psi_{\text{fix}}$, where ψ_{comb} is the dual witness constructed by Bun and Thaler using Eq. (3) to prove Theorem 4, and ψ_{fix} "zeros out" ψ_{comb} on points x such that $0 \neq \widetilde{\mathsf{sgn}}(\psi_{\text{comb}}(x)) \neq \widetilde{\mathsf{sgn}}(\text{OR}_M(f,\ldots,f))$. This ensures that $\psi_{\text{comb}} + \psi_{\text{fix}}$ is perfectly correlated with F.

Sherstov used Theorem 5 to give a depth three circuit with threshold degree $\tilde{\Omega}(n^{2/5})$. He also established the following result, which yields a polynomially stronger lower bound for depth $k > 3$.

▶ **Theorem 6** (Sherstov [46]). *For any* $k \geq 2$, *there is a depth* k *(read-once) Boolean circuit computing a function* F *satisfying* $\deg_{\pm}(F) = \Omega(n^{(k-1)/(2k-1)})$.

Sherstov's proof of Theorem 6 is not a refinement of the proof Theorem 4 from [10]. Rather it relies on an elaborate inductive construction of a dual polynomial (which is nonetheless reminiscent of Eq. (3)).

In the full version of the paper, we explain why any dual witness establishing Theorem 1 must qualitatively depart from the dual witnesses constructed in prior work. In brief, we first argue that the dual witnesses constructed in prior work are implicitly tailored to show optimality of a specific technique for approximating block-composed functions. We then explain that this technique is far from optimal for the functions to which Theorem 1 applies.

1.3 Applications

This section gives an overview of our applications to query and communication complexity. Due to space constraints, formal definitions of the complexity classes involved in these applications, and statements and proofs of the relevant theorems, are deferred to the full version of the paper.

Notation. Given a query or communication model $\mathbf{C}$ and a function f, the notation $\mathbf{C}(f)$ denotes the least cost of a protocol computing f in the model $\mathbf{C}$. Following Babai et al. [4], we define a corresponding complexity class, also denoted $\mathbf{C}$, consisting of all problems that have polylogarithmic cost protocols in the model $\mathbf{C}$. Throughout, we use the superscript cc to denote communication complexity classes, and the subscript query to denote query complexity classes. Any complexity class without a subscript refers to a classical (Turing Machine) class.

[5] By De Morgan's laws and the observation that $\widetilde{\deg}_{-,\varepsilon}(f) = \widetilde{\deg}_{+,\varepsilon}(\overline{f})$, the following is an equivalent formulation of Theorem 5. Suppose that $\widetilde{\deg}_{+,2/3}(f) \geq d$. Then $\deg_{\pm}(\text{AND}_M(f,\ldots,f)) \geq \min\{d, cM\}$ for some constant $c > 0$.

1.3.1 Query Complexity

Connecting Query Complexity, Approximate Degree, and Oracle Separations. A significant motivation for studying query complexity is that separations of query complexity classes immediately yield oracle separations of their classical counterparts. Such oracle separations are sometimes construed as evidence that the same separation applies to the classes' classical counterparts. At a minimum, oracle separations imply a formal barrier (called the *relativization* barrier [5]) to disproving the corresponding Turing Machine separation.

It is well-known that approximate degree lower bounds imply lower bounds on (even quantum) query complexity. So to summarize, approximate degree lower bounds imply query complexity lower bounds, which in turn often imply oracle separations for classical complexity classes.

ODD-MAX-BIT, Counting, and the Polynomial Hierarchy. An important question in complexity theory is to determine the relative power of alternation (as captured by the polynomial-hierarchy **PH**), and counting (as captured by the complexity class #**P** and its decisional variant **PP**). Both **PH** and **PP** generalize **NP** in natural ways. Toda famously showed that their power is related: $\mathbf{PH} \subseteq \mathbf{P}^{\mathbf{PP}}$ [50].

Beigel [7] was interested in determining how much of the Polynomial Hierarchy is contained in **PP** itself, and he set out to give an oracle separating $\mathbf{P}^{\mathbf{NP}}$ from **PP**. To do so, he introduced the function OMB and observed that OMB is in the query complexity of analog of $\mathbf{P}^{\mathbf{NP}}$ – essentially, the query protocol uses the **NP** oracle to perform a binary search for the largest index i^* such that $x_{i^*} = -1$. Then, to show that OMB is not in the query complexity analog of **PP**, Beigel proved a lower bound on the approximate degree of OMB. (Recall from Section 1.2.2 that in [7] Beigel proved that for any $d > 0$, there is an $\varepsilon \in 1 - 2^{-\Omega(n/d^2)}$ such that $\widetilde{\deg}_{\varepsilon}(\mathrm{OMB}_n) \geq d$).

Thus, Beigel's result separated the query complexity classes $\mathbf{PP}_{\mathsf{query}}$ and $\mathbf{P}^{\mathbf{NP}}_{\mathsf{query}}$, and this in turn implied an oracle separating the classical classes **PP** from $\mathbf{P}^{\mathbf{NP}}$.

An Improved Separation for Query Complexity. Quantitatively, Beigel's analysis implies that $\mathbf{PP}_{\mathsf{query}}(\mathrm{OMB}) = \Omega(n^{1/3})$, and prior to our work, this was the best known separation between $\mathbf{PP}_{\mathsf{query}}(f)$ and $\mathbf{P}^{\mathbf{NP}}_{\mathsf{query}}(f)$ for any function f. We improve on this separation by giving a function F in $\mathbf{P}^{\mathbf{NP}}_{\mathsf{query}}$ such that $\mathbf{PP}_{\mathsf{query}}(F) = \tilde{\Omega}(n^{2/5})$.

Details of the separation. The function F we use to exhibit this improved separation is

$$F := \mathrm{OMB}_{n^{2/5}}(\overline{\mathrm{ED}}_{n^{3/5}}, \ldots, \overline{\mathrm{ED}}_{n^{3/5}}), \tag{4}$$

where $\overline{\mathrm{ED}}$ is the negation of the well-studied Element Distinctness function (due to space constraints, we defer a formal definition of the Element Distinctness function to the full version of the paper). Prior work has shown that $\overline{\mathrm{ED}}_N$ satisfies $\widetilde{\deg}_{+,2/3}(\overline{\mathrm{ED}}_N) = \tilde{\Omega}(N^{2/3})$ [10], so Theorem 1 implies that $\widetilde{\deg}_{+,\varepsilon}(F) = \tilde{\Omega}(n^{2/5})$ even for $\varepsilon = 1 - 2^{-n^{2/5}}$. This in turn implies the claimed lower bound $\mathbf{PP}_{\mathsf{query}}(F) = \tilde{\Omega}(n^{2/5})$. Meanwhile, $\overline{\mathrm{ED}}$ is in $\mathbf{NP}_{\mathrm{query}}$, and hence the same binary search-based $\mathbf{P}^{\mathbf{NP}}_{\mathsf{query}}$ protocol that works for OMB also works for F.

1.3.2 Communication Complexity

Babai, Frankl, and Simon [4] defined the (two-party) communication analogs of many complexity classes from the Turing Machine world. Since their seminal paper, these communication classes have been studied intensely, with the following motivation.

Relationship to Turing Machine Complexity. Just as query complexity separations are sometimes construed as evidence that the same separation applies to the classes' classical counterparts, so too are communication complexity separations. In addition, Aaronson and Wigderson [1] showed that a separation of communication complexity classes implies a formal barrier (called the *algebraization* barrier) to disproving the analogous separation in the Turing Machine world. Their result is analogous to how query complexity separations imply that the relativization barrier applies in the Turing Machine world. Thus, studying $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}$ and $\mathbf{PP}^{\mathbf{cc}}$ sheds additional light on the relationship between their Turing Machine counterparts. These communication classes are also of interest in their own right, as we now explain.

The class $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}$. $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}$ lies near the frontier of our current understanding of communication complexity classes, in that it is one of the most powerful communication models against which we know how to prove lower bounds. This communication class has received considerable attention in recent years: Impagliazzo and Williams [19] were the first to prove lower bounds against this class, and Papakonstantinou et al. [31] characterized the class in terms of limited memory communication models. Göös et al. [18] related $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}$ to various other communication classes near the frontier of understanding.

The class $\mathbf{PP}^{\mathbf{cc}}$. $\mathbf{PP}^{\mathbf{cc}}$ captures the difficulty of computing functions to small-bias, and it turns out to be characterized by an important combinatorial quantity known as *discrepancy* [22]. Motivated in part by this characterization, $\mathbf{PP}^{\mathbf{cc}}$ has received intense study (cf. [40, 39, 8, 18, 22, 29, 48] and many others).

An improved separation between $\mathbf{PP}^{\mathbf{cc}}$ and $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}$. Buhrman, Vereshchagin, and de Wolf [8] gave the first separation between $\mathbf{PP}^{\mathbf{cc}}$ and $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}$.[6] Specifically, they "lifted" Beigel's query complexity lower bound for OMB to the communication setting, showing that a certain communication problem G derived from OMB satisfies $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}(G) = O(\log^2 n)$, but $\mathbf{PP}^{\mathbf{cc}}(G) = \Omega(n^{1/3})$. Prior to our work, this was the best separation between these two communication classes.

We improve on this separation. By applying Sherstov's pattern matrix method [40] to the function F of Eq. (4), we obtain a communication problem F' that satisfies $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}(F') = O(\log^2 n)$, but $\mathbf{PP}^{\mathbf{cc}}(F') = \tilde{\Omega}(n^{2/5})$.

An improved separation between $\mathbf{PP}^{\mathbf{cc}}$ and $\mathbf{UPP}^{\mathbf{cc}}$ for an $\mathbf{AC}^{\mathbf{0}}$ function. Buhrman et al.'s function G also exhibited the first separation between $\mathbf{PP}^{\mathbf{cc}}$ and a related communication class called $\mathbf{UPP}^{\mathbf{cc}}$, which captures the difficulty of computing f to strictly positive bias (Sherstov [37] independently separated these two classes). In more detail, the function G used by Buhrman et al. satisfies $\mathbf{UPP}^{\mathbf{cc}}(G) = O(\log n)$, while $\mathbf{PP}^{\mathbf{cc}}(G) = \Omega(n^{1/3})$, and until our work this remained the best known separation between $\mathbf{PP}^{\mathbf{cc}}$ and $\mathbf{UPP}^{\mathbf{cc}}$ for any function in AC^0. Our communication problem F' improves on this separation, giving a function F' in AC^0 satisfying $\mathbf{UPP}^{\mathbf{cc}}(F') = O(\log n)$, but $\mathbf{PP}^{\mathbf{cc}}(F') = \tilde{\Omega}(n^{2/5})$.

To further motivate this application, we mention that $\mathbf{PP}^{\mathbf{cc}}$ is characterized not only by discrepancy, but also by the learning-theoretic notion of *margin complexity* [29, 28], while $\mathbf{UPP}^{\mathbf{cc}}$ is characterized by the notion of *dimension complexity* [33]. Both margin complexity

[6] Buhrman et al. framed their result as an exponential separation between the $\mathbf{PP}^{\mathbf{cc}}$ and a related class called $\mathbf{UPP}^{\mathbf{cc}}$. As pointed out in subsequent work [18], their result also separates $\mathbf{P}^{\mathbf{NP}^{\mathbf{cc}}}$ and $\mathbf{PP}^{\mathbf{cc}}$.

and dimension complexity underly state-of-the-art learning algorithms for constant-depth circuits in a variety of learning models (for details, see [24, 35, 10, 40, 23] and the references therein). Separating these two quantities sheds light on the relative power of these algorithms.

1.3.3 Roadmap for the Rest of the Paper

We introduce notation and establish preliminary lemmas in Section 2. Section 3 provides an intuitive overview of the dual witness we construct to prove Theorem 1, before providing proof details. In the full version of the paper, we collect formal definitions of approximate degree and its one-sided variants, along with their dual characterizations, and formalize our applications to query and communication complexity.

2 Notation and Preliminary Facts

Given a set $T \subseteq \{-1,1\}^N$, we let $\mathbb{I}_T$ denote the indicator vector of T; that is, $\mathbb{I}_T(x) = 1$ if $x \in T$, and $\mathbb{I}_T(x) = 0$ otherwise. Given a dual polynomial $\psi : \{-1,1\}^N \to \mathbb{R}$, we define the L_1-*weight* of T under ψ to be $W_\psi(T) = \sum_{x \in T} |\psi(x)|$. We use the standard notation $\|\psi\|_1 := W_\psi(\{-1,1\}^N)$, and refer to $\|\psi\|_1$ as the L_1-*norm* of ψ. Define the function $\widetilde{\mathsf{sgn}} : \mathbb{R} \to \{-1,0,1\}$ via: $\widetilde{\mathsf{sgn}}(t) = 1$ if $t > 0$, $\widetilde{\mathsf{sgn}}(t) = -1$ if $t < 0$, and $\widetilde{\mathsf{sgn}}(t) = 0$ if $t = 0$. We say that a dual polynomial ψ for a function f *makes an error* on input x if $0 \neq \widetilde{\mathsf{sgn}}(\psi(x)) \neq \widetilde{\mathsf{sgn}}(f(x))$.

Crucial to our proof are the following two facts that provide methods of combining multiple dual witnesses while preserving their pure high degree.

▶ **Fact 7.** *If $\psi_1, \psi_2 : \left(\{-1,1\}^N\right)^M \to \{-1,1\}$ both have pure high degree d, then so does $\psi_1 + \psi_2$.*

▶ **Fact 8.** *Suppose that $\psi_1, \ldots, \psi_M : \{-1,1\}^N \to \{-1,1\}$ are each defined over disjoint sets of variables, and there is some i such that ψ_i has pure high degree d. Then so does the function $\psi : \left(\{-1,1\}^N\right)^M \to \{-1,1\}$ defined via $\psi(x_1, \ldots, x_M) = \prod_{i=1}^M \psi_i(x_i)$.*

3 Proof of Theorem 1

This section proves Theorem 1, which we restate here for the reader's convenience. Recall from the introduction that for any Boolean function $f : \{-1,1\}^N \to \{-1,1\}$, F denotes the function $\mathrm{OMB}_M(f, \ldots, f)$ that maps $\{-1,1\}^{M \cdot N}$ to $\{-1,1\}$.

▶ **Theorem 1** (restated). *If $\widetilde{deg}_{+,2/3}(f) \geq d$, then $\widetilde{deg}_{+,\varepsilon}(F) \geq d$ for $\varepsilon = 1 - 2^{-M}$*

Proof. Let ψ_{in} denote a dual witness for the fact that $\widetilde{\deg}_{+,2/3}(f) \geq d$, normalized to ensure that its L_1-norm is 1. Recall from Section 1.2.1 that ψ_{in} satisfies three properties: (a) ψ_{in} has pure high degree at least d, (b) ψ_{in} has correlation $\varepsilon' \geq 2/3$ with f, and (c) ψ_{in} has positive one-sided error for f, i.e., $\psi_{\text{in}}(x_i) \geq 0$ for all $x_i \in f^{-1}(+1)$. Let E denote the set of all $x_i \in \{-1,1\}^N$ on which $\psi_{\text{in}}(x_i)$ is in error, i.e., $0 \neq \widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_i)) \neq \widetilde{\mathsf{sgn}}(f(x_i))$.

Proof Overview. For any vector $x = (x_1, \ldots, x_M) \in \left(\{-1,1\}^N\right)^M$, we think of x_M as the "most significant" block in x, because if $f(x_M) = -1$, then F evaluates to -1 regardless of the values of the other blocks $x_1, \ldots, x_{M-1}$. Similarly, we think of x_1 as the "least significant block" of x.

We think of our dual witness ψ_{comb} as being constructed iteratively. The first iteration creates a dual witness $\psi^{(1)}$ that "uses" the least significant block x_1 to "achieve" pure high

degree at least d. That is, $\psi^{(1)}$ will be uncorrelated with any polynomial p, unless the degree of p is at least d *even when restricted to the variables in the first block*. However, $\psi^{(1)}$ will only have correlation ε' with F, and hence it will make errors if $\varepsilon' < 1$. The second iteration creates a dual witness $\psi^{(2)}_{\text{comb}} = \psi^{(1)} + \psi^{(2)}$, where $\psi^{(2)}$ is a correction term that zeros out there errors of $\psi^{(1)}$. Moreover, $\psi^{(2)}$ will use the second block x_2 to achieve pure high degree at least d. By Fact 7, this ensures that $\psi^{(2)}_{\text{comb}}$ also has pure high degree at least d.

If $\psi^{(2)}$ zeroed out all of the errors of $\psi^{(1)}$ without introducing any new errors, then $\psi^{(2)}_{\text{comb}}$ would have perfect correlation with F, and we would be done. Unfortunately, $\psi^{(2)}$ does introduce new errors. But we have made tangible progress: we show that the number of errors $\psi^{(2)}$ makes, relative to $\psi^{(1)}$, falls by a factor of $W_{\psi_{\text{in}}}(f^{-1}(+1))/W_{\psi_{\text{in}}}(E) = \varepsilon'/(1-\varepsilon')$. Since $\varepsilon' \geq 2/3$, we conclude that $\varepsilon'/(1-\varepsilon') \geq 2$, and hence that $\psi^{(2)}$ makes at most half as many errors as $\psi^{(1)}$.

In general, the ith iteration adds in a correction term $\psi^{(i)}$ that zeros out all of the errors of the dual witness $\psi^{(i-1)}_{\text{comb}}$ constructed in the previous iteration. $\psi^{(i)}$ will use the ith input block x_i to achieve pure high degree at least d, and will introduce at most a $W_{\psi_{\text{in}}}(E)/W_{\psi_{\text{in}}}(f^{-1}(+1)) \leq 1/2$ fraction of the errors made by $\psi^{(i-1)}$. At the end of iteration M, we have constructed a dual witness $\psi_{\text{comb}} := \sum_{i=1}^{M} \psi^{(i)}$ that makes only a $\big(W_{\psi_{\text{in}}}(E)/W_{\psi_{\text{in}}}(f^{-1}(+1))\big)^M = ((1-\varepsilon')/\varepsilon')^M \leq 2^{-M}$ fraction of the errors made by $\psi^{(1)}$, and we are done.

Proof Details. Throughout, we assume without loss of generality that M is odd (we only exploit this assumption in the proof of Lemma 17, which shows that ψ_{comb} has positive one-sided error for F).

Properties of ψ_{in}. Throughout, we let $Q^-, Q^+ \subseteq \{-1,1\}^N$ denote the set of inputs x_i for which $\psi_{\text{in}}(x_i) < 0$ and $\psi_{\text{in}}(x_i) > 0$ respectively. We assume $d \geq 1$, as otherwise Theorem 1 holds trivially. We make use of the following simple facts about $\mathbb{I}_{Q^+}$ and $\mathbb{I}_{Q^-}$.

▶ **Fact 9.** $\sum_{x_i \in \{-1,1\}^N} \mathbb{I}_{Q^-}(x_i) \cdot |\psi_{in}(x_i)| = \sum_{x_i \in \{-1,1\}^N} \mathbb{I}_{Q^+}(x_i) \cdot |\psi_{in}(x_i)| = 1/2$.

Proof. Since ψ_{in} witnesses the fact that $\widetilde{\deg}_{+,1/2}(f) \geq d$, ψ_{in} has pure high degree at least $d \geq 1$. In particular, ψ_{in} is uncorrelated with any constant function. Hence, $\sum_{x_i \in \{-1,1\}^N} \psi_{\text{in}}(x_i) = 0$. Since $\sum_{x_i \in \{-1,1\}^N} |\psi_{\text{in}}(x_i)| = 1$, it follows that $\sum_{x_i \in \{-1,1\}^N : x_i \in Q^+} |\psi_{\text{in}}(x_i)| = \sum_{x_i \in \{-1,1\}^N : x_i \in Q^-} |\psi_{\text{in}}(x_i)| = 1/2$, which is equivalent to what we wished to prove. ◀

A crucial implication of the fact that ψ_{in} has positive one-sided error is that if ψ_{in} outputs a negative value on input x_i, we can "trust" that $f(x_i) = -1$. This is formalized as follows.

▶ **Fact 10.** *For all $x_i \in Q^-$, it holds that $f(x_i) = -1$. Equivalently, $E \subseteq f^{-1}(-1)$, or in other words $E \cap f^{-1}(+1) = \emptyset$.*

The following two facts relate the correlation of ψ_{in} with f to the L_1-weight of the sets E and $f^{-1}(+1)$ under ψ_{in}.

▶ **Fact 11.** $W_{\psi_{in}}(E) = (1-\varepsilon')/2$.

Proof. By Property (a), $\varepsilon' = \sum_{x_i \in \{-1,1\}^N} \psi_{\text{in}}(x_i) \cdot f(x_i) = 1 - 2\sum_{x_i \in E} |\psi_{\text{in}}(x_i)|$. ◀

▶ **Fact 12.** $W_{\psi_{in}}(f^{-1}(+1)) = \varepsilon'/2$.

Proof. This holds by the following sequence of equalities:

$$1/2 = \sum_{x_i \in Q^+} |\psi_{\text{in}}(x_i)| = \sum_{x_i \in E} |\psi_{\text{in}}(x_i)| + \sum_{x_i \in f^{-1}(+1)} |\psi_{\text{in}}(x_i)| = (1/2 - \varepsilon'/2) + \sum_{x_i \in f^{-1}(+1)} |\psi_{\text{in}}(x_i)|.$$

The first equality holds by Fact 9, the second because ψ_{in} satisfies Property (c), and the third by Fact 11. ◀

Construction of ψ_{comb}. The dual witness we construct is:

$$\psi_{\text{comb}}(x_1, \ldots, x_M) = \sum_{i=1}^{M} \psi^{(i)}, \text{ where} \tag{5}$$

$$\psi^{(i)} = (-1)^{i-1} \cdot (2/\varepsilon')^{M-1} \left(\prod_{j<i} \mathbb{I}_E(x_j) \cdot |\psi_{\text{in}}(x_j)| \right) \cdot \psi_{\text{in}}(x_i) \cdot \left(\prod_{j=i+1}^{M} \mathbb{I}_{f^{-1}(+1)}(x_j) \cdot |\psi_{\text{in}}(x_j)| \right). \tag{6}$$

Recall that, to show that ψ_{comb} is a dual witness for the property $\widetilde{\deg}_{+,\varepsilon}(F) \geq d$ for $\varepsilon = 1 - 2^{-M}$, it suffices to establish three properties of ψ_{comb} (cf. Section 1.2.1): (a) it must have pure high degree at least d, (b) it must satisfy $\sum_{x \in (\{-1,1\}^N)^M} \psi_{\text{comb}}(x) \cdot F(x) \geq \|\psi\|_1 \cdot \varepsilon$, where $\|\psi\|_1 = \sum_{(x \in \{-1,1\}^N)^M} |\psi_{\text{comb}}(x)|$, and (c) it must have positive one-sided error. We establish each in turn below, in Propositions 13, 14, and 17.

▶ **Proposition 13.** *ψ_{comb} has pure high degree at least d.*

Proof. Since ψ_{in} has pure high degree at least d, Fact 8 implies that each term $\psi^{(i)}$ in the sum within Eq. (5) also has pure high degree at least d. The lemma then follows by Fact 7. ◀

▶ **Proposition 14.** $\sum_{x \in (\{-1,1\}^N)^M} \psi_{comb}(x) \cdot F(x) \geq \|\psi\|_1 \cdot \varepsilon$.

The proof of Proposition 14 will make use of the following two lemmas.

▶ **Lemma 15.** $\|\psi\|_1 \geq 1/2$.

Proof. Consider the set $S = \{(x_1, \ldots, x_M) : x_1 \in Q^- \text{ and } x_2, \ldots, x_M \in f^{-1}(+1)\}$. We claim that the weight, $W_{\psi_{\text{comb}}}(S)$, that ψ_{comb} places on the set S is $1/2$. The lemma clearly follows.

To see this, fix $x = (x_1, \ldots, x_M) \in S$. We first note that for all $i \geq 2$, $\psi^{(i)}(x) = 0$. Indeed, $Q^- \cap E = \emptyset$ (cf. Fact 10), and hence $\mathbb{I}_E(x_1) = 0$. Thus, it is immediate from Eq. (6) that $\psi^{(i)}(x) = 0$ for $i \geq 2$.

So it suffices to show that $\sum_{x \in S} -\psi^{(1)}(x) \geq 1/2$. This follows from the following calculation:

$$\sum_{x \in S} -\psi^{(1)}(x) = (2/\varepsilon')^{M-1} \cdot \left(\sum_{x_1 \in Q^-} -\psi_{\text{in}}(x_1) \right) \cdot \left(\prod_{j=2}^{M} \left(\sum_{x_j \in \{-1,1\}^N} \mathbb{I}_{f^{-1}(+1)}(x_j) \cdot |\psi_{\text{in}}(x_j)| \right) \right)$$

$$= (2/\varepsilon')^{M-1} \cdot (1/2) \cdot \prod_{j=2}^{M} (\varepsilon'/2) = 1/2,$$

where the first equality holds by Eq. (6), and the second holds by Facts 9 and 12. ◀

▶ **Lemma 16.** *Let $E_{comb} \subseteq \left(\{-1,1\}^N\right)^M$ denote the set of inputs on which ψ_{comb} makes an error, i.e., $0 \neq \widetilde{\mathsf{sgn}}(\psi_{comb}(x)) \neq \widetilde{\mathsf{sgn}}(F(x))$. Let $E^M \subseteq \left(\{-1,1\}^N\right)^M$ denote $\{(x_1,\ldots,x_M) : x_i \in E \text{ for all } i\}$. Then $E_{comb} = E^M$.*

Proof. We first show that $E^M \subseteq E_{\text{comb}}$ before showing that $E_{\text{comb}} \subseteq E^M$. Suppose that $x = (x_1,\ldots,x_M) \in E^M$. Fact 10 states that $E \subseteq f^{-1}(-1)$, and hence $\mathbb{I}_{f^{-1}(+1)}(x_M) = 0$. It is then immediate from Eq. (6) that $\psi^{(i)}(x) = 0$ for all $i < M$. Meanwhile, by Eq. (6) it holds that

$$\widetilde{\mathsf{sgn}}(\psi^{(M)}(x)) = (-1)^{M-1} \cdot \widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_M)) = (-1)^{M-1}.$$

Here, we used the fact that $\widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_M)) > 0$ if $x_M \in E$. (To see this, note that since $x_M \in E$, it holds that $0 \neq \widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_M)) \neq f(x_M) = -1$, where the final equality holds because $E \subseteq f^{-1}(-1)$.) At the same time,

$$F(x) = \text{OMB}_M(-1,-1,\ldots,-1) = (-1)^M.$$

Thus, $x \in E_{\text{comb}}$ as claimed.

Fix any $x = (x_1,\ldots,x_M) \in \left(\{-1,1\}^N\right)^M$ such that there exists an $i \in \{1,\ldots,M\}$ satisfying $x_i \notin E$. To show that $E_{\text{comb}} \subseteq E^M$, we must show that $x \notin E_{\text{comb}}$. To this end, let i^* be the smallest coordinate such that $x_{i^*} \notin E$. It is clear that $\psi_{\text{comb}}(x) = 0$ if $\psi_{\text{in}}(x_i) = 0$ for any $i \in [M]$, and hence $x \notin E_{\text{comb}}$. So assume throughout that $\psi_{\text{in}}(x_i) \neq 0$ for all i. The proof proceeds via a case analysis.

- Case 1: There exists a $j > i^*$ such that $x_j \notin f^{-1}(+1)$. In this case, $\mathbb{I}_{f^{-1}(+1)}(x_j) = 0$, so it is immediate from Eq. (6) that $\psi^{(k)}(x) = 0$ for all $k < j$. Meanwhile, since $\mathbb{I}_E(x_{i^*}) = 0$, it is immediate from Eq. (6) that $\psi^{(k)}(x) = 0$ for all $k \geq j$. Thus, $\psi_{\text{comb}}(x) = \sum_{k=0}^{M} \psi^{(k)}(x) = 0$, implying that $x \notin E_{\text{comb}}$.
- Case 2: $i^* = 1$, and $x_j \in f^{-1}(+1)$ for all $j > i^*$. In this case, it is clear by Eq. (6) that

$$\widetilde{\mathsf{sgn}}(\psi^{(1)}(x)) = (-1)^0 \cdot \widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_1)) = \widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_1)) = \widetilde{\mathsf{sgn}}(f(x_1)) = F(x_1,\ldots,x_M). \quad (7)$$

  Here, the third equality holds because $x_1 \notin E$, and the fourth equality exploits the fact that if $x_j \in f^{-1}(+1)$ for all $j > 1$, then $F(x) = f(x_1)$.
  Meanwhile, since $x_1 \notin E$, it holds that $\mathbb{I}_E(x_1) = 0$, and so it is clear by Eq. (6) that $\psi^{(k)}(x) = 0$ for all $k \geq 2$. Combining this with Eq. (7), we conclude that $\widetilde{\mathsf{sgn}}(\psi_{\text{comb}}(x)) = \widetilde{\mathsf{sgn}}(\psi^{(1)}(x)) = F(x_1,\ldots,x_M)$. Thus, $x \notin E_{\text{comb}}$.
- Case 3: $i^* \geq 2$, and $x_j \in f^{-1}(+1)$ for all $j > i^*$. First, we argue that $\psi^{(k)} = 0$ for all $k < i^* - 1$. Indeed, for all such k, $x_{k+1} \in E \subseteq f^{-1}(-1)$ (cf. Fact 10), and so it holds that $\mathbb{I}_{f^{-1}(+1)}(x_{k+1}) = 0$. Hence, it is immediate from Eq. (6) that $\psi^{(k)}(x) = 0$.
  Next, we argue that $\psi^{(k)} = 0$ for all $k \geq i^* + 1$. Indeed, $x_{i^*} \notin E$, so $\mathbb{I}_E(x_{i^*}) = 0$. It is then immediate from Eq. (6) that $\psi^{(k)}(x) = 0$ for all $k \geq i^* + 1$.
  Finally, we claim that either $\psi^{(i^*-1)}(x) + \psi^{(i^*)}(x) = 0$ or $\widetilde{\mathsf{sgn}}(\psi^{(i^*-1)}(x) + \psi^{(i^*)}(x)) = F(x)$. This follows from the following calculation.
  - Case 3a: Suppose $x_{i^*} \notin f^{-1}(+1)$, i.e., that $\mathbb{I}_{f^{-1}(+1)}(x_{i^*}) = 0$. Then is clear from Eq. (6) that $\psi^{(i^*-1)}(x) = 0$. Meanwhile, since $x_{i^*} \notin E$, it is clear from Eq. (6) that

$$\widetilde{\mathsf{sgn}}(\psi^{(i^*)}(x)) = (-1)^{i^*-1} \cdot \widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_{i^*})) = (-1)^{i^*-1} \cdot f(x_{i^*}) = F(x),$$

    where the final equality exploits the fact that if $x_j \in f^{-1}(+1)$ for all $j > i^*$, and $x_{i^*-1} \in E \subseteq f^{-1}(-1)$ (Fact 10), then $F(x) = (-1)^{i^*-1} \cdot f(x_{i^*})$.

- Case 3b: Suppose $x_{i^*} \in f^{-1}(+1)$. We claim that it holds that $\psi^{(i^*-1)}(x) = -\psi^{(i^*)}(x)$. To see this, note that in this case

$$\psi^{(i^*-1)}(x) = (-1)^{i^*-2} \cdot (2/\varepsilon')^{M-1} \cdot \psi_{\text{in}}(x_{i^*-1}) \cdot \prod_{j \neq i^*-1} |\psi_{\text{in}}(x_j)|, \text{ and} \tag{8}$$

$$\psi^{(i^*)}(x) = (-1)^{i^*-1} \cdot (2/\varepsilon')^{M-1} \cdot \psi_{\text{in}}(x_{i^*}) \cdot \prod_{j \neq i^*} |\psi_{\text{in}}(x_j)|. \tag{9}$$

  Both of the above quantities are clearly equal in absolute value, but it remains to show that $\psi^{(i^*-1)}(x) = -\psi^{(i^*)}(x)$. Since $x_{i^*-1} \in E \subseteq f^{-1}(-1)$ (Fact 10), it holds that $\widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_{i^*-1})) = +1$. Meanwhile, since $x_{i^*} \notin E$, $\widetilde{\mathsf{sgn}}(\psi_{\text{in}}(x_{i^*})) = f(x_{i^*}) = +1$. Hence, $\widetilde{\mathsf{sgn}}(\psi^{(i^*-1)}(x)) = (-1)^{i^*-2}$, while $\widetilde{\mathsf{sgn}}(\psi^{(i^*)}(x)) = (-1)^{i^*-1}$, completing the proof.

Combining all of the above, we conclude that $\psi_{\text{comb}}(x) = \sum_{j=1}^{M} \psi_{\text{comb}}^{(j)}(x) = \psi_{\text{comb}}^{(i^*-1)}(x) + \psi_{\text{comb}}^{(i^*)}(x)$, and the latter expression is either equal to 0 or agrees in sign with $F(x)$. Thus, $x \notin E_{\text{comb}}$. This completes the proof of Lemma 16. ◀

Proof of Proposition 14. Note that

$$\begin{aligned}\sum_{x \in (\{-1,1\}^N)^M} \psi_{\text{comb}}(x) \cdot F(x) &= \sum_{x \in (\{-1,1\}^N)^M} |\psi_{\text{comb}}(x)| - 2 \sum_{x \in E_{\text{comb}}} |\psi_{\text{comb}}(x)| \\ &= \|\psi\|_1 - 2 \sum_{x \in E_{\text{comb}}} |\psi_{\text{comb}}(x)|, \end{aligned} \tag{10}$$

where we recall from Lemma 16 that $E_{\text{comb}} = E^M$ is the set of points on which ψ_{comb} makes an error. Observe that for each j:

$$\sum_{x \in E^M} \psi^{(j)}(x) \leq (2/\varepsilon')^{M-1} \prod_{i=1}^{M} \left(\sum_{x_i \in E} |\psi_{\text{in}}(x_i)| \right) \leq (2/\varepsilon')^{M-1} \cdot \prod_{i=1}^{M} ((1-\varepsilon')/2) \leq 3^{M-1}/6^M < 2^{-M-1} \tag{11}$$

Here, the first equality holds because, for all $x \in E^M$ and $j < M$, $\psi^{(j)}(x) = 0$; this follows by combining Eq. (6) with the fact that $E \cap f^{-1}(+1) = \emptyset$ (Fact 10) (see also the $E^M \subseteq E_{\text{comb}}$ direction in the proof of Lemma 16). The second inequality holds by Fact 11, and the third because $\varepsilon' \geq 2/3$. Combining Lemma 15 with Eq. (10) and Eq. (11), we conclude that $\sum_{x \in (\{-1,1\}^N)^M} \psi_{\text{comb}}(x) \cdot F(x) \geq \|\psi\|_1 - 2^{-M-1} \geq \|\psi\|_1 (1 - 2^{-M})$, completing the proof. ◀

▶ **Proposition 17.** *$\psi_{comb}(x) \geq 0$ for all $x \in F^{-1}(+1)$.*

Proof. Lemma 16 implies that the set E_{comb} on which ψ_{comb} makes an error is equal to E^M. Since $E \subseteq f^{-1}(-1)$ (cf. Fact 10), and we assumed that M is odd, it is obvious from the definition of F that $E^M \subseteq F^{-1}(-1)$. It follows that ψ_{comb} makes no errors on $F^{-1}(+1)$, implying the proposition. ◀

Theorem 1 follows from Propositions 13, 14, 17 and the dual characterization of $\widetilde{\deg}_{+,\varepsilon}$. ◀

Acknowledgements. The author is grateful to Mark Bun, Mika Göös, and the anonymous reviewers for insightful comments on earlier versions of this manuscript.

References

1. S. Aaronson and A. Wigderson. Algebrization: A new barrier in complexity theory. *TOCT*, 1(1), 2009.
2. Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004.
3. Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory of Computing*, 1(1):37–46, 2005.
4. László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *FOCS*, pages 337–347, 1986.
5. Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4(4):431–442, 1975.
6. Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001.
7. Richard Beigel. Perceptrons, PP, and the Polynomial Hierarchy. *Computational Complexity*, 4:339–349, 1994.
8. Harry Buhrman, Nikolai K. Vereshchagin, and Ronald de Wolf. On computation and communication with small bias. In *CCC*, pages 24–32, 2007.
9. Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and markov-bernstein inequalities. In *ICALP (1)*, pages 303–314, 2013.
10. Mark Bun and Justin Thaler. Hardness amplification and the approximate degree of constant-depth circuits. In *ICALP, Part I*, pages 268–280, 2015.
11. Karthekeyan Chandrasekaran, Justin Thaler, Jonathan Ullman, and Andrew Wan. Faster private release of marginals on small databases. In *ITCS*, pages 387–402, 2014.
12. Arkadev Chattopadhyay and Anil Ada. Multiparty communication complexity of disjointness. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(002), 2008.
13. Matei David and Toniann Pitassi. Separating NOF communication complexity classes RP and NP. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(014), 2008.
14. Matei David, Toniann Pitassi, and Emanuele Viola. Improved separations between nondeterministic and randomized multiparty communication. *TOCT*, 1(2), 2009.
15. R. de Wolf. A note on quantum algorithms and the minimal degree of ϵ-error polynomials for symmetric functions. *Quantum Information & Computation*, 8(10):943–950, 2010.
16. Dmitry Gavinsky and A. A. Sherstov. A separation of NP and conp in multiparty communication complexity. *Theory of Computing*, 6(1):227–245, 2010.
17. Dmitry Gavinsky and A. A. Sherstov. A separation of NP and coNP in multiparty communication complexity. *Theory of Computing*, 6(1):227–245, 2010.
18. Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:49, 2015. To appear in ICALP, 2016.
19. Russell Impagliazzo and Ryan Williams. Communication complexity with synchronized clocks. In *CCC*, pages 259–269, 2010.
20. Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008.
21. Varun Kanade and Justin Thaler. Distribution-independent reliable learning. In *COLT*, pages 3–24, 2014.
22. Hartmut Klauck. Lower bounds for quantum communication complexity. *SIAM J. Comput.*, 37(1):20–46, 2007.
23. Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *J. Comput. Syst. Sci.*, 68(2):303–318, 2004.
24. Adam R. Klivans and Rocco A. Servedio. Toward attribute efficient learning of decision lists and parities. *Journal of Machine Learning Research*, 7:587–602, 2006.

25 Samuel Kutin. Quantum lower bound for the collision problem with small range. *Theory of Computing*, 1(1):29–36, 2005.

26 Troy Lee. A note on the sign degree of formulas. *CoRR*, abs/0909.4607, 2009.

27 Troy Lee and Adi Shraibman. Disjointness is hard in the multiparty number-on-the-forehead model. *Computational Complexity*, 18(2):309–336, 2009.

28 Nati Linial and Adi Shraibman. Learning complexity vs. communication complexity. In *CCC*, pages 53–63, 2008.

29 Nati Linial and Adi Shraibman. Lower bounds in communication complexity based on factorization norms. *Random Struct. Algorithms*, 34(3):368–394, 2009.

30 Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

31 Periklis A. Papakonstantinou, Dominik Scheder, and Hao Song. Overlays and limited memory communication. In *CCC*, pages 298–308, 2014.

32 Ramamohan Paturi. On the degree of polynomials that approximate symmetric boolean functions (preliminary version). In *STOC*, pages 468–474, 1992.

33 Ramamohan Paturi and Janos Simon. Probabilistic communication complexity. *J. Comput. Syst. Sci.*, 33(1):106–123, 1986.

34 Anup Rao and Amir Yehudayoff. Simplified lower bounds on the multiparty communication complexity of disjointness. In *CCC*, pages 88–101, 2015.

35 A. A. Razborov and A. A. Sherstov. The sign-rank of ac^0. *SIAM J. Comput.*, 39(5):1833–1855, 2010.

36 Rocco A. Servedio, Li-Yang Tan, and Justin Thaler. Attribute-efficient learning and weight-degree tradeoffs for polynomial threshold functions. In *COLT*, pages 14.1–14.19, 2012.

37 A. A. Sherstov. Halfspace matrices. *Computational Complexity*, 17(2):149–178, 2008.

38 A. A. Sherstov. Approximate inclusion-exclusion for arbitrary symmetric functions. *Computational Complexity*, 18(2):219–247, 2009.

39 A. A. Sherstov. Separating AC^0 from depth-2 majority circuits. *SIAM J. Comput.*, 38(6):2113–2129, 2009.

40 A. A. Sherstov. The pattern matrix method. *SIAM J. Comput.*, 40(6):1969–2000, 2011.

41 A. A. Sherstov. Strong direct product theorems for quantum communication and query complexity. In *STOC*, pages 41–50, 2011.

42 A. A. Sherstov. The multiparty communication complexity of set disjointness. In *STOC*, pages 525–548, 2012.

43 A. A. Sherstov. Approximating the AND-OR tree. *Theory of Computing*, 9(20):653–663, 2013.

44 A. A. Sherstov. Communication lower bounds using directional derivatives. In *STOC*, pages 921–930, 2013.

45 A. A. Sherstov. The intersection of two halfspaces has high threshold degree. *SIAM J. Comput.*, 42(6):2329–2374, 2013.

46 A. A. Sherstov. Breaking the Minsky-Papert barrier for constant-depth circuits. In *STOC*, pages 223–232, 2014.

47 A. A. Sherstov. The power of asymmetry in constant-depth circuits. In *FOCS*, pages 431–450, 2015.

48 Yaoyun Shi and Yufan Zhu. Quantum communication complexity of block-composed functions. *Quantum Information & Computation*, 9(5):444–460, 2009.

49 Justin Thaler, Jonathan Ullman, and Salil P. Vadhan. Faster algorithms for privately releasing marginals. In *ICALP, Part I*, pages 810–821, 2012.

50 S. Toda. On the computational power of PP and +P. In *FOCS*, pages 514–519, 1989.

Dynamic Graph Stream Algorithms in $o(n)$ Space

Zengfeng Huang[1] and Pan Peng*[2]

1 **University of New South Wales, Sydney, Australia**
`zengfeng.huang@unsw.edu.au`
2 **Department of Computer Science, TU Dortmund, Dortmund, Germany; and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China**
`pan.peng@tu-dortmund.de`

Abstract

In this paper we study graph problems in dynamic streaming model, where the input is defined by a sequence of edge insertions and deletions. As many natural problems require $\Omega(n)$ space, where n is the number of vertices, existing works mainly focused on designing $\tilde{O}(n)$ space algorithms. Although sublinear in the number of edges for dense graphs, it could still be too large for many applications (e.g. n is huge or the graph is sparse). In this work, we give single-pass algorithms beating this space barrier for two classes of problems. We present $o(n)$ space algorithms for estimating the number of connected components with additive error εn and $(1+\varepsilon)$-approximating the weight of minimum spanning tree. The latter improves previous $\tilde{O}(n)$ space algorithm given by Ahn et al. (SODA 2012) for connected graphs with bounded edge weights. We initiate the study of approximate graph property testing in the dynamic streaming model, where we want to distinguish graphs satisfying the property from graphs that are ε-far from having the property. We consider the problem of testing k-edge connectivity, k-vertex connectivity, cycle-freeness and bipartiteness (of planar graphs), for which, we provide algorithms using roughly $\tilde{O}(n^{1-\varepsilon})$ space, which is $o(n)$ for any constant ε. To complement our algorithms, we present $\Omega(n^{1-O(\varepsilon)})$ space lower bounds for these problems, which show that such a dependence on ε is necessary.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases dynamic graph streams, sketching, property testing, minimum spanning tree

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.18

1 Introduction

Graphs or networks are a natural way to describe structural information. For example, users of Facebook and the acquaintance relations among them form a social network, the proteins together with interactions between them define a biological network, and web-pages and hyperlinks give rise to a huge web graph. Due to the rapid development of information technology, many such graphs become extremely large, and are constantly changing, which poses great challenges for analyzing their structures. Over the last decade, the data stream model [34] has proven to be successful in dealing with *big data*. In this model, the algorithm should make only one pass (or a few passes) over the stream, and use sublinear working space. The time required to output the final answer and process each element is also important. There is a growing body of work studying graph problems over data streams. Graph streams

* Supported by ERC grant No. 307696.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 18; pp. 18:1–18:16

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

were first considered by Henzinger et al. [24], and later have been extensively studied in the *insertion-only* model (eg., [17, 18, 34]), where there is no edge deletion in the stream. Recently, starting from the seminal works of Ahn, Guha and McGregor [2, 3], the interest has shifted to the *dynamic streaming model*, where the edges can be both inserted and deleted (see eg., [28, 29, 1, 5, 9, 10, 7, 31, 6, 33, 15, 23]). In this setting, most algorithms designed are linear sketch-based, which is also an effective technique for processing distributed graphs. For more information about graph streaming algorithms see the recent survey by McGregor [32].

For graph streams, both insertion-only and dynamic, the research in the past has mostly focused on the *semi-streaming model*, in which the algorithms are allowed to use $\tilde{O}(n)$ space, where n is the number vertices in the graph. (For notational convenience, we will use $\tilde{O}(g)$ and $\tilde{\Omega}(g)$ to hide $\operatorname{poly}\log(g)$ factors.) The reason behind this is that even in the insertion-only model, many natural graph problems require $\Omega(n)$ space (e.g. testing if the graph is connected [18]). Note that the allowed space in semi-streaming model is sublinear in the input size as the number of edges of the graph might be as large as $\Omega(n^2)$. However, in many real applications n is huge and the input graph is already very sparse, an $\tilde{O}(n)$ algorithm might be even worse than just storing all the edges. From this perspective, one may naturally ask the question *which kind of problems can be solved with even less space, i.e., $o(n)$ space.*

To the best of our knowledge, very few results are known in this direction. Chitnis et al. [10] and Fafianie and Kratsch [16] introduced parameterized graph stream algorithms which may only use $o(n)$ space with some promise of the size of the solution. This parameterized setting has been further investigated in [9]. In addition, it has been shown that the size of the maximum matching can be approximated within constant factor in $\tilde{O}(n^{4/5})$ space for graphs with bounded arboricity [14, 9, 7].

In this paper, we study two classes of graph problems that admit single-pass $o(n)$ space algorithms in the dynamic streaming model. The first class contains the problems of estimating the number of connected components and the weight of minimum spanning tree (MST). We show that one can estimate the number of connected components within an *additive* error of εn with $o(n)$ space and post-processing time, for any small constant $\varepsilon > 0$. We also present an algorithm to $(1+\varepsilon)$-approximate the weight of MST with $o(n)$ space and post-processing time for connected graphs with bounded edge weights, which improves the best known algorithm with $\tilde{O}(n)$ space in the same setting given by Ahn et al. [2]. It is worthy noting that the problem of estimating the number of connected components within small *multiplicative* error requires $\Omega(n)$ space, as it is generally harder than the problem of (exactly) testing graph connectivity; and that estimating the weight of MST for graphs with arbitrarily large edge weights (e.g., $\Omega(\log n)$) requires $\Omega(n)$ space (see Theorem 12). Previously these two problems have been studied in the framework of *sublinear time algorithms* (see eg. [8, 39]).

The second class consists of problems that are relaxations of deciding graph properties. Given a huge graph, it is very useful to know whether the graph has some predetermined property, such as k-connectivity, bipartiteness, cycle-freeness and etc., which provide valuable information about the graph. However, besides the requirement of $\Omega(n)$ space, exactly testing of these properties sometimes is a too strong requirement for analyzing highly dynamic graphs, since the answer may change in the next second due to an insertion or deletion of a single edge. In this paper, we initiate the study of *approximate graph property testing* in the dynamic streaming model: we want to test whether a graph satisfies some property or one has to modify a small constant fraction of edges to make it have the property. This notion of approximation is adapted from the framework of *property testing* [21, 22, 36], and a large number of existing literatures have given efficient testing algorithms (called *testers*) for many properties under different query models (see surveys [20, 38]). We show that some

Table 1 Upper and lower bounds of streaming testers.

	Space $\tilde{O}$	Space lower bound Ω
Connectivity	$n^{1-\varepsilon}$	$n^{1-8\varepsilon}$
k-edge connectivity	$k^{1+\varepsilon} \cdot n^{1-\varepsilon}$	
k-vertex connectivity	$\frac{k^{1+\varepsilon/4}}{\varepsilon} \cdot n^{1-\varepsilon/4}$	
Cycle-freeness	$n^{1-\varepsilon+\varepsilon^2}$	$n^{1-8\varepsilon}$
Bipartiteness of planar graphs	$n^{1-\Omega(\varepsilon^2)}$	$n^{1-4\varepsilon}$

fundamental properties can be tested in both $o(n)$ space and post-processing time in the dynamic streaming model and we also present close lower bounds for these problems which hold even in the insertion-only model. We remark that McGregor [32] also suggested to study the (approximate) property testers in graph streaming model, and asked whether more space-efficient algorithms exist for these problems, and we thus give affirmative answer to this question.

1.1 Our results

Now we formally state our main results. Our results regarding estimating the number of connected components and the MST weight are as follows.

- **Estimating the number of connected components.** We present a dynamic streaming algorithm that estimates the number of connected components within additive error εn in $\tilde{O}(n^{1-\varepsilon+\varepsilon^{q+1}})$ space and post-processing time for any constant $q \geq 1$. We note that a lower bound of $\Omega(n^{1-O(\varepsilon)})$ for this problem follows from the work [41].
- **Estimating the weight of minimum spanning tree (MST).** In this problem, we want to estimate the weight of the MST of a graph with edge weights in the set $\{1, 2, \cdots, W\}$. We give a dynamic streaming algorithm that computes a $(1+\varepsilon)$-approximation of the MST weight and uses space and post-processing time $\tilde{O}(Wn^{1-\frac{\varepsilon}{W-1}+\frac{\varepsilon^t}{(W-1)^t}})$ for any constant $t \geq 1$. By an argument in [8], the result can be extended to non-integral weights, as long as the ratio between the largest and the smallest weight is bounded. A space lower bound of $\Omega(n^{1-\frac{4\varepsilon}{W-1}})$ is shown for this problem.

We also present approximate testing algorithms for a number of fundamental graph properties. Before stating the performance of these algorithms, we first introduce some definitions. Given a graph property Π, an m-edge graph G is called *ε-far from having* Π if one has to modify more than εm edges of G to get a graph G' satisfying Π. This distance definition is adapted from [36] and is most suitable for general graphs where neither edge density nor maximum degree is restricted. We call an algorithm a *(dynamic) streaming tester* for Π, if it makes a single-pass over a stream of edge insertions and deletions, with probability at least $2/3$, accepts any graph satisfying Π, and rejects any graph that is ε-far from having Π.

We give sketch-based streaming testers for properties of being connected, k-edge connected, k-vertex connected, cycle-freeness and bipartite (for planar graphs). The performance of our testers are summarized in Table 1. We stress that most of our testers have (asymptotically) the same post-processing time as the space they used except for testing k-edge connectivity when $k \geq \Omega(n^{\varepsilon/(1+\varepsilon)})$ and k-vertex connectivity when $k \geq \Omega(n^{\varepsilon/(4+\varepsilon)})$.

1.2 Our techniques

To estimate the number of connected components with small additive error εn, we note that it is sufficient to estimate the number $\mathrm{scc}(G)$ of connected components of small size (i.e., $O(1/\varepsilon)$), since the number of components of size larger than this is at most $O(\varepsilon n)$ (see also [8]). To estimate $\mathrm{scc}(G)$, the following vertex sampling framework is used: we sample a sufficient large set of vertices S by sampling each vertex in G with some probability p, and then use the statistics of the sampled connected components of the original graph to estimate $\mathrm{scc}(G)$. For any small connected component C in G, it is likely that all the vertices in C will be sampled out. Conditioned on this, we add $1/p^{|C|}$ to our final estimator, which is the reciprocal of the probability that C is entirely sampled out. Now the task is then to identify which subsets of S are connected components in the original graph. A trivial way is to check all subsets of S, which takes too much time. A more efficient way is to only check all the connected components in $G[S]$, since a sampled component of G must also form a component in $G[S]$. We carefully use a set of linear sketches to do this. More specifically, we first recover all connected components in $G[S]$ by invoking a sketch-based streaming algorithm given in [2], which only needs space near-linear in $|S|$. Then we use (different) linear sketches to check if any of these components is indeed a connected component of the original graph. We remark that the first set of linear sketches of a vertex v sketch its neighborhood information in $G[S]$, while the second set sketch its neighborhood information in G. Our $o(n)$ space streaming algorithm for $(1+\varepsilon)$-approximating the weight of MST follows via a connection between the number of connected components and the weight of MST established in [8].

To give testers for some graph property Π in dynamic streaming model, we start from the observation that if a graph G is far from having Π, then typically, there exist many small disjoint subgraphs, each of which is a witness that the graph G does not satisfy Π. (For example, if Π is connectivity, then there exists at least $\Omega(\varepsilon m)$ connected components of size at most $O(1/\varepsilon)$ in a graph that is ε-far from being connected.) This implies that by sampling a sufficient large set of vertices, with high probability, one of such subgraphs will be entirely sampled. Checking which vertices form a witness of the original graph can then be done by using the aforementioned framework. Different sketches will be used for testing different properties.

To prove lower bounds for our studied problems, we give reductions from *Boolen Hidden Hypermatching (BHH)* problem that was studied in [41]. Our reductions share similarity with the reduction in [41] to the cycle-counting problem and the reductions in [27, 30] to the approximate max-cut problem.

1.3 Related work

Ahn et al. [2] initiated the study of graph sketches, and gave dynamic semi-streaming algorithms for computing a spanning forest (which can be used to count the exact number of connected components), and $(1+\varepsilon)$-approximate the weight of MST. They also proposed algorithms to *exactly* testing of a set of properties, including testing connectivity, k-edge connectivity, and bipartiteness. Recently, Guha et al. gave dynamic streaming algorithms for exactly testing of k-vertex connectivity [23]. All these algorithms use $\tilde{O}(n)$ space ($\tilde{O}(kn)$ for k-connectivity). On the other hand, the randomized space lower bounds for these exact testing problems were known to be $\Omega(n)$ in the insertion-only model [17, 18]. Recently, Sun and Woodruff improved these lower bounds to $\Omega(n \log n)$ [40]. Verbin and Yu [41] proved a lower bound for cycle-counting, which implied a lower bound of $\Omega(n^{1-O(\varepsilon)})$ for estimating the number of components.

In the *random order* insertion-only model Kapralov et al. [26] gave a one pass streaming algorithm that estimates the maximum matching size with polylogarithmic approximation ratio in polylogarithmic space. Although sublinear in n, the model considered is very different from ours.

Sublinear time algorithms for estimating the number of connected component and the weight of MST were first given by Chazelle et al. [8]. Later these two problems have been further considered in geometric settings [11, 13, 19]. In particular, Frahling et al. studied the problem of $(1+\varepsilon)$-approximating the weight of MST in dynamic geometric data stream [19].

There has been a rich line of work on graph property testing in the query model (see surveys [38, 20]) and the goal there is to design fast algorithms that make as few queries as possible. The query models that are mostly related to ours are bounded degree model and general graph model. In particular, our definition of ε-far is adapted from the general graph model. Goldreich and Ron [22] initiated the study of property testers in bounded degree graph model, and gave testers for connectivity, k-edge connectivity, $2,3$-vertex connectivity, cycle-freeness, Eulerianity. Testing k-vertex connectivity in bounded degree graphs for arbitrary constant k was given in [42]. These testers have later been generalized to general graph model [36, 35]. Testing bipartiteness in planar graphs was studied in [12].

After having submitted the paper, we became aware that Hossein Jowhari [25] has independently studied the problem of estimating the number of connected components and provided similar results as ours, while he did not consider the streaming property testers considered here.

2 Preliminaries

2.1 Notations

We use $[n] = \{1, \cdots, n\}$ to denote the vertex set of the graph G defined by the stream, and let m denote the number of edges of G. For an undirected graph $G = ([n], E)$ and a vertex $i \in [n]$, we let $\Gamma(i)$ denote all the neighbors of i. For a set $C \subseteq [n]$, let $\Gamma(C)$ denote the set of vertices in $V \setminus C$ that have at least one neighbor in C, that is, $\Gamma(C) = \cup_{i \in C} \Gamma(i) \setminus C$. Let $E(C, V \setminus C)$ denote the set of edges crossing C and $V \setminus C$. We will use $G[C]$ to denote the subgraph induced by C.

For each vertex i, we define two vectors $\mathbf{\Delta}^i \in \{-1, 0, 1\}^{\binom{n}{2}}$ and $\mathbf{\Lambda}^i \in \{0, 1\}^n$ to encode the neighborhood information of i as follows:

$$\mathbf{\Delta}^i_{j,k} = \begin{cases} 1 & \text{if } i = j < k \text{ and } (j,k) \in E \\ -1 & \text{if } j < k = i \text{ and } (j,k) \in E \\ 0 & \text{otherwise} \end{cases} \qquad \mathbf{\Lambda}^i_j = \begin{cases} 1 & \text{if } j \in \Gamma(i) \text{ or } j = i \\ 0 & \text{otherwise} \end{cases}$$

By simple induction arguments, it is easy to prove that for any vertex set $C \subset V$, the nonzero entries in the vector $\mathbf{\Delta}^C := \sum_{i \in C} \mathbf{\Delta}^i$ corresponds to the edges between C and its complement $V \setminus C$. The nonzero entries in $\sum_{i \in C} \mathbf{\Lambda}^i$ corresponds exactly to vertices in $C \cup \Gamma(C)$.

2.2 Linear sketches

Linear sketch (or sketch for short) is a powerful tool widely used in the streaming model and other areas. Given a large vector $\mathbf{x} \in \mathbb{R}^n$, we want to construct a small sketch $\mathcal{L}(\mathbf{x})$, from which certain properties of $\mathbf{x}$ can be recovered. We call $\mathcal{L}$ a linear sketch if $\mathcal{L}(\mathbf{x}+\mathbf{y}) = \mathcal{L}(\mathbf{x}) + \mathcal{L}(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y}$, and this additive property make it trivial to implement linear sketches in the

dynamic streaming model. As in the previous works, we will use linear sketches as our main tool.

AGM sketch. We will use a dynamic streaming algorithm for constructing a spanning forest of a graph by Ahn, Guha and McGregor [2], which is summarized as follows.

▶ **Theorem 1** (AGM sketch [2]). *There exists a single-pass sketch-based dynamic streaming algorithm that uses $O(n \log^3 n)$ space, and recovers a spanning forest of the graph with probability 0.99. The recovery time of the algorithm is $\tilde{O}(n)$, and the update time is $\operatorname{poly}\log n$.*

AMS sketch. To check whether the input vector $\mathbf{x}$ is $\mathbf{0}$ or not, one can simply maintain a constant approximation of its *second frequency moment*, that is $F_2(\mathbf{x}) := \sum_i x_i^2$, which can be done in $O(\log n)$ space by using the classical AMS sketch that was introduced by Alon, Matias and Szegedy [4].

Exact k-sparse recovery. We call a vector k-sparse if $|\mathrm{supp}(\mathbf{x})| \leq k$. Given a non-zero vector $\mathbf{x} \in \mathbb{R}^n$, the goal here is to recover $\mathbf{x}$ if $\mathbf{x}$ is k-sparse, otherwise outputs **Fail**. We have the following result from [37].

▶ **Lemma 2** ([37]). *There exists an $O(k \log n \log_k \delta^{-1})$ space sketch-based algorithm that takes as input a non-zero vector $\boldsymbol{x} \in \mathbb{R}^n$, and with probability $1-\delta$, recovers $\boldsymbol{x}$ if $\boldsymbol{x}$ is k-sparse, otherwise outputs **Fail**. The update time is $O(\operatorname{poly}\log n)$ and the recovery time is $O(k \cdot \operatorname{poly}\log n)$.*

3 Estimating the number of connected components and MST weight

In this section, we present and analyze our algorithms for estimating the number of the connected components in a graph and $(1+\varepsilon)$-approximating the weight of the MST.

3.1 Estimating the number of connected components

Our first observation is that, to estimate the number of connected components within additive error εn, we can simply ignore all the large components (see also [8]). In particular, the number of components of size larger than $\Omega(1/\varepsilon)$ is at most $O(\varepsilon n)$. Thus it will be sufficient to estimate the number of components of small size, for which we have the following theorem.

▶ **Theorem 3.** *For any constant $t \geq 1$, there exists a one-pass dynamic streaming algorithm that uses $O(e^t n^{1-\varepsilon} \cdot \operatorname{poly}\log n)$ space and post-processing time to estimates the number of connected components of size at most $1/\varepsilon$ within an additive error $\varepsilon^t n$. The update time is $O(\operatorname{poly}\log n)$.*

By invoking Theorem 3 with parameter $\varepsilon' = (1-\varepsilon^q)\varepsilon$ and $t = (q+1)$, we get an estimator for the number of connected components of size smaller than $1/\varepsilon'$ within additive error at most $\varepsilon^{q+1} n$. Since the number of components of size at least $1/\varepsilon'$ is at most $\varepsilon' n = \varepsilon n - \varepsilon^{1+q} n$, the estimator also approximates the total number of connected components within additive error at most εn. The space of the algorithm is $\tilde{O}(e^{q+1} n^{1-\varepsilon+\varepsilon^{q+1}})$, and we have the following result.

▶ **Theorem 4.** *Let $q \geq 1$ be a constant. There exists a one-pass dynamic streaming algorithm that with constant success probability, estimates the number of connected components of a graph within an additive error εn in $O(e^{q+1} n^{1-\varepsilon+\varepsilon^{q+1}} \cdot \operatorname{poly}\log n)$ space and post-processing time.*

Algorithm 1 `EstimateNumSCC`

1: Sample each vertex with probability $p := (\varepsilon^{2t}n/16)^{-\varepsilon}$. If more than $16np$ vertices are sampled, then abort and output **Fail**. Let S denote the set of sampled vertices.
2: Maintain an AGM sketch of $G[S]$ using Theorem 1.
3: For each $v \in S$, maintain an AMS sketch $AMS(\mathbf{\Delta}^v)$, sketching the neighborhood of v in G.
4: **Post-Processing:**
5: Use the AGM sketch to recover a spanning forest F of the induced graph $G[S]$ using Theorem 1.
6: For each component $C \in F$, estimate $F_2(\mathbf{\Delta}^C)$ using the AMS sketch $AMS(\mathbf{\Delta}^C) = \sum_{v\in C} AMS(\mathbf{\Delta}^v)$, and set $X_C = 1$ if $F_2 = 0$, otherwise set $X_C = 0$. For each $1 \le \ell \le \frac{1}{\varepsilon}$, let $X_\ell := \sum_{C:|C|=\ell} X_C$.
7: Output $Y := \sum_{\ell \le \frac{1}{\varepsilon}} \frac{X_\ell}{p^\ell}$.

Now we give the proof of Theorem 3. Recall that the vectors $\mathbf{\Delta}^C$ encode the information of the number of edges between C and $V \setminus C$.

Proof of Theorem 3. Let $\mathrm{scc}(G)$ denote the number of connected components of size at most $1/\varepsilon$ in G. Our algorithm for estimating $\mathrm{scc}(G)$ is as follows. We first sample each vertex with probability $p := (\varepsilon^{2t}n/16)^{-\varepsilon}$. Let S be the set of sampled vertices. We then use the AGM sketch from Theorem 1 to maintain a spanning forest F of the subgraph induced by S. Then for each component C in F, we test whether C is actually a connected component in G by testing whether the vector $\mathbf{\Delta}^C := \sum_{v\in C} \mathbf{\Delta}^v$ is $\mathbf{0}$, which can be done by the AMS sketch. If $\mathbf{\Delta}^C = \mathbf{0}$, we set $X_C = 1$, otherwise set $X_C = 0$. Our estimator is then defined as $\sum_C \frac{X_C}{p^{|C|}}$, where C ranges over all components of F with size at most $\frac{1}{\varepsilon}$. See Algorithm 1 for the details.

Note that the algorithm samples at most $16np = O(\varepsilon^{-2t\varepsilon} \cdot n^{1-\varepsilon})$ vertices and we maintained an AGM sketch on $G[S]$ and an AMS sketch for each sampled vertex, which imply that the space complexity of the algorithm is $O(\varepsilon^{-2t\varepsilon} n^{1-\varepsilon} \cdot \mathrm{poly}\log n)$. By simple calculus, for any ε, it holds that $\varepsilon^{-2\varepsilon} \le e^{2/e} < e$, so the space is at most $\tilde{O}(e^t n^{1-\varepsilon})$. The post-processing time is near linear in the space, and the update time is $O(\mathrm{poly}\log n)$.

Now we prove the correctness of the above algorithm. First we note that the expected number of sampled vertices in Step (1) is np, and thus by Markov inequality, the probability that more than $16np$ vertices are sampled is at most $\frac{1}{16}$. Also note that with probability at least $1 - \frac{1}{16}$, the AGM sketch returns a true spanning forest of $G[S]$. In addition, since the number of components in F is at most n, we will query the AMS sketch at most n times. Thus if we set the error probability of the AMS sketch to be $\frac{1}{16n}$ (with an extra $\log n$ factor in space), then with probability at least $1 - \frac{1}{16}$, all invocations of AMS sketches for testing if $\mathbf{\Delta}^C = \mathbf{0}$ will give the correct answer. Conditioned on this event, X_ℓ defined in Step (6) is exactly the number of connected components B of size ℓ in G such that all vertices in B are sampled out, which is true since for any component $C \in F$, $F_2(\mathbf{\Delta}^C) = \mathbf{0}$ if and only if C is a connected component in G.

Let $B_1, \cdots, B_{\mathrm{scc}(G)}$ be the connected components of size at most $\frac{1}{\varepsilon}$ of G. For any integer $\ell \le \frac{1}{\varepsilon}$, let $\mathcal{B}_\ell$ denote the set of connected components of size ℓ in G, that is, $\mathcal{B}_\ell = \{B_i : 1 \le i \le \mathrm{scc}(G), |B_i| = \ell\}$. Let $b_\ell := |\mathcal{B}_\ell|$. Note that $\mathrm{scc}(G) = \sum_{\ell \le \frac{1}{\varepsilon}} b_\ell$. For any set B, let Z_B denote the indicator random variable that all the vertices in B have been sampled. Note that $\Pr[Z_B = 1] = p^{|B|}$. Now by the above argument, $X_\ell = \sum_{B\in\mathcal{B}_\ell} Z_B$,

and $\mathrm{E}[X_\ell] = b_\ell \cdot p^\ell$. Furthermore, we have $Y = \sum_{\ell \le \frac{1}{\varepsilon}} \frac{X_\ell}{p^\ell} = \sum_{\ell \le \frac{1}{\varepsilon}} \frac{\sum_{B \in \mathcal{B}_\ell} Z_B}{p^\ell}$, and thus $\mathrm{E}[Y] = \sum_{\ell \le \frac{1}{\varepsilon}} b_\ell = \mathrm{scc}(G)$.

Note that all Z_{B_i}'s are mutually independent for all i, so it holds that

$$\begin{aligned}\mathrm{Var}[Y] &= \sum_{\ell \le \frac{1}{\varepsilon}} \frac{\sum_{B \in \mathcal{B}_\ell} \mathrm{Var}[Z_B]}{p^{2\ell}} = \sum_{\ell \le \frac{1}{\varepsilon}} \frac{b_\ell (p^\ell - p^{2\ell})}{p^{2\ell}} \le \sum_{\ell \le \frac{1}{\varepsilon}} \frac{b_\ell}{p^\ell} \\ &\le \frac{\sum_{\ell \le \frac{1}{\varepsilon}} b_\ell}{p^{1/\varepsilon}} = \frac{\mathrm{scc}(G)}{p^{1/\varepsilon}} \le \frac{n}{p^{1/\varepsilon}} = \varepsilon^{2t} n^2/16, \qquad (1)\end{aligned}$$

where we use the fact that $\mathrm{scc}(G) \le n$, and $p = (\varepsilon^{2t} n/16)^{-\varepsilon}$. Then by Chebyshev's inequality,

$$\Pr[|Y - \mathrm{scc}(G)| \ge \varepsilon^t n] = \Pr[|Y - \mathrm{E}[Y]| \ge \varepsilon^t n] \le \frac{\mathrm{Var}[Y]}{\varepsilon^{2t} n^2} \le 1/16.$$

By the union bound, the algorithm will succeed with probability at least $\frac{2}{3}$. ◀

3.2 Approximating the weight of minimum spanning tree

We use the previous algorithm on estimating the number of connected components to approximate the weight of a minimum spanning tree of a weighted graph. Let $W \ge 2$ be an integer, G be a connected graph with integer edge weights from $[W] := \{1, \cdots, W\}$, and $c(\mathrm{MST})$ be the weight of an MST of G. For any $1 \le \ell \le W$, let $G^{(\ell)}$ denote the subgraph of G consisting of all edges of weight at most ℓ. Let $\mathrm{cc}^{(\ell)}$ denote the number of connected components of $G^{(\ell)}$. Chazelle et al. [8] give the following lemma relating the weight of MST to the number of connected components of $G^{(\ell)}$.

▶ **Lemma 5** ([8]). *It holds that* $c(MST) = n - W + \sum_{\ell=1}^{W-1} cc^{(\ell)}$.

For a connected graph with integer edge weights, the weight of any MST is at least $n-1$, so it is sufficient to estimate $cc^{(\ell)}$ within an additive error of $\varepsilon n/(W-1)$ for each ℓ. To do this, we can simply run $W-1$ parallel instances of Theorem 4, each of which sketches a subgraph $G^{(\ell)}$. Then the space of the algorithm will be $\tilde{O}(W n^{1-\frac{\varepsilon}{W-1}})$.

▶ **Theorem 6.** *Let $t \ge 1$ be any constant. There exists a single-pass dynamic streaming algorithm that uses space and post-processing time $O(e^t W n^{1-\frac{\varepsilon}{W-1}+\frac{\varepsilon^t}{(W-1)^t}} \operatorname{poly}\log n)$ to compute a $(1+\varepsilon)$-approximation of the weight of the MST.*

We remark that Ahn et al. [2] have given a dynamic streaming algorithm for this problem for any graph with maximum edge weight upper bounded by $O(\mathrm{poly}(n))$, and their algorithm uses space $O(n \cdot \operatorname{poly}\log n)$. Our algorithm uses $o(n)$ space for any connected graph with maximum edge weight bounded by $o(\log n)$ (for constant ε), which improves the algorithm of [2] in this setting. We also note that $\Omega(n)$ space is necessary for estimating the weight of MST for graphs with maximum edge weight at least $c \log n$ for constant ε and some large universal constant c (see Theorem 12). Finally, we remark that the algorithm can also be extended to the setting where non-integral weights are allowed (see [8] for more details).

4 Dynamic streaming testers

In this section, we give our streaming testers for k-edge connectivity and cycle-freeness. Due to space constraints, we present the testers for k-vertex connectivity, Eulerianity and planar graph bipartiteness in the full version of the paper.

Algorithm 2 TestConnectivity

1: Sample each vertex with probability $p := (\varepsilon n/10)^{-\varepsilon}$. If more than $16np$ vertices are sampled, abort and output **Fail**. Let S denote the set of sampled vertices.
2: For each $v \in S$, maintain an AMS sketch $AMS(\mathbf{\Delta}^v)$, sketching the neighborhood of v in G.
3: Maintain an AGM sketch of $G[S]$ using Theorem 1.
4: **Post-Processing:**
5: Use the above sketch to construct a spanning forest F of $G[S]$ as guaranteed by Theorem 1.
6: For each connected component $C \in F$, estimate $F_2(\mathbf{\Delta}^C)$ using the AMS sketch $AMS(\mathbf{\Delta}^C) = \sum_{v \in C} AMS(\mathbf{\Delta}^v)$. If the answer $\tilde{F}_2 = 0$, **Reject**.
7: **Accept**.

4.1 Testing k-edge connectivity

A graph is k-edge connected if the minimum cut of the graph has size at least k. We start from the simplest case, i.e., $k = 1$, which is equivalent to the problem of testing connectivity.

4.1.1 Connectivity

It is clear that if G is ε-far from being connected, one must add at least εm edges to make it connected, which implies that there are at least $\varepsilon m + 1$ connected components in G [22, 36]. Therefore, we can also solve this by estimating the number connected components by setting the error parameter appropriately, however, by a more careful analysis, we can improve this by a factor of $O(n^{O(\varepsilon)})$.

▶ **Theorem 7.** *There exists a dynamic streaming tester for* 1*-edge connectivity that runs in* $\tilde{O}(n^{1-\varepsilon})$ *post-processing time and space.*

Proof. First observe that one can simply reject the input graph if $m < n - 1$, since in this case, the graph is disconnected. Thus, in the following we assume $m \geq n - 1$ and our tester is described in Algorithm 2.

It is easy to see that Algorithm 2 only use $\tilde{O}(|S|)$ space, which is bounded by $\tilde{O}(np) = \tilde{O}(\varepsilon^{-\varepsilon} n^{1-\varepsilon}) = \tilde{O}(n^{1-\varepsilon})$. The post-processing time is nearly linear in the size of S, since the AGM algorithm needs $\tilde{O}(|S|)$ post-processing time, and we invoke at most $|S|$ AMS queries, each of which takes $\tilde{O}(1)$ time. The update time is $\operatorname{poly}\log n$.

For the correctness of the algorithm, we condition on the event that the number of sampled vertices is at most $16np$, which occurs with probability at least $1 - \frac{1}{16}$, and on the event that the spanning forest F is constructed correctly, which occurs with probability 0.99. By setting the error probability of the AMS sketch to be $1/n^2$ (with an extra $\log n$ factor in space), with probability 0.99, all the answers from AMS sketches are all correct, and we also condition on this.

If G is connected, then it will always be accepted, since for each $C \in F$, $\mathbf{\Delta}^C \neq \mathbf{0}$, and conditioned on the correctness of the AMS sketch, $\tilde{F}_2$ will never be 0. On the other hand, if the graph is ε-far from being connected, the number of connected components in G, denoted as $\operatorname{cc}(G)$, is at least $1 + \varepsilon m \geq \varepsilon n$. Let $B_1, \cdots, B_{\operatorname{cc}(G)}$ denote all connected components in G. Let $p_i = p^{|B_i|}$ for $1 \leq i \leq \operatorname{cc}(G)$. Using the inequality $1 - x \leq e^{-x}$ for all x, the probability that none of the components is entirely sampled out is $(1-p_1)\cdot(1-p_2)\cdot\cdots\cdot(1-p_{\operatorname{cc}(G)}) \leq e^{-\sum_i p_i}$. Then by the AM-GM inequality, this probability is at most

$$e^{-\operatorname{cc}(G)\cdot(\prod_i p_i)^{1/\operatorname{cc}(G)}} = e^{-\operatorname{cc}(G)\cdot p^{n/\operatorname{cc}(G)}} \leq e^{-\operatorname{cc}(G)\cdot p^{1/\varepsilon}} \leq e^{-\varepsilon n \cdot p^{1/\varepsilon}} \leq 1/16,$$

where we use the fact that $p = (\varepsilon n/10)^{-\varepsilon}$ and $\mathrm{cc}(G) \geq \varepsilon n$. So the probability that at least one of the components is sampled out is at least 15/16. Conditioned on this, $F_2(\boldsymbol{\Delta}^C) = 0$ for some component in $G[S]$ and the algorithm will output **Reject**. By union bound, our algorithm will succeed with probability $1 - \frac{1}{16} - 0.01 - 0.01 - \frac{1}{16} > 3/4$. ◀

4.1.2 k-edge connectivity: $k \geq 2$

By using a slightly more involved argument and replacing AMS sketches with $(k-1)$-sparse recovery sketches, we can generalize the above idea to testing k-edge connectivity for $k \geq 2$. We have the following theorem on testing k-edge connectivity. See the full version for the proof.

▶ **Theorem 8.** *Let $k \leq O(n^{\varepsilon/(1+\varepsilon)})$. There exists a single-pass dynamic streaming tester for k-edge connectivity with post-processing time and space $O(k^{1+\varepsilon} \cdot n^{1-\varepsilon} \cdot \mathrm{poly}\log n)$.*

4.2 Testing cycle-freeness

Now we consider the problem of testing cycle-freeness, which is equivalent to testing if the graph is a forest. Let $\mathrm{cc}(G)$ denote the number of connected components of the input graph G. Let $B_1, \cdots, B_{\mathrm{cc}(G)}$ be the connected components in G. Note that if G is cycle-free, then for each $i \leq \mathrm{cc}(G)$, $|E(B_i)| = |B_i| - 1$, and thus the total number of edges in G is

$$m = \sum_{i=1}^{\mathrm{cc}(G)} |E(B_i)| = \sum_{i=1}^{\mathrm{cc}(G)} (|B_i| - 1) = n - \mathrm{cc}(G),$$

that is, $\mathrm{cc}(G) = n - m$. If G is ε-far from being cycle-free, i.e., one has to delete more than εm edges to make it cycle-free, then $\mathrm{cc}(G) > n - m + \varepsilon m$. Therefore, to test cycle-freeness of a graph, it will be sufficient to approximate the number of connected components with additive error $\varepsilon m/2$. One may try to directly invoke Algorithm 1 with parameter $\varepsilon' = \frac{\varepsilon m}{2n}$. However, m could be much smaller than n and we do not know m in advance. We overcome this obstacle by a case analysis.

▶ **Theorem 9.** *There exists a single-pass dynamic streaming algorithm that tests cycle-freeness of a graph with space and post-processing time $O(n^{1-\varepsilon+\varepsilon^2} \cdot \mathrm{poly}\log n)$.*

Proof. Note that if $m > n - 1$, then the graph must contain at least one cycle, and thus we can safely reject the graph. In the following, we assume that $m \leq n - 1$. Our algorithm for testing cycle-freeness depends on the construction of AGM sketch, in which each vertex u maintains a linear sketch of $\boldsymbol{\Delta}^u$ (denoted as $\mathcal{A}(\boldsymbol{\Delta}^u)$). Each such sketch has size $\mathrm{poly}\log n$ and the property that $\mathcal{A}(\mathbf{0}) = \mathbf{0}$ (it consists of $O(\log n)$ l_0-samplers, see [2] for details). Our main idea is to maintain a sparse recovery sketch for the AGM sketch (i.e. a composition of sparse recovery sketch and AGM sketch). Now we describe our algorithm as follows.

Note that the space used by the algorithm is $\max\{\tilde{O}(np), k \cdot \mathrm{poly}\log n\} = \tilde{O}(n^{1-\varepsilon+\varepsilon^2})$, and the post-processing time is near linear in space. Now we prove the correctness of the algorithm. We define $G' \subseteq G$ to be a subgraph which consists of all the vertices of positive degree. Let $n' = |G'|$. Note that $m \geq n'/2$. If $n' \leq n^{1-\eta}$, then the vector $\boldsymbol{\Upsilon}$ is $\tilde{O}(n^{1-\eta})$-sparse, since for all isolated vertices u, we have $\mathcal{A}(\boldsymbol{\Delta}^u) = \mathbf{0}$, and thus we can recover the entire $\boldsymbol{\Upsilon}$ exactly. Then by Step (2) and Theorem 1, we can get the exact number of components of G'. Since the number of vertices of G' is $|Y|$, and $\lambda = m$ is the total number of edges, then the graph is cycle-free if and only if $\tilde{c}_1 = |Y| - \lambda$.

Algorithm 3 TestCycleFreeness

1: Maintain a count λ of the number of edges.
2: Let $\eta = \varepsilon/(1+\varepsilon+\varepsilon^2)$. Let $k = n^{1-\eta}\text{poly}\log n$. Maintain an exact k-sparse recovery sketch $\mathcal{S}$ of the vector $\boldsymbol{\Upsilon} := (\mathcal{A}(\boldsymbol{\Delta}^u))_{u\in V}$ using Lemma 2.
3: Run Algorithm 1 with parameter $p = (2^{2t}n^{1-\eta}/16)^{-\eta}$, while in step (6) of Algorithm 1, ignore all the isolated vertices that are sampled out (i.e., set $X_C = 0$ whenever $|C| = 1$).
4: **Post-Processing:**
5: Recover $\boldsymbol{\Upsilon}$ from $\mathcal{S}$.
6: **if** The recovery does not fail **then**
7: Use $\boldsymbol{\Upsilon}$ to construct a spanning forest on vertex set $Y := \{u : \mathcal{A}(\boldsymbol{\Delta}^u) \neq \mathbf{0}\}$ using Theorem 1. Let $\tilde{c}_1$ denote the number of connected components of this forest. If $\tilde{c}_1 = |Y| - \lambda$, **Accept**; otherwise, **Reject**.
8: **else**
9: Let $\tilde{c}_2$ be the resulting estimator of Algorithm 1 in Step 3. If $\tilde{c}_2 \leq n - (1-\varepsilon-\frac{\varepsilon^3}{4})\lambda$, **Accept**; otherwise, **Reject**.
10: **end if**

If $n' > n^{1-\eta}$, then by Theorem 3, $\tilde{c}_2$ is an estimator for the number of components in G' of size smaller than $1/\eta$ with additive error $\eta^t\sqrt{n'n^{1-\eta}}$. This follows by the upper bound $\eta^{2t}n^{1-\eta}n'/16$ of the variance of the estimator (which can be shown similarly to inequality (1) in Section 3) and the Chebyshev's inequality. Now note that the additive error is at most $\eta^t n' \leq \varepsilon^3 m/8$ for some constant t since $n' > n^{1-\eta}$ and $m \geq n'/2$. Let L be the number of components in G' of size larger than $1/\eta$, then $-\varepsilon^3 m/8 \leq \text{cc}(G') - \tilde{c}_2 \leq L + \varepsilon^3 m/8$ holds with high probability. Conditioned on this, Step (8) outputs the correct answer if $L + \varepsilon^3 m/8 + \varepsilon^3 m/8 = L + \varepsilon^3 m/4 < \varepsilon m$. Now if $L < \varepsilon m/4$, we are done. If $L \geq \varepsilon m/4$, then by our choice of η and the fact that $m \geq L \cdot (1/\eta - 1)$, $\varepsilon m \geq L + L\varepsilon^2 \geq L + \varepsilon^3 m/4$. This completes the proof of the theorem. ◀

5 Lower bounds

In this section we present lower bounds, which hold in the insertion-only model. Our proofs are based on the reductions to the *Boolean Hidden Hypermatching (BHH)* problem (See [41]), which are in the same spirit as the lower bound proof for the *Cycle Counting* problem in [41]. We first give the definition of the boolean hidden hypermatching problem.

▶ **Definition 10** (BHH_n^t)**.** In this problem, Alice gets a boolean vector $x \in \{0,1\}^n$, where $n = 2kt$ for some integer k. Bob gets a partition (or hypermatching) of the set $[n]$, $\{m_1, \cdots, m_{n/t}\}$, where the size of each m_i is t, and a vector $w \in \{0,1\}^{n/t}$. For convenience, we will also use the corresponding n-dimensional boolean indicator vector M_i to represent m_i, and let M be a $n/t \times n$ matrix, the i row of which is M_i. The promise of the input is either $Mx + w = \mathbf{1}$ or $Mx + w = \mathbf{0}$, where all the operations are modulo 2. The goal of the problem is to output 1 when $Mx + w = \mathbf{1}$, and output 0 otherwise.

▶ **Theorem 11** ([41])**.** *The randomized one-way communication complexity of BHH_n^t when $n = 2kt$ for some integer $k \geq 1$ is $\Omega(n^{1-1/t})$.*

Our lower bounds will be built upon the following basic construction.

Construction of $G(x, M)$. Given vector x and matrix M respectively, Alice and Bob construct a bipartite graph $G(x, M) = (U, V, E)$, where $U = \{u_1, \cdots, u_{2n}\}$ and $V =$

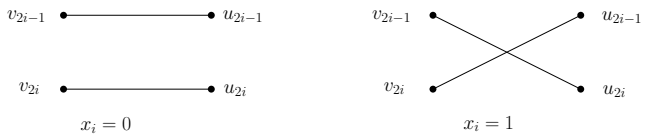


Figure 1 Parallel (left) and crossing (right) matching according to the value of x_i.

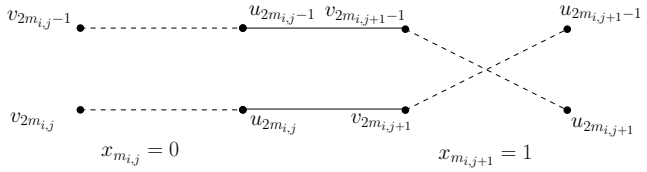


Figure 2 Bob connects $(u_{2m_{i,j}-1}, v_{2m_{i,j+1}-1})$ and $(u_{2m_{i,j}}, v_{2m_{i,j+1}})$ for each $j \in [t-1]$.

$\{v_1, \cdots, v_{2n}\}$, as follows. Given $x \in \{0,1\}^n$, Alice adds a perfect matching between U and V. For each $i \in [n]$, if $x_i = 0$, she adds two parallel edges (u_{2i-1}, v_{2i-1}) and (u_{2i}, v_{2i}); otherwise if $x_i = 1$, she adds two crossing edges (u_{2i-1}, v_{2i}) and (u_{2i}, v_{2i-1}) (see Figure 1).

Given M, Bob will do the following. For each $i \in [n/t]$ and the hyperedge $m_i \subset [n]$ (that corresponds to the ith row M_i), we use $m_{i,j} \in [n]$ to denote the jth element in m_i and we let $S_i := \{w | w = v_{2m_{i,j}-1}$ or $v_{2m_{i,j}}$ or $u_{2m_{i,j}-1}$ or $u_{2m_{i,j}}, j \in [t]\}$. For each $i \in [n/t]$ and $j \in [t-1]$, Bob adds two edges $(u_{2m_{i,j}-1}, v_{2m_{i,j+1}-1})$ and $(u_{2m_{i,j}}, v_{2m_{i,j+1}})$ (See Figure 2).

Observe that the edges added by Alice and Bob form two paths p_{2i-1}, p_{2i} over vertex set S_i, where p_{2i-1} starts from $v_{2m_{i,1}-1}$ and p_{2i} starts from $v_{2m_{i,1}-1}$ for each i. The entire graph $G(x, M)$ consists of $2n/t$ disjoint paths $\{p_1 \cdots, p_{2n/t}\}$.

Note that $G(x, M)$ has the following property. Based on the value of $(Mx)_i$, we have: 1) if $(Mx)_i = 0$, then p_{2i-1} is a path from $v_{2m_{i,1}-1}$ to $u_{2m_{i,t}-1}$ and p_{2i} is a path from $v_{2m_{i,1}}$ to $u_{2m_{i,t}}$; 2) if $(Mx)_i = 1$, then p_{2i-1} is a path from $v_{2m_{i,1}-1}$ to $u_{2m_{i,t}}$ and p_{2i} is a path from $v_{2m_{i,1}}$ to $u_{2m_{i,t}-1}$.

5.1 Minimum spanning tree

▶ **Theorem 12.** *In the insertion-only model, if all edges of the graph have weights in $[W]$, any algorithm that $(1 \pm \varepsilon)$-approximates the weight of the MST must use $\Omega(n^{1-\frac{4\varepsilon}{W-1}})$ bits of space.*

Proof. Given x and M, Alice and Bob first construct the graph $G(x, M)$ as describe above. Next Bob adds $(u_{2m_{i,t}-1}, v_{2m_{i,1}-1})$ and $(u_{2m_{i,t}}, v_{2m_{i,1}})$ if $w_i = 0$; adds $(u_{2m_{i,t}-1}, v_{2m_{i,1}})$ and $(u_{2m_{i,t}}, v_{2m_{i,1}-1})$ if $w_i = 1$. The weight of all the edges added so far is 1. Next Bob places edges $(v_{2m_{i,t}}, v_{2m_{i+1,1}})$ with weight 1 for $i = 1, \cdots, n/t - 1$ and edges $(v_{2m_{i,t}}, u_{2m_{i,t}})$ with weight W for each $i \in [n/t]$, so that the graph become connected. By similar argument as above, if $Mx + w = \mathbf{0}$, all the edges $(v_{2m_{i,t}}, u_{2m_{i,t}})$ must be picked in any minimum spanning tree, since each of these edges forms a cut, and thus the weight of any MST is $nW/t + 4n - n/t - 1 = 4n\varepsilon + 4n - 1$, where we set $t = (W-1)/4\varepsilon$. On the other hand, when $Mx + w = \mathbf{1}$, the weight of the MST is $4n - 1$, since in this case, the graph is already connected without those edges with weight W. So if the algorithm can compute an $(1+\varepsilon)$-approximation of the weight of the minimum spanning tree, it solves the BHH_n^t problem. This completes the proof. ◀

5.2 Testing connectivity

▶ **Theorem 13.** *In the insertion-only model, to distinguish whether a graph of $4n$ vertices is connected or $\frac{1}{8t+1}$-far from being connected, any algorithm must use $\Omega(n^{1-1/t})$ bits of space.*

Proof. Given x and M, Alice and Bob first construct the graph $G(x, M)$. Next Bob adds another set of edges based on vector w. If $w_i = 0$, he adds $(u_{2m_{i,t}-1}, v_{2m_{i,1}-1})$ and $(u_{2m_{i,t}}, v_{2m_{i,1}})$; if $w_i = 1$, he adds $(u_{2m_{i,t}-1}, v_{2m_{i,1}})$ and $(u_{2m_{i,t}}, v_{2m_{i,1}-1})$. So when $(Mx)_i + w_i = 0$, p_{2i-1} and p_{2i} become 2 disjoint cycles. On the other hand, when $(Mx)_i + w_i = 1$, p_{2i-1} and p_{2i} together form a larger cycle. Now Bob places $(v_{2m_{i,t}}, v_{2m_{i+1,1}})$ in E for $i = 1, \cdots, n/t - 1$ which connect p_{2i} with $p_{2(i+1)}$ for all $i \in [n/t - 1]$, i.e. all the paths in $G(x, M)$ with even indices become a connected component. The total number of edges is $8n + n/t$. When $Mx + w = \mathbf{0}$, the graph has $n/t + 1$ components which is $\frac{1}{8t+1}$-far from connected; when $Mx + w = \mathbf{1}$ the graph is connected. So if a streaming algorithm can distinguish whether a graph of size $4n$ is connected or $1/8t$-far from being connected, it solves BHH_n^t, since Alice can first run the algorithm on her part of the graph and send the memory to Bob, and then Bob continues to run the algorithm on his part and output the answer. Therefore, the communication lower bound of BHH_n^t implies a space lower bound of testing connectivity. ◀

5.3 Testing cycle-freeness

As in the proof of Theorem 13, given x and M, Alice and Bob first construct $G(x, M)$. Then, for $i \in [n/t]$, Bob adds $(u_{2m_{i,t}-1}, v_{2m_{i,1}-1})$ if $w_i = 0$; adds $(u_{2m_{i,t}-1}, v_{2m_{i,1}})$ if $w_i = 1$. The total number of edges is less than $8n$. Through similar arguments, it is easy to verify that if if $Mx + w = \mathbf{0}$, the graph has exactly n/t cycles and n/t paths, which is $1/8t$-far from cycle-free. On the contrary, if $Mx + w = \mathbf{1}$, the graph has n/t paths and no cycle. So if an algorithm can distinguish whether a graph of size $4n$ is cycle-free or $1/8t$-far from cycle-free, it solves BHH_n^t.

▶ **Theorem 14.** *In the insertion-only model, any algorithm that can distinguish whether a graph of $4n$ vertices is cycle-free or $1/8t$-far from being cycle-free, must use $\Omega(n^{1-1/t})$ bits of space.*

5.4 Testing bipartiteness of planar graphs

Alice and Bob first construct the graph $G(x, M)$. Next, for each $i \in [n/t]$, Bob adds edges $(v_{2m_{i,1}-1}, \xi_1)$ and $(v_{2m_{i,1}}, \xi_2)$, where ξ_1, ξ_2 are new vertices. For $i \in [n/t]$, Bob also adds $(u_{2m_{i,t}-1}, \xi_1)$ and $(u_{2m_{i,t}}, \xi_2)$ if $w_i = 0$; adds $(u_{2m_{i,t}-1}, \xi_2)$ and $(u_{2m_{i,t}}, \xi_1)$ if $w_i = 1$. For this problem we assume t is odd. So by similar arguments, we can easily verify that, if $Mx + w = \mathbf{0}$, the graph contains $2n/t$ edge-disjoint cycles of length $2t + 1$, and if $Mx + w = \mathbf{1}$, the graph has no odd cycle, and thus bipartite. The graph constructed is planar and has $4n + 2$ vertices and $8n + 4n/t$ edges, so we have the following lower bound for testing bipartiteness.

▶ **Theorem 15.** *In the insertion-only model, any algorithm that can distinguish whether a planar graph of $4n + 2$ vertices is bipartite or $\frac{1}{4t+2}$-far from being bipartite, must use $\Omega(n^{1-1/t})$ bits space.*

References

1. Kook Jin Ahn, Graham Cormode, Sudipto Guha, Andrew McGregor, and Anthony Wirth. Correlation clustering in data streams. In *Proceedings of the 32nd International Conference on Machine Learning, ICML*, pages 6–11, 2015.
2. Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 459–467. SIAM, 2012.
3. Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st symposium on Principles of Database Systems*, pages 5–14. ACM, 2012.
4. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 20–29. ACM, 1996.
5. Sepelir Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 1345–1364. SIAM, 2016. URL: `http://dl.acm.org/citation.cfm?id=2884435.2884528`.
6. Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E Tsourakakis. Space-and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *ACM Symposium on Theory of Computing*, 2015.
7. Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. *ESA*, 2015.
8. Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM Journal on computing*, 34(6):1370–1379, 2005.
9. Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vorotnikova. Kernelization via sampling with applications to dynamic graph streams. *SODA*, 2016.
10. Rajesh Chitnis, Graham Cormode, MohammadTaghi Hajiaghayi, and Morteza Monemizadeh. Parameterized streaming: maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1234–1251. SIAM, 2015.
11. Artur Czumaj, Funda Ergün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 35(1):91–109, 2005.
12. Artur Czumaj, Morteza Monemizadeh, Krzysztof Onak, and Christian Sohler. Planar graphs: Random walks and bipartiteness testing. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 423–432. IEEE, 2011.
13. Artur Czumaj and Christian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 39(3):904–922, 2009.
14. Hossein Esfandiari, Mohammad T Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1217–1233. SIAM, 2015.
15. Hossein Esfandiari, MohammadTaghi Hajiaghayi, and David P Woodruff. Applications of uniform sampling: Densest subgraph and beyond. *arXiv preprint arXiv:1506.04505*, 2015.
16. Stefan Fafianie and Stefan Kratsch. Streaming kernelization. In *Mathematical Foundations of Computer Science 2014*, pages 275–286. Springer, 2014.

17 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2):207–216, 2005.

18 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM Journal on Computing*, 38(5):1709–1727, 2008.

19 Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 142–149. ACM, 2005.

20 Oded Goldreich. Introduction to testing graph properties. In *Property testing*, pages 105–141. Springer, 2011.

21 Oded Goldreich, Shari Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

22 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32:302–343, 2002.

23 Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems*, pages 241–247. ACM, 2015.

24 Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. In *External Memory Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, May 20-22, 1998*, pages 107–118, 1998.

25 Hossein Jowhari. Estimating the number of connected components in graph streams. Personal communication.

26 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–751. SIAM, 2014.

27 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Streaming lower bounds for approximating max-cut. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1263–1282. SIAM, 2015.

28 Michael Kapralov, Yin Tat Lee, Christopher Musco, and Aaron Sidford. Single pass spectral sparsification in dynamic streams. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 561–570. IEEE, 2014.

29 Michael Kapralov and David Woodruff. Spanners and sparsifiers in dynamic streams. In *Proceedings of the 2014 ACM symposium on Principles of distributed computing*, pages 272–281. ACM, 2014.

30 Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 367–376. ACM, 2015.

31 Christian Konrad. Maximum matching in turnstile streams. *ESA*, 2015.

32 Andrew McGregor. Graph stream algorithms: A survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.

33 Andrew McGregor, David Tench, Sofya Vorotnikova, and Hoa T Vu. Densest subgraph in dynamic graph streams. *MFCS*, 2015.

34 S Muthukrishnan. Data streams: Algorithms and applications. *Theoretical Computer Science*, 1(2):117–236, 2005.

35 Yaron Orenstein and Dana Ron. Testing eulerianity and connectivity in directed sparse graphs. *Theoretical Computer Science*, 412(45):6390–6408, 2011.

36 Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Structures & Algorithms*, 20(2):165–183, 2002.

37 Eric Price. Efficient sketches for the set query problem. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 41–56. SIAM, 2011.

38 Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends® in Theoretical Computer Science*, 5(2):73–205, 2010.

39 Ronitt Rubinfeld and Asaf Shapira. Sublinear time algorithms. *SIAM Journal on Discrete Mathematics*, 25(4):1562–1588, 2011.

40 Xiaoming Sun and David P. Woodruff. Tight bounds for graph problems in insertion streams. In *The 18th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'2015)*, 2015.

41 Elad Verbin and Wei Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 11–25. SIAM, 2011.

42 Yuichi Yoshida and Hiro Ito. Property testing on k-vertex-connectivity of graphs. In *Automata, Languages and Programming*, pages 539–550. Springer, 2008.

Diameter and k-Center in Sliding Windows*

Vincent Cohen-Addad[1], Chris Schwiegelshohn[2], and Christian Sohler[3]

1 **Département d'informatique, École normale supérieure, Paris, France**
`vincent.cohen@ens.fr`
2 **Efficient Algorithms and Complexity Theory, TU Dortmund, Dortmund, Germany**
`chris.schwiegelshohn@tu-dortmund.de`
3 **Efficient Algorithms and Complexity Theory, TU Dortmund, Dortmund, Germany**
`christian.sohler@tu-dortmund.de`

Abstract

In this paper we develop streaming algorithms for the diameter problem and the k-center clustering problem in the sliding window model. In this model we are interested in maintaining a solution for the N most recent points of the stream. In the diameter problem we would like to maintain two points whose distance approximates the diameter of the point set in the window. Our algorithm computes a $(3+\epsilon)$-approximation and uses $O(1/\epsilon \ln \alpha)$ memory cells, where α is the ratio of the largest and smallest distance and is assumed to be known in advance. We also prove that under reasonable assumptions obtaining a $(3-\epsilon)$-approximation requires $\Omega(N^{1/3})$ space.

For the k-center problem, where the goal is to find k centers that minimize the maximum distance of a point to its nearest center, we obtain a $(6+\epsilon)$-approximation using $O(k/\epsilon \ln \alpha)$ memory cells and a $(4+\epsilon)$-approximation for the special case $k=2$. We also prove that any algorithm for the 2-center problem that achieves an approximation ratio of less than 4 requires $\Omega(N^{1/3})$ space.

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

Keywords and phrases Streaming, k-Center, Diameter, Sliding Windows

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.19

1 Introduction

Analyzing big data sets from streams is a topic that has received considerable attention among theoretical and applied researchers. One of the most popular summarization and aggregation tasks studied in this context is to determine k clusters that represent key features of the input data with respect to certain properties.

In this paper we focus on variants of the k-center problem where we aim to find k points such that the maximum distance over all points to their closest center is minimized.

In the standard streaming setting, we constrain our algorithms to use as little space as possible while computing high-quality solutions. The complexity of clustering in general and k-center in particular is well understood for insertion-only streams where input points arrive one by one. The more general settings like dynamic streams and the sliding window

* Supported by Deutsche Forschungsgemeinschaft within the Collaborative Research Center SFB 876, project A2

model have also received some attention for other clustering objectives. Both models aim to incorporate dynamic behavior; in dynamic streams input points are removed via a dedicated delete operation and in the sliding window model older input expires as new elements arrive. In this paper, we consider a fixed window size consisting of N points, which is the most widely studied variant of this model in theoretical computer science, but our algorithms also work in the case that the maximum number of points within the window is a function of time.

Our Contribution

The metric diameter problem is to find two points of maximum distance among a set of points lying in some metric space. For this problem, we give a $(3+\varepsilon)$-approximation algorithm in the sliding window model that stores $O(\frac{1}{\varepsilon}\log\alpha)$ points, where $\alpha = \frac{\max_{p,q}\operatorname{dist}(p,q)}{\min_{p,q}\operatorname{dist}(p,q)}$ is the ratio of largest and smallest possible distance between the points. This is a substantial improvement over the best and to our knowledge only sliding window algorithm for diameter in general metric spaces by Chan and Sadjad [8] which computes a $(2^{m+2}-2+\varepsilon)$ approximation with $O(N^{1/(m+1)}\log\alpha)$ points for any $m > 0$.

Under reasonable assumptions (which are common for our model of computation), we obtain a lower bound of $\Omega(\sqrt[3]{N})$ for any algorithm achieving a $3-\varepsilon$ approximation to the diameter problem for any $\varepsilon > 0$.

To our knowledge there exists no previous work on k-center in sliding windows. For 2 centers our diameter algorithm yields a $(4+\varepsilon)$-approximate clustering. Under the aforementioned assumptions, we are also able to obtain a matching lower bound. For arbitrary values of k, we are able to obtain a $(6+\varepsilon)$-approximate algorithm using $O(k/\varepsilon\log\alpha)$ points in metric spaces.

Techniques

The popular histogram approaches introduced by Datar et al. [12] and Braverman and Ostrovsky [6] do not seem to be applicable to max-norm objectives such as diameter and k-center. Our algorithm for the diameter (see Section 3) aims to find for each estimate of the value γ of the diameter two certificate points with distance greater than γ, while maintaining the two most recent points close to the two points forming the certificate. With every additional input point, we check whether we are able to update the certificate to a more recent timestamp.

For our lower bounds, we utilized the fact that any algorithm working in the metric distance model is restricted to storing only input points. For deterministic algorithms, we are then able to insert an appropriately hard instance based on the points forgotten by the input. For randomized algorithms, we add additional points in which we hide a hard, randomly chosen instance for deterministic algorithms. A more in-depth description of our approach as well as a discussion on the generality of our results can be found in Section 5.

The analysis of the 2-center algorithm is similar to that of the popular 2-approximation by Gonzales [14] and Hochbaum and Shmoys [16]. Since it does not yield an approximation of the optimum value, this technique seems difficult to generalize for larger values of k.

Related Work

Diameter

Feigenbaum et al. [13] were the first to consider the diameter in the sliding window model. For d dimensions in Euclidean space, their algorithm uses $O(\left(\frac{1}{\varepsilon}\right)^{(d+1)/2}\log^3 N(\log\alpha+\log\log N+$

$\frac{1}{\varepsilon}$)) bits of space. They also give a lower bound of $\Omega(\frac{1}{\varepsilon}\log N \log \alpha)$ for a $(1+\varepsilon)$ approximation factor in one dimension and, implicitly, a $\Omega(\log \alpha)$ space bound for any multiplicative approximation factor. This lower bound was later matched by Chan and Sadjad [8], who also gave an improved space bound of $O((\frac{1}{\varepsilon})^{(d+1)/2} \log \frac{\alpha}{\varepsilon})$ points for higher dimensions. For more general metric spaces, they obtain a $(2^{m+2}-2+\varepsilon)$ approximation with $O(N^{1/(m+1)})$ points.

In the metric distance oracle model (formally defined in Section 2) there exists a folklore 2 approximation that maintains the first point p and the point with maximum distance from p. Guha [15] showed this algorithm to be essentially optimal, as no algorithm storing less than $\Omega(n)$ points can achieve a ratio better than $2-\varepsilon$ for any $\varepsilon > 0$. For Euclidean spaces, the best streaming algorithm with a polynomial dependency on d is due to Agarwal and Sharathkumar [2] with an almost tight approximation ratio of $\sqrt{2}+\varepsilon$ in $O(d\varepsilon^{-3}\log(1/\varepsilon))$ space. Agarwal et al. [1] proposed a $(1+\varepsilon)$-approximation using $O(\varepsilon^{-(d-1)/2})$ points. Similar space bounds seem likely for dynamic streams although none have been published to our knowledge. For large d, Indyk [17] gave a sketching scheme with approximation factor $c > \sqrt{2}$ and space $O(dn^{1/(c^2-1)}\log n)$.

k-Center

In one of the earliest works on clustering in streams, Charikar et al. [9] gave a number of incremental clustering algorithms for metric k-center, among other results. While storing no more than $k+1$ points at any given time, they were able to derive a deterministic 8 approximation and a randomized $2e \approx 5.437$ approximation. They also show that no incremental algorithm can be better than 3. McCutchen and Khuller [19] and Guha [15] independently derived a $(2+\varepsilon)$-approximate algorithm using $O(k/\varepsilon \log 1/\varepsilon)$ space, with Guha giving an almost tight lower bound of $\Omega(n)$ space for any algorithm achieving a better approximation ratio than 2. In their paper, McCutchen and Khuller [19] also studied the problem with z outliers, giving a $(4+\varepsilon)$ approximate algorithm that stores $O(\varepsilon^{-1}kz)$ points, see also Charikar et al. [10] for an earlier treatment of the problem. Further improvements are possible in Euclidean spaces. Zarrabi-Zadeh showed how to maintain ε-coresets in streams using $O(k\varepsilon^{-d})$ points for k-center [20]. For small values of k, Kim and Ahn [18] were able to break the 2 barrier without having an exponential dependency on d, giving a $1.8+\varepsilon$ approximation while storing $O(2^k(k+3)!\varepsilon^{-1})$ points. The special case of $k=1$ for Euclidean distances also known as the minimum enclosing ball problem is one of the most extensively studied topics in streaming literature. We only review the best known bounds. Zarrabi-Zadeh gave an insertion-only algorithm storing $O(\varepsilon^{-(d-1)/2}\log 1/\varepsilon)$ points [21]. Agarwal and Sharathkumar [2] showed that no algorithm with polynomial dependency on d can achieve a better approximation ratio than $(1+\sqrt{2})/2 \approx 1.207$ and provided an algorithm which after a re-analysis by Chan and Pathak [7] is now known to give a 1.22 approximation with space roughly $O(d)$.

We are not aware of any work on k-center in sliding windows, though Babcock et al. [3] and more recently Braverman et al. [4, 5] gave a $O(1)$ approximation for metric and a $(1+\varepsilon)$ approximation for Euclidean k-median and k-means problems. The related problem of cut sparsification has also received some attention, see Crouch et al. [11].

The structure of the paper is as follows. Section 2 introduces the model and the definitions. Section 3 is dedicated to the description and analysis of our algorithm for the diameter problem. The analysis and description of our algorithm for the k-center problem is in Section 4. Finally, Section 5 contains the lower bounds for both the diameter and k-center problems.

Algorithm 1 Sliding Window Algorithm for $(\gamma, 3 \cdot \gamma)$-gap Diameter

```
1:  c_old, q, r ← first point of the stream;
2:  c_new ← null;
3:  for all element p of the stream do
4:      if certificate point c_old expires then
5:          if (c_new ≠ null ∧ c_old = q) then
6:              c_old ← r; c_new ← null;
7:          if (c_new ≠ null ∧ c_old ≠ q) then
8:              c_old ← q; c_new ← null;
9:          if c_new = null then
10:             c_old ← r;
11:     Insert(p);
12:     r ← p;
13: procedure Insert(p)
14:     if c_new = null then
15:         if dist(p, r) > γ then
16:             c_old, q ← r; c_new ← p;
17:         else if dist(p, c_old) > γ then
18:             q ← r; c_new ← p;
19:     else
20:         if dist(p, r) > γ then
21:             c_old, q ← r; c_new ← p;
22:         else if dist(p, c_new) > γ then
23:             c_old ← c_new; q ← r; c_new ← p;
24:         else if dist(p, q) > γ then
25:             if c_old ≠ q then
26:                 c_old ← q; q ← r; c_new ← p;
```

2 Preliminaries

Let (A, dist) be a metric space where A is a set of points and $\text{dist} : A \times A \mapsto R_+$ is a distance function. A stream is a (potentially infinite) sequence of points from the metric space A (note that a point can be appear multiple times in the stream). The sliding window of size N contains the most recent N elements of the stream.

We introduce the *Time To Live* value of a point p: Upon insertion $TTL(p)$ is set to the window size N and with each subsequently inserted point it is decremented. We say that p *expires* if $TTL(p) = 0$. We extend the common use of TTL to negative numbers to indicate the number of points submitted after expiration, i. e., $TTL(p) = -10$ means that 10 points were submitted after the expiration of p. We define the aspect ratio $\alpha = \frac{\max_{p,q \in A} \text{dist}(p,q)}{\min_{p,q \in A} \text{dist}(p,q)}$. To query the distance between two points p and q, we invoke a distance oracle $\text{dist}(p, q)$. We assume that the oracle can accessed only for those points we currently keep in memory and that the oracle itself requires no additional space.

3 The Metric Diameter Problem

For a given estimate γ of the diameter, our algorithm for the metric diameter problem either produces two witness points at distance greater than γ or a point c that has a certain degree of centrality among the points in the current window. More formally, all points of the window inserted up to the insertion time of c will be proven to have distance at most 2γ from one another and points inserted after c will have distance at most γ from c. Thus, the diameter is at most 3γ.

Specifically, Algorithm 1 aims at maintaining a certificate for the diameter consisting of two points c_{old} and c_{new} such that $\text{dist}(c_{old}, c_{new}) > \gamma$ and $TTL(c_{old}) < TTL(c_{new})$. In addition, we also store the point q submitted immediately prior to c_{new} and the most recent point r. When a new point arrives, we test whether, based on the points we currently keep in memory, we can produce two points each with a larger TTL than $TTL(c_{old})$ with distance more than γ. If we find such a pair, we update the points accordingly, if not we update r and possibly q.

The algorithm has two different states depending on whether it found a pair of points of distance more than γ or not. The first state is indicated by $c_{new} = \textbf{null}$ and corresponds to the case that no such pair of points has been found. In this case, the algorithm maintains the following invariant, which certifies that the diameter of the points in the sliding window is at most $3 \cdot \gamma$.

We first observe that c_{old} is always inside the sliding window.

▶ **Invariant 1.** If $c_{new} = \textbf{null}$, the following statements hold:
(a) For any points a, b with $0 \leq TTL(a), TTL(b) \leq TTL(c_{old})$, we have $\text{dist}(a, b) \leq 2 \cdot \gamma$.
(b) For any point a with $TTL(a) > TTL(c_{old})$, we have $\text{dist}(a, c_{old}) \leq \gamma$.

The second state corresponds to the case that we discover two points c_{old} and c_{new} with distance more than γ and is indicated by $c_{new} \neq \textbf{null}$. Besides the obvious invariant that $TTL(c_{new}) > TTL(c_{old})$, we also have to maintain the following technical invariants that are required for a new assignment of c_{old} when it expires from the window.

▶ **Invariant 2.** If $c_{new} \neq \textbf{null}$ then the following statements hold:
(a) $\text{dist}(c_{old}, c_{new}) > \gamma$.
(b) For any point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, we have $\text{dist}(a, c_{old}) \leq \gamma$.
(c) For any point a with $TTL(c_{new}) < TTL(a)$, we have $\text{dist}(a, c_{new}) \leq \gamma$.
(d) If $c_{old} \neq q$ then for any point a with $TTL(q) < TTL(a)$, we have $\text{dist}(a, q) \leq \gamma$.

We observe that all the invariants hold initially, i.e. before line 3 of the algorithm is executed the first time. We also observe that Invariants 2a)-d) only apply to points that appear after c_{old}. It suffices to focus on these points because we only change c_{old} to points that arrive later and so we maintain our certificate at least until the time when c_{old} expires (and so all earlier points are gone).

▶ **Lemma 1.** *If $c_{new} =$ **null** and Invariant 1 is satisfied before* Insert *then one of the following statements holds:*
1. *If $c_{new} =$ **null** after line 12 then Invariant 1 is satisfied.*
2. *If $c_{new} \neq$ **null** after line 12 then Invariant 2 is satisfied.*

Proof. We never execute lines 19–26 of Insert. If $\text{dist}(p, r) > \gamma$ then $c_{new} \neq \textbf{null}$ and Invariant 2(a) holds due to line 16, Invariant 2(b) holds due to the fact that there exists no point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, Invariant 2(c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 2(d) holds due to $c_{old} = q$. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{old}) > \gamma$ then $c_{new} \neq \textbf{null}$, and Invariant 2(a) holds due to line 18, Invariant 2(b) holds due to Invariant 1(b), Invariant 2(c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 2(d) holds due to $r = q$ (before line 12) and line 15. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{old}) \leq \gamma$ then Invariant 1 continues to be satisfied for all points with TTL smaller than p and for the point p due to line 17. ◀

▶ **Lemma 2.** *If $c_{new} \neq$ **null** and Invariant 2 is satisfied before* Insert*, then Invariant 2 is satisfied after line 12.*

Proof. If $\text{dist}(p, r) > \gamma$ then Invariant 2(a) holds due to line 21, Invariant 2(b) holds due to the fact that there exists no point a with $TTL(c_{old}) < TTL(a) < TTL(c_{new})$, Invariant 2(c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 2(d) holds due to $c_{old} = q$. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{new}) > \gamma$ then Invariant 2(a) holds due to line 23, Invariant 2(b) and 2(d) hold due to Invariant 2(c) before Insert, and Invariant 2(c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$. If $\text{dist}(p, r) \leq \gamma$

and $\text{dist}(p, c_{new}) \leq \gamma$, and $\text{dist}(p, q) > \gamma$ and $q \neq c_{old}$ then Invariant 2(a) holds due to line 26, Invariant 2(b) holds due to Invariant 2(d) before INSERT, Invariant 2(c) holds due to the fact that there exists no point a with $TTL(c_{new}) < TTL(a)$, and Invariant 2(d) holds due to $q = r$ and line 20. If $\text{dist}(p, r) \leq \gamma$ and $\text{dist}(p, c_{new}) \leq \gamma$, and $\text{dist}(p, q) > \gamma$ and $q = c_{old}$ or $\text{dist}(p, q) \leq \gamma$, the Invariants 2(a)–(d) hold for all points except for the newest, for which the invariants hold due to lines 20, 22, 24 and 25. ◀

▶ **Lemma 3.** *If $c_{new} =$ **null** and Invariant 1 is satisfied before the line 4, then it is satisfied before INSERT (line 11).*

Proof. If c_{old} does not expire, then the claim obviously holds. Otherwise we execute line 10. Let c'_{old} be the expired point. Then we have for any two points a, b with $TTL(c'_{old}) < TTL(a), TTL(b) \leq TTL(r)$ $\text{dist}(a, c'_{old}), \text{dist}(b, c'_{old}) \leq \gamma$ due to Invariant 1(b) before line 4 and hence $\text{dist}(a, b) \leq 2\gamma$. Invariant 1(b) follows from $c_{old} = r$. ◀

▶ **Lemma 4.** *If $c_{new} \neq$ **null** and Invariant 2 is satisfied before the line 4, then one of the following statements holds before INSERT (line 11):*
1. *$c_{new} =$ **null** and Invariant 1 is satisfied.*
2. *$c_{new} \neq$ **null** and Invariant 2 is satisfied.*

Proof. If c_{old} does not expire the second claim immediately holds. Otherwise, we denote by c'_{old} the expired point. If $c'_{old} = q$, then we execute line 6. We set $c_{old} = r$, naturally satisfying Invariant 1(b). Further, at this time we have $TTL(c_{new}) = 1$, and for any two points a, b with $TTL(c_{new}) \leq TTL(a), TTL(b) \leq TTL(r)$ we have $\text{dist}(a, c_{new}), \text{dist}(b, c_{new}) < \gamma$ due to Invariant 2(c) before line 4 and hence $\text{dist}(a, b) \leq 2\gamma$, satisfying Invariant 1(a).

If $c'_{old} \neq q$, then we execute line 8. We set $c_{old} = q$. For any two points a, b with $TTL(c'_{old}) < TTL(a), TTL(b) < TTL(c_{new})$ $\text{dist}(a, c'_{old}), \text{dist}(b, c'_{old}) \leq \gamma$ due to Invariant 2(b) before line 4 and hence $\text{dist}(a, b) \leq 2\gamma$, satisfying Invariant 1(a). For all points a with $TTL(a) > q$, Invariant 1(b) after line 12 follows from Invariant 2(d) before line 4. ◀

The proof of the theorem is a direct consequence of the invariants but included for completeness.

▶ **Theorem 5.** *Given a set of points A with aspect ratio α and a window of size N, there exists an algorithm computing a $3(1+\epsilon)$-approximate solution for the metric diameter problem storing at most $8/\epsilon \cdot \ln \alpha$ points. The update time per point is $O(\varepsilon^{-1} \log \alpha)$.*

Proof. For any given estimate γ, we either have two points at distance at least γ, or Invariant 1 holds. In the latter case, we can bound the maximum diameter of two points p and q via the following case analysis.
$\boldsymbol{TTL(p), TTL(q) \leq TTL(c_{old})}$**:** Then $\text{dist}(p, q) \leq 2\gamma$.
$\boldsymbol{TTL(p) \leq TTL(c_{old}) < TTL(q)}$**:** Then $\text{dist}(p, q) \leq \text{dist}(p, c_{old}) + \text{dist}(c_{old}, q) \leq 3\gamma$.
$\boldsymbol{TTL(c_{old}) < TTL(p), TTL(q)}$**:** Then $\text{dist}(p, q) \leq \text{dist}(p, c_{old}) + \text{dist}(c_{old}, q) \leq 2\gamma$.
Now define an exponential sequence to the base of $(1 + \varepsilon)$, such that any value between $\min \text{dist}(p, q)$ and $\max \text{dist}(p, q)$ is $(1 + \varepsilon)$ approximated. For each power of $(1 + \varepsilon)$, we run Algorithm 1. Let γ be the largest value for which one of the instances of Algorithm 1 returns two points. The next larger estimate $\gamma \cdot (1 + \varepsilon)$ guarantees us no diameter of size $3(1 + \varepsilon)\gamma$, proving an approximation guarantee of at most $\frac{3(1+\varepsilon)\gamma}{\gamma} = 3(1 + \varepsilon)$ The memory usage of the algorithm consists of 4 points per instance of Algorithm 1 and $\log_{1+\varepsilon} \alpha = \frac{\ln \alpha}{\ln(1+\varepsilon)} \leq \frac{2}{\varepsilon} \ln \alpha$ instances. ◀

▶ Remark. To adapt this algorithm for windows where the maximum number of points are time dependent (e. g., the diameter of all points seen in the last hour) rather than the last N points, we can simply decouple the insertion procedure from the deletion routine. Whenever a point we currently keep in memory expires, we execute lines (4-10) and whenever a new point arrives, we call the INSERT procedure and line 12. Neither the invariants nor the proofs are affected in any way by this change.

4 The k-Center Problem

A 4-Approximation for Metric 2 Center

We run Algorithm 1 and show that, in the case of $k = 2$, it outputs a solution of cost at most 4 times the optimal solution. More precisely, let γ be the smallest estimate such that Algorithm 1 produces one point c with $\text{dist}(q, c) \leq 2\gamma$ for any point q in the current window. Further let a and b be the two points at distance greater than $\frac{\gamma}{1+\varepsilon}$ outputted by Algorithm 1 for the next smaller estimate. W.l.o.g let $\text{dist}(a, c) \geq \text{dist}(b, c)$. Then $\{a, c\}$ form a 4 approximation.

▶ **Theorem 6.** *Given a set of points A with aspect ratio α and a window of size S, there exists an algorithm computing a $4(1+\epsilon)$-approximate solution for the 2-center problem storing at most $8/\epsilon \cdot \ln \alpha$ points. The update time per point is $O(\varepsilon^{-1} \log \alpha)$.*

Proof. For c, the conditions of Invariant 1 apply, i.e. for any point p in our current window, we have $\text{dist}(p, c) \leq 2\gamma$. We now distinguish between two cases.

OPT $\geq \frac{\gamma}{2(1+\varepsilon)}$: We have $\text{dist}(p, \{a, c\}) \leq \text{dist}(p, c) \leq 2\gamma \leq 4 \cdot (1 + \varepsilon) \cdot \text{OPT}$.

OPT $< \frac{\gamma}{2(1+\varepsilon)}$: We first observe that a and b each fall into distinct clusters as their pairwise minimum distance is at least $\frac{\gamma}{1+\varepsilon}$. If a and c lie in distinct clusters, we have a 2-approximate solution, so we assume this not be the case. Then $\text{dist}(a, c) \leq 2 \cdot \text{OPT}$ and by construction, $\text{dist}(a, c) \geq \text{dist}(b, c)$. Then for any point p in the same cluster as b we have $\text{dist}(p, b) \leq 2 \cdot \text{OPT}$ and hence $\text{dist}(p, c) \leq \text{dist}(p, b) + \text{dist}(b, c) \leq 2 \cdot \text{OPT} + 2 \cdot \text{OPT} = 4 \cdot \text{OPT}$.

The proof of the space bound is analogous to that of Theorem 5. ◀

6-Approximation for Metric k-Center

A high level description of our algorithm is as follows, see also Algorithm 2 for pseudocode. We maintain a set A of at most $k+1$ *attraction points*. For each attraction point a, we maintain the newest point $R(a)$ within radius 2γ as a *representative*, i.e. $R(a) = \underset{p:\ \text{dist}(p,a)\leq 2\gamma}{\text{argmax}} TTL(R(a))$. When an attraction point expires, the representative point remains in memory. Call the set of representative points whose attraction points expired, the *orphaned representatives* O, and the set of representative points whose attraction points are still in the current window *active representatives* R. A new point p may become an attraction point if its distance is greater than 2γ to any point in A upon insertion. If the cardinality of A is greater than k, we retain the newest $k+1$ attraction points of A and all points with a greater TTL than the minimum TTL of A.

When asked to provide a clustering, we iterate through all estimates and either provide a counter example, or find a clustering which is then guaranteed to be a $6(1+\varepsilon)$-approximation. Our set of centers C first consists of an arbitrarily chosen point $p \in A \cup R \cup O$. Thereafter we greedily add any point point $q \in A \cup R \cup O$ with distance $\text{dist}(q, C) > 2\gamma$. If upon

termination $|C| > k$, we have a certificate for OPT $> \gamma$ and move to the next higher estimate. The smallest estimate with $|C| \leq k$ is then guaranteed to be a 6 approximation.

We start by giving the space bound.

Algorithm 2 Sliding Window Algorithm for $(\gamma, 6 \cdot \gamma)$-gap k-Center

```
1:  A, R, O ← ∅;
2:  for all element p of the stream do
3:      if q ∈ O expires then
4:          O ← O \ {q};
5:      if a ∈ A expires then
6:          DeleteAttraction(a);
7:      Insert(p);
8:  procedure DeleteAttraction(a)
9:      O ← O ∪ {R(a)};
10:     R ← R \ {R(a)};
11:     A ← A \ {a};
12: procedure Insert(p)
13:     D ← {a ∈ A | dist(p, a) ≤ 2 · γ};
14:     if D = ∅ then
15:         A ← A ∪ {p}
16:         R(p) ← p
17:         R ← R ∪ {R(p)}
18:         if |A| > k + 1 then
19:             a_old ← argmin_{a∈A} TTL(a);
20:             DeleteAttraction(a_old);
21:         if |A| > k then
22:             t ← min_{a∈A} TTL(a);
23:             for all q ∈ O do
24:                 if TTL(q) < t then
25:                     O ← O \ {q};
26:     else
27:         for all a ∈ D do
28:             Exchange R(a) with p in R;
```

▶ **Lemma 7.** *At any given time, the number of points kept in memory is bounded by at most* $3(k+1)$.

Proof. We number all attraction points we keep in memory via the sequence in which they arrived, i.e. a_1 is the first attraction point, a_2 the second, etc. Call this sequence S. Note that in this sequence a_1 also expires before a_2.

At any given time, we maintain at most $k+1$ attraction points A and $k+1$ active representative points R due to lines 18-20 and the subroutine DeleteAttraction (lines 8-11). What remains to be shown is that the number of orphaned representative points O also never exceeds $k+1$.

First, we show that $TTL(a_{i+k+1}) > TTL(R(a_i)) \geq TTL(a_i)$. We distinguish between two cases. If a_i expires, then a_{i+k+1} gets inserted after a_i exits the window, hence $TTL(a_{i+k+1}) > N+1+TTL(a_i)$ and $TTL(R(a_i))+N \leq TTL(a_i)$. Otherwise, a_i gets deleted via lines 18-20 in the exact same time step in which a_{i+k+1} got inserted, in which case the claim also holds.

Now consider any point of time and let j be the maximum index of any attraction point in S that has expired. By the above reasoning, any representative spawned by $a_{j-(k+1)}$ is no longer in memory, and the space bounds holds. ◀

▶ **Lemma 8.** *Let P be a set of points in a given window, $\gamma > 0$ an estimate of the clustering cost, $A \cup R \cup O$ the set of points we currently keep in memory with $|A| \leq k$. Then* $\max_{q \in P} \text{dist}(p, R \cup O) \leq 4\gamma$.

Proof. We note that for any attraction point a, the representative $R(a)$ has maximum TTL among all points with distance at most 2γ. When a point p arrives, it has distance at most 2γ to some attraction point (which may be identical to p if we create a new one). Hence, if $R(a)$ is still in memory, the claim holds for p.

We now argue that by executing lines 18-25, all points p with $\text{dist}(p, R \cup O) > 4\gamma$ have $TTL(p) < \min_{a \in A} TTL(a)$. If $TTL(p) > \min_{a \in A} TTL(a)$, then there exists an attraction point a' such that $\text{dist}(p, a') \leq 2\gamma$. Then we have $TTL(R(a')) \geq TTL(p) > \min_{a \in A} TTL(a)$ and $\text{dist}(p, R(a')) \leq 4\gamma$. Due to lines 24-25, $R(a')$ is never deleted until it expires. ◀

Combining these lemmas and using arguments analogous to those of the proof of Theorem 5, we have:

▶ **Theorem 9.** *Given a set of points P with aspect ratio α and a window size N, there exists an algorithm computing a $6(1+\epsilon)$-approximate solution for the metric k-center problem storing $6(k+1)\ln(\alpha)/\epsilon)$ points. The update time per point is $O(k^2\varepsilon^{-1}\log\alpha)$.*

Proof. Again define an exponential sequence to the base $(1+\varepsilon)$ and run Algorithm 2 in parallel for all powers of $(1+\varepsilon)$ as objective value estimates. The space bound then follows from Lemma 7.

For each estimate γ, we greedily compute a clustering of $A \cup R \cup O$ where the pairwise distance between centers is greater than 2γ. Now consider the smallest estimate γ' for which the greedy clustering requires at most k centers C.

We have $\max_{p \in A \cup R \cup O} \text{dist}(p, C) \leq 2\gamma'$. We further have for any point q in the current window $\max_{q \in P} \text{dist}(q, C) \leq \max_{q \in P} \text{dist}(q, R \cup O) + \max_{p \in A \cup R \cup O} \text{dist}(p, C) \leq 4\gamma' + 2\gamma' \leq 6\gamma'$ due to Lemma 8. Since we have $\text{OPT} > \frac{\gamma'}{1+\varepsilon}$, C is a $6(1+\varepsilon)$ approximation. ◀

5 Lower Bounds

Our lower bounds for the studied problems hold for the metric oracle distance model. Whenever we wish to know the distance between two points p, q, we have to store the points in their entirety in order to invoke the oracle. The fundamental assumption used in the proofs of this section is that the algorithm cannot create new points, unlike, for instance, in Euclidean spaces, where we can store projections, means and similar points. In particular, this implies that once a point is discarded by the algorithm, it cannot be recalled by any means at a later date. Without any assumptions as to how the points are encoded, we measure the space complexity of an algorithm via the number of stored points, rather than the number of bits. We do not consider the space required to store the distance oracle, the TTL of each point or any other information we might wish to store. A similar reasoning can be also found in the paper by Guha [15], where the author was able to derive a lower bound of $\Omega(k^2)$ points for any deterministic single-pass streaming algorithm approximating the cost of the optimal k-center clustering up to a factor $2 + 1/k$.

We first describe an adversarial input sequence for deterministic algorithms for the diameter problem and give a proof for randomized algorithms. With some modification, these ideas can be extended to the k-center problem, as well. We aim for a lower bound of $\Omega(\sqrt{N})$ points for deterministic algorithms. We divide the input into $\sqrt{N}$ buckets containing $\sqrt{N}$ points each. Unless the distances between two points are further specified, we will set these distances to 1. Denote the ith bucket by B_i and the jth point of bucket B_i by

$p_{i,j}$ with $i, j \in \{0, \ldots, \sqrt{N} - 1\}$. The points appear bucket by bucket, that is, $p_{i,j}$ is the $i \cdot \sqrt{N} + j + 1$th input point.

We assume that an algorithm always stores less than $\sqrt{N}$ points. Therefore, the algorithm must discard at least one point of bucket B_i before reading the first point of bucket B_{i+1}. Let f_i be such a discarded point in B_i. To any point from some bucket B_j, $j > i$, we then set the distance to f_i to be 2. By the same reason, there is at least one bucket without any stored points when the $N+1$st input point is read. Let B_t be this bucket. We now introduce the $N+1$st input point p that satisfies the distances $\text{dist}(p, p_{i,j}) = 1$ if $i > t$, $\text{dist}(p, p_{i,j}) = 3$ and $\text{dist}(p, p_{i,j}) = 2$ otherwise.

We proceed to insert copies of p until all points in buckets B_i with $i < t$ are expired. Therefore, there is no pair of points in memory with distance larger than 1. The algorithm can only output two points at distance 1 whereas the true diameter is 3.

▶ **Theorem 10.** *For windows of size N, any deterministic sliding window algorithm outputting a solution of cost greater than $\frac{1}{3}OPT$ for the distance oracle metric diameter problem with constant aspect ratio requires $\Omega(\sqrt{N})$ points.*

Recall that the algorithm by Chan and Sadjad [8] achieves a $2^{m+2} - 2 + \varepsilon$ approximation using $O(N^{1/(m+1)} \log \alpha)$ points, and it also falls under the same computational restrictions for the algorithms of this lower bound. Therefore, this lower bound cannot be strengthened by much, as their algorithm achieves a better approximation than 3 using roughly $N^{0.76}$ points by setting $m < \log 5/4$.

To utilize this instance for randomized algorithms, we require two modifications. First, we add additional points per bucket and uniformly choose f_i such that a randomized algorithm has little chance of retaining the correct point per bucket. Second, we use p (and its copies) to uniformly select a bucket B_t which the algorithm will have discarded with good probability.

▶ **Theorem 11.** *For windows of size N, any randomized sliding window algorithm outputting a solution of cost greater than $\frac{1}{3}OPT$ with probability greater than $\frac{1}{2}$ for the distance oracle metric diameter problem with constant aspect ratio requires $\Omega(\sqrt[3]{N})$ points.*

Proof. For ease of exposition, we will use a window of size $\Theta(N)$. The theorem then follows by rescaling N. We use $4N^{1/3}$ buckets consisting of $32N^{2/3}$ points each. In the following, any distance that is not further specified is set to be 1.

We iteratively replace one randomly chosen point from bucket B_i with f_i where f_i has distance 2 to any point from bucket B_j with $j > i$. At the end of the stream, we insert a point p with the following distances. First, choose a random bucket B_t and set $\text{dist}(p, f_t) = 3$ and $\text{dist}(p, q) = 2$, where $q \in B_t \setminus \{f_t\}$. Any point inserted after bucket B_t has distance 1 to p and any point inserted before B_t has distance 2. We then repeatedly add copies of p at total of N times.

To show that the distances still satisfy the triangle inequality, we first observe that only $\text{dist}(p, f_t)$ is neither 1 or 2 and thus requires special consideration. Here, we have $3 = \text{dist}(p, f_t) \leq \text{dist}(p, q) + \text{dist}(q, f_t) = 1 + 2$ for $q \in B_i$ with $i > t$, and $3 = \text{dist}(p, f_t) \leq \text{dist}(p, q) + \text{dist}(q, f_t) \leq 2 + 1$ for $q \in B_i$ with $i \leq t$.

At any given time, the algorithm has to output a pair of points whose distance is within a factor 3 of the diameter of the current window. Observe that if the algorithm did not store any of the replaced points $\{f_0, \ldots, f_{4N^{1/3}-1}\}$ and not any point of bucket B_t then the algorithm is not able to produce two points at distance greater than 1. Hence, by Yao's minimax principle, it is sufficient to bound the number of points used by any deterministic algorithm against the above input distribution.

We first bound the probability that the algorithm stores some point f_i. Call this event A. If we assume that the algorithm did not store any of the points $\{f_1, \dots, f_i\}$ it follows that the points in bucket B_{i+1} all have the same distance to the stored points. This implies that we can assume that the decision which points of bucket B_{i+1} will be kept is already fixed. The probability that f_i is one of these points is bounded by the hypergeometric distribution with population $32 \cdot N^{2/3}$, $N^{1/3}$ samples and 1 success in both population and sample: $\frac{\binom{32\cdot N^{2/3}-1}{N^{1/3}-1}\cdot\binom{1}{1}}{\binom{32\cdot N^{2/3}}{N^{1/3}}}$. Then the probability that no f_i is stored for any of the $4 \cdot N^{1/3}$ buckets can be lower bounded by

$$1 - \mathbb{P}[A] \geq \left(1 - \frac{\binom{32\cdot N^{2/3}-1}{N^{1/3}-1}\cdot\binom{1}{1}}{\binom{32\cdot N^{2/3}}{N^{1/3}}}\right)^{4\cdot N^{1/3}} = \left(1 - \frac{N^{1/3}}{32 \cdot N^{2/3}}\right)^{4\cdot N^{1/3}} \geq 1 - \frac{1}{8} = \frac{7}{8}.$$

Now we bound the probability that the algorithm retains any point from bucket t upon submission, which we call event B. Again, conditioned on the event that A does not hold ($\overline{A}$), the buckets from which the algorithm stores at least one point are fixed. The probability that B_t is among the stored buckets again follows a hypergeometric distribution with population $4 \cdot N^{1/3}$, $N^{1/3}$ samples and 1 success in both population and sample. Therefore $\mathbb{P}[B|\overline{A}] = \frac{\binom{4\cdot N^{1/3}-1}{N^{1/3}-1}\cdot\binom{1}{1}}{\binom{4\cdot N^{1/3}}{N^{1/3}}} = \frac{1}{4}$. Since one of the events A or B has to hold for the algorithm to output a solution with approximation factor greater than $\frac{1}{3}$, the probability that an algorithm storing less than $N^{1/3}$ points produces a solution with the desired approximation guarantee is at most $\mathbb{P}[A \cup B] \leq \mathbb{P}[A] + \mathbb{P}[B] = \mathbb{P}[A] + \mathbb{P}[B|A] \cdot \mathbb{P}[A] + \mathbb{P}[B|\overline{A}] \cdot \mathbb{P}[\overline{A}] \leq 2 \cdot \mathbb{P}[A] + \mathbb{P}[B|\overline{A}] \leq \frac{2}{8} + \frac{1}{4} = \frac{1}{2}$. ◀

We only briefly describe the k-center lower bound. The instance is also divided into sufficiently large buckets, from which the algorithm is forced to discard one point each. The main difference with the previous proof will be that the distances between all the points (except for a randomly chosen missing point f_t) are 2 and the distance from f_t to the more recent buckets is 4.

▶ **Theorem 12.** *For windows of size N, any randomized sliding window algorithm achieving an approximation factor less than 4 with probability greater than $\frac{1}{2}$ for the distance oracle metric 2 center problem with constant aspect ratio requires $\Omega(\sqrt[3]{N})$ points.*

References

1. Pankaj K. Agarwal, Jirí Matousek, and Subhash Suri. Farthest neighbors, maximum spanning trees and related problems in higher dimensions. *Comput. Geom.*, 1:189–201, 1991. doi:10.1016/0925-7721(92)90001-9.
2. Pankaj K. Agarwal and R. Sharathkumar. Streaming algorithms for extent problems in high dimensions. *Algorithmica*, 72(1):83–98, 2015. doi:10.1007/s00453-013-9846-4.
3. Brian Babcock, Mayur Datar, Rajeev Motwani, and Liadan O'Callaghan. Maintaining variance and k-medians over data stream windows. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 234–243, 2003. doi:10.1145/773153.773176.
4. Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering on sliding windows in polylogarithmic space. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 350–364, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.350.

5 Vladimir Braverman, Harry Lang, Keith Levin, and Morteza Monemizadeh. Clustering problems on sliding windows. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1374–1390, 2016. doi:10.1137/1.9781611974331.ch95.

6 Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131, 2010. doi:10.1137/090749281.

7 Timothy M. Chan and Vinayak Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. *Comput. Geom.*, 47(2):240–247, 2014. doi:10.1016/j.comgeo.2013.05.007.

8 Timothy M. Chan and Bashir S. Sadjad. Geometric optimization problems over sliding windows. *Int. J. Comput. Geometry Appl.*, 16(2-3):145–158, 2006. doi:10.1142/S0218195906001975.

9 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 626–635, 1997. doi:10.1145/258533.258657.

10 Moses Charikar, Liadan O'Callaghan, and Rina Panigrahy. Better streaming algorithms for clustering problems. In *Proc. of the 35th Annual ACM Symp. on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 30–39, 2003. doi:10.1145/780542.780548.

11 Michael S. Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 337–348, 2013. doi:10.1007/978-3-642-40450-4_29.

12 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002. doi:10.1137/S0097539701398363.

13 Joan Feigenbaum, Sampath Kannan, and Jian Zhang. Computing diameter in the streaming and sliding-window models. *Algorithmica*, 41(1):25–41, 2004. doi:10.1007/s00453-004-1105-2.

14 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985. doi:10.1016/0304-3975(85)90224-5.

15 Sudipto Guha. Tight results for clustering and summarizing data streams. In *Database Theory – ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23-25, 2009, Proceedings*, pages 268–275, 2009. doi:10.1145/1514894.1514926.

16 Dorit S. Hochbaum and David B. Shmoys. A unified approach to approximation algorithms for bottleneck problems. *J. ACM*, 33(3):533–550, 1986. doi:10.1145/5925.5933.

17 Piotr Indyk. Better algorithms for high-dimensional proximity problems via asymmetric embeddings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 12-14, 2003, Baltimore, Maryland, USA.*, pages 539–545, 2003.

18 Sang-Sub Kim and Hee-Kap Ahn. An improved data stream algorithm for clustering. *Comput. Geom.*, 48(9):635–645, 2015. doi:10.1016/j.comgeo.2015.06.003.

19 Richard Matthew McCutchen and Samir Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, August 25-27, 2008. Proceedings*, pages 165–178, 2008. doi:10.1007/978-3-540-85363-3_14.

20 Hamid Zarrabi-Zadeh. Core-preserving algorithms. In *Proc. of the 20th Annual Canadian Conf. on Computational Geometry, Montréal, Canada, August 13-15, 2008*, 2008.

21 Hamid Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica*, 60(1):46–59, 2011. doi:10.1007/s00453-010-9392-2.

Approximate Hamming Distance in a Stream

Raphaël Clifford[1] and Tatiana Starikovskaya[2]

1 **University of Bristol, Bristol, UK**
`raphael.clifford@bristol.ac.uk`
2 **University of Bristol, Bristol, UK**
`tat.starikovskaya@gmail.com`

Abstract

We consider the problem of computing a $(1+\varepsilon)$-approximation of the Hamming distance between a pattern of length n and successive substrings of a stream. We first look at the one-way randomised communication complexity of this problem. We show the following:

- If Alice and Bob both share the pattern and Alice has the first half of the stream and Bob the second half, then there is an $\mathcal{O}(\varepsilon^{-4}\log^2 n)$ bit randomised one-way communication protocol.
- If Alice has the pattern, Bob the first half of the stream and Charlie the second half, then there is an $\mathcal{O}(\varepsilon^{-2}\sqrt{n}\log n)$ bit randomised one-way communication protocol.

We then go on to develop small space streaming algorithms for $(1+\varepsilon)$-approximate Hamming distance which give worst case running time guarantees per arriving symbol.

- For binary input alphabets there is an $\mathcal{O}(\varepsilon^{-3}\sqrt{n}\log^2 n)$ space and $\mathcal{O}(\varepsilon^{-2}\log n)$ time streaming $(1+\varepsilon)$-approximate Hamming distance algorithm.
- For general input alphabets there is an $\mathcal{O}(\varepsilon^{-5}\sqrt{n}\log^4 n)$ space and $\mathcal{O}(\varepsilon^{-4}\log^3 n)$ time streaming $(1+\varepsilon)$-approximate Hamming distance algorithm.

1998 ACM Subject Classification F.2 Analysis of algorithms and problem complexity

Keywords and phrases Hamming distance, communication complexity, data stream model

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.20

1 Introduction

We study the complexity of one of the most basic problems in pattern matching, that of approximating the Hamming distance. Given a pattern P of length n the task is to output a $(1+\varepsilon)$-approximation of the Hamming distance between P and every n-length substring of a longer text. We provide the first efficient one-way randomised communication protocols as well as a new, fast and space efficient streaming algorithm for this problem.

The general task of efficiently computing the Hamming distances offline between a pattern and a text has been studied for many years. When the input is binary and the text has length proportional to that of the pattern, then all outputs can be computed exactly in $\mathcal{O}(n\log n)$ time by repeated application of the fast Fourier transform [14]. For larger alphabets, $\mathcal{O}(n\sqrt{n\log n})$ time solutions were first discovered in the 1980s [1, 22]. The fastest randomised algorithm for $(1+\varepsilon)$-approximate Hamming distance computation for large alphabets was due for many years to Karloff from 1993 [20] running in $\mathcal{O}(\varepsilon^{-2}n\log^2 n)$ time overall. In a breakthrough paper in 2015 a new algorithm was given improving the time complexity to $\mathcal{O}(\varepsilon^{-1}n\log^3 n\log\varepsilon^{-1})$ [21]. These fast methods all require linear space and up until this point no sublinear space solutions have been known.

The first basic question that arises is whether it is in fact possible to give a $(1+\varepsilon)$-approximation to the Hamming distance in a streaming setting while using only sublinear space. In order to explore this question we start our study by considering two natural

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 20; pp. 20:1–20:14


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

communication complexity problems which may also be of independent interest. Any lower bound for these communication problems will give a lower bound for the space usage of a corresponding streaming algorithm. This follows from a standard reduction where a space efficient streaming algorithm is converted into a communication protocol by taking a snapshot of memory after some symbol of the input has been read in and then sending this snapshot to the other player. On the other hand, the communication upper bounds we provide will set targets for space bounds for algorithms in the streaming setting.

Any streaming pattern matching algorithm using a pattern of length n can be reduced to repeated application of a streaming algorithm that runs on texts of length $2n$. This is done by splitting the stream into substreams of length $2n$ which overlap by n symbols. As a result we consider communication problems with these parameter settings for pattern and text length.

▶ **Problem 1.** *Consider a text T of length $2n$ and a pattern P of length n. Let Alice hold the information about the first half of the text and the whole of the pattern, and let Bob hold the information about the second half of the text and the whole of the pattern. Bob must output $(1+\varepsilon)$-approximations of the Hamming distance for each alignment of P and T.*

A lower bound for the communication complexity of this problem follows from a combination of the lower bound for the communication complexity of a windowed counting problem introduced by Datar et al. in 2002 [13] and the one-way communication complexity lower bound for approximating the Hamming distance between two n-bit strings from [18].

For the first part consider the following communication problem. Assume that there is a bit vector B of length $2n$. Let Alice hold the information about the first half of B, and let Bob hold the information about the second half of B. Bob must output $(1+\varepsilon)$-approximations of the number of set bits in each window of length n. Datar et al. showed that Alice will have to send to Bob $\Omega(\varepsilon^{-1}\log^2\varepsilon^{-1}n)$ bits of information. There is a straightforward reduction from this basic counting problem to Problem 1 which then gives us the same lower bound. We set $T=B$ and $P=00\ldots0$ and then a $(1+\varepsilon)$-approximation of the Hamming distance at an alignment i of P and T gives a $(1+\varepsilon)$-approximation of the number of set bits in the window $T[i,i+n-1]$. For the second part we use Theorem 4.1 from [18] which states that the one-way communication complexity of $(1+\varepsilon)$-approximate Hamming distance for two strings of length n is $\Omega(\varepsilon^{-2}\log n)$ for constant error probability. Combining these two lower bounds together we get a lower bound of $\Omega(\varepsilon^{-2}\log n+\varepsilon^{-1}\log^2\varepsilon^{-1}n)$ for the communication complexity of Problem 1.

Our first result is an efficient one-way communication protocol for Problem 1 whose complexity is only slightly higher than this lower bound. In our protocol Alice uses the fact that Bob knows the pattern as well to give an efficient encoding for parts of her half of the text which are at small Hamming distance from the pattern.

▶ **Theorem 1.** *Problem 1 has one-way randomised communication complexity $\mathcal{O}(\varepsilon^{-4}\log^2 n)$.*

As a model for streaming pattern matching, this communication upper bound requires that a copy of the pattern is available at all times. Our main interest is however in algorithms whose total space complexity is sublinear in the pattern size. In order to model this situation more accurately we now consider a stronger three party communication problem.

▶ **Problem 2.** *Assume that there is a text T of length $2n$ and a pattern P of length n. Let Alice hold the information about the pattern, let Bob hold the information about the first half of the text, and let Charlie hold the information about the second half of the text. Alice will send one message to Bob who will then send one message to Charlie. Charlie must output $(1+\varepsilon)$-approximations of the Hamming distance for each alignment of P and T.*

Somewhat surprisingly, we are still able to obtain a sublinear protocol for this new problem although the bound is higher than for the simpler Problem 1. The main technical elements of this communication protocol combine the newly introduced idea of approximate periods with succinct run-length encoded representations of the input.

▶ **Theorem 2.** *Problem 2 has one-way randomised communication complexity* $\mathcal{O}(\frac{\sqrt{n}}{\varepsilon^2}\log n)$.

Having investigated the communication complexity of $(1+\varepsilon)$-approximate Hamming distance we can now define the streaming $(1+\varepsilon)$-approximate Hamming distance problem.

▶ **Problem 3.** *Consider a pattern P of length n and a stream arriving one symbol at a time. We must output a $(1+\varepsilon)$-approximation to the Hamming distance between P and the latest n-length suffix of the stream as soon as a new symbol arrives. In this setting we cannot, for example, store a copy of the pattern or stream without accounting for it in our space usage.*

The upper bounds for the communication complexity of Problem 2 suggest space upper bounds we shall aim for in order to develop an optimal algorithm for the $(1+\varepsilon)$-approximate Hamming distance in the streaming setting. We make the first step towards this direction and show two randomised sublinear-space algorithms for the problem. We start by giving a solution for the case when both the pattern and the text are binary strings.

▶ **Theorem 3.** *When both P and T are binary, there is an algorithm for Problem 3 which uses $\mathcal{O}(\varepsilon^{-3}\sqrt{n}\log^2 n)$ bits of space and runs in $\mathcal{O}(\varepsilon^{-2}\log n)$ time per arriving symbol.*

The same bounds hold for alphabets of constant size σ as we can map each occurrence of the i^{th} symbol of the alphabet in the pattern or in the text to a binary string $0^{i-1}10^{\sigma-i}$, which will result in doubling the Hamming distance between the pattern and the text at each particular alignment.

For polynomial size alphabets our bounds are higher by a factor $\varepsilon^{-2}\log^2 n$ and our approach is based on the mapping idea of Karloff [20]. In that paper he showed that there exists $\Theta(\varepsilon^{-2}\log^2 n)$ mappings map_j of the alphabet onto $\{0,1\}$ such that an $(1+\varepsilon/3)$-approximation of the Hamming distance between P and T at a particular alignment can be given by a normalised average of the Hamming distances between $map_j(P)$ and $map_j(T)$ at this alignment. Moreover, Karloff showed that the mappings can be generated in $\mathcal{O}(\varepsilon^{-2}\log^3 n)$ space and $\mathcal{O}(\log n)$ time per symbol. For each pattern-text pair mapped on to a binary alphabet we then run the algorithm of Theorem 3 to find $(1+\varepsilon/3)$-approximations and finally obtain:

▶ **Theorem 4.** *There is an algorithm for Problem 3 which uses $\mathcal{O}(\varepsilon^{-5}\sqrt{n}\log^4 n)$ bits of space and runs in $\mathcal{O}(\varepsilon^{-4}\log^3 n)$ time per arriving symbol.*

Our solution has guaranteed worst case complexity per arriving symbol and uses roughly the square root of the space required by the known offline $(1+\varepsilon)$-approximate algorithms. A key technical innovation for our space reduction is the notion we introduce of a super-sketch. This a compact and efficiently updateable representation of consecutive text substrings which we require to be able to achieve sublinear space. For simplicity we will make the natural assumption throughout that $\varepsilon < 1/2$.

1.1 Related work and lower bounds

The one-way communication complexity of a number of variants of Hamming distance computation has been studied over the years. These includes $(1+\varepsilon)$-approximation [18],

the so called gap Hamming problem [9] and a bounded version known as k-mismatch [15]. However all this previous work has assumed that both Alice and Bob have strings of the same length and so need only give a single output. There has also been great interest in efficient streaming algorithms over the last 20 years, following the seminal work of [2]. In relation specifically to pattern matching problems, where space is not limited but where an output must be computed after every new symbol of the text arrives, the Hamming distance between the pattern and the latest suffix of the stream can be computed online in $O(\sqrt{n \log n})$ worst-case time per arriving symbol or $O(\sqrt{k} \log k + \log n)$ time for the k-mismatch version [11]. Both these methods however require $\Theta(n)$ space. Using the same approach, a number of other approximate pattern matching algorithms have also been transformed into efficient linear space online algorithms including [4, 3, 5, 8, 7, 6, 23]. In 2013 a small space streaming pattern matching algorithm was shown for parameterised matching [17] and in 2016 for the k-mismatch problem [10]. The latter k-mismatch paper is of particular relevance to our work. In [10] as a part of a space-efficient streaming algorithm for the k-mismatch problem, the authors presented a $(1+\varepsilon)$-approximate algorithm with space $\mathcal{O}(\varepsilon^{-2}k^2 \log^7 n)$ and running time $\mathcal{O}(\varepsilon^{-2} \log^5 n)$ per arriving symbol that returns a $(1+\varepsilon)$-approximation for all alignments of the pattern and text where the Hamming distance is at most k. The algorithm we give in this current paper can be seen a generalisation of this work, both removing the requirement for a prespecified threshold k and also using less space when $k \gtrsim n^{1/4}$.

2 Overview

In this section we give an overview of the main ideas needed for our results. We will make extensive use of sketching. Alon, Matias and Szegedy were first to show that sketching can be used to approximate frequency statistics of a stream with a particular emphasis on F_2 [2]. Later their sketching technique was generalized to allow approximation of $||x_1 - x_2||_p$ for two vectors x_1 and x_2 and any $p \in (0; 2]$ by Indyk et al. [16, 12]. We will use the sketches of Indyk et al. to show the communication complexity upper bounds. These sketches are based on p-stable distributions and have the advantage that they can be used even for large-size alphabets. For our streaming algorithm where we assume that the input alphabet is binary we will use simpler sketches based on the original technique of Alon et al.

2.1 Communication complexity

To show communication complexity bounds we will be using sketches based on p-stable distributions (see [16] and Sections 4.1 and 5.1 of [12]). Setting σ to be the alphabet size, a sketch of a string x is defined as a vector $sk(x)$ of length $\Theta(\varepsilon^{-2})$ such that

$$sk(x)[i] = \sum_j Y_{i,j} \cdot x[j]$$

where each $Y_{i,j}$ is drawn independently from a random stable distribution with parameter $p \leq \varepsilon / \log \sigma$. For two vectors x_1 and x_2 it can be shown that with constant probability the median of values $|sk(x_1)[i] - sk(x_2)[i]|$, appropriately scaled, is a $(1+\varepsilon)$-approximation of the Hamming distance. Importantly, variables $Y_{i,j}$ can be generated when we need them with the help of Nisan's pseudo-random generator, which requires only $\mathcal{O}(\log^2 n)$ random bits.

2.1.1 Problem 1 – both Alice and Bob know the pattern

The main idea of our communication complexity upper bound for Problem 1 is that if the Hamming distance between the text and the pattern at a particular alignment is (relatively) small, then Alice and Bob can use the pattern to describe the part of the text aligned with the pattern.

At each alignment the pattern can be divided into two parts – a prefix, aligned with Alice's half of the text, and a suffix, aligned with Bob's half of the text. Alice needs to transmit information that will help Bob approximate the Hamming distance between these different prefixes of the pattern and her half of the text. She does so by selecting a logarithmic number of prefixes of the pattern with Hamming distances $\Theta(\varepsilon^{-j})$ from the text. She then divides the part of the text aligned with each of these prefixes into blocks such that the mismatches are evenly spread across the blocks, and sends each block's starting position and sketch to Bob.

When Bob wants to compute the Hamming distance between a prefix P' of the pattern and the text and he knows that this Hamming distance is at least $\Theta(\varepsilon^{-(j-1)})$, he uses the prefix P_j with Hamming distance $\Theta(\varepsilon^{-j})$ and the sketches of associated text blocks. The part of Alice's text aligned with P' can be composed of several full blocks and at most one block suffix. Hamming distances between P' and the full blocks can be approximated with the help of the sketches. To approximate the Hamming distance between P' and the suffix of the block, Bob will substitute the suffix with the aligned part of P_j. As the number of mismatches between the suffix and P_j is small compared to $\Theta(\varepsilon^{-(j-1)})$, it will give a good approximation of the Hamming distance.

2.1.2 Problem 2 – only Alice knows the pattern

We start by reviewing some notation introduced in [10].

▶ **Definition 5.** The x-period of a string S of length n is the smallest integer $\ell > 1$ such that the Hamming distance between $S[1, n-\ell]$ and $S[\ell, n]$ is at most x.

▶ **Definition 6.** We define the ℓ-RLE encoding of S to be the ordered set of the run-length encodings of strings $S_i = S[i]S[\ell+i]S[2\ell+i]\dots S[\lfloor (n-i)/\ell \rfloor \cdot \ell + i]$, where $i \in [1, \ell]$. The size of the ℓ-RLE encoding is the total number of runs in the encodings of strings S_i.

▶ **Example 7.** Let $S = aabaabaabaabaabaabaac$. The 3-RLE encoding of S is: the run-length encoding $(a, 7)$ of $S_1 = aaaaaaa$, the run-length encoding $(a, 7)$ of $S_2 = aaaaaaa$, and the run-length encoding $(b, 6)(c, 1)$ of $S_3 = bbbbbbc$. The size of the encoding is $1 + 1 + 2 = 4$.

Note that ℓ-RLE encoding of S is deterministic and lossless. In [10] it was also shown that if ℓ is the x-period of a string S for some integer x, then the size of the ℓ-RLE encoding is $\mathcal{O}(\ell + x)$. Intuitively, this is because each new run in the encoding of S_i corresponds to a mismatch between $S[1, n-\ell]$ and $S[\ell, n]$, and therefore there can be at most $\ell + x$ runs.

We now explain the idea of the communication protocol for Problem 2. Let the block size $B = \sqrt{n}$ and the threshold $\tau = 2\varepsilon^{-1}\sqrt{n}$. Bob will compute a sketch for each B^{th} suffix of his half of the text and send it to Charlie. Consider a particular alignment of the pattern and of the text.

Case 1: Hamming distance is large. The pattern can be divided into three parts: a prefix of length at most $B - 1$, a middle part aligned with one of the n/B sketched suffixes of Bob's half of the text, and a suffix aligned with Charlie's half of the text. If the Hamming

distance at the alignment is larger than τ, then the prefix can be discarded as it will change the Hamming distance by at most $B = (\varepsilon/2) \cdot \tau$. The Hamming distance between the rest of the pattern and the text can be approximated easily. Charlie has received the sketch of the middle part of the pattern as well as the sketch of the suffix of Alice's half of the text which aligns with it. Charlie can combine the sketch from Alice's part of the text with the information he has about his half of the text and then compare this sketch to the pattern sketch as required.

Case 2: Hamming distance is small. The main challenge is therefore alignments where the Hamming distance is smaller than τ. If the $(2+\varepsilon)\tau$-period of the pattern is larger than B, then there are at most n/B such alignments. In this case, Bob can simply send the Hamming distances for all these alignments to Bob. If the period is at most B, then Bob will find the first alignment with small Hamming distance and will use the ℓ-RLE encoding of the pattern and the full list of mismatches to describe the text. Using this description Charlie will be able to fully recover the corresponding suffix of the text and to compute the Hamming distances for all remaining alignments. The only technicality is that Bob does not know Charlie's half of the text and thus will not be able to compute the Hamming distances between the whole pattern and the text. We elaborate on this in Section 3.2.

2.2 A small space streaming algorithm

In our small space streaming algorithm we will use simpler sketches which provide a $(1+\varepsilon)$-approximation to the Hamming distance between two binary strings of the same length B. The method is now folklore but is essentially an application of the technique of the Johnson-Lindenstrauss lemma [19]. To do this we create a random $\varepsilon^{-2} \times B$ matrix M whose entries are from $\{-1, 1\}$. The sketch of a string x of length B is then defined to be equal to Mx, and it is known that the appropriately scaled square of the L_2 norm of the difference of the sketches of two strings gives a $(1+\varepsilon)$-approximation of the Hamming distance between them. We will also be using M to define sketches of strings of length $\ell < B$. In this case, we simply append the strings with $(B-\ell)$ zeros and use the method describe above. The original analysis applies here as well. Finally, we will use M to define "super-sketches" of strings of length $n - B$. Assume that a string of length $n - B$ is divided into $n/B - 1$ non-overlapping blocks of size B. A super-sketch is then defined to be a linear combination of the sketches of the blocks. We elaborate more on sketches and super-sketches in Section 4.

Now we give a high-level overview of our algorithm. The algorithm starts by preprocessing the pattern P. It computes and stores a super-sketch of each $(n - B)$-length substring of P. The algorithm then processes the text in non-overlapping blocks of length B, computing a sketch for each block. The blocks' sketches can be maintained efficiently as we need to maintain only one sketch at a time. The algorithm also maintains a super-sketch of the last $n/B - 1$ blocks. To compute an approximation of the Hamming distance at a particular alignment, the algorithm divides the pattern into three parts: a prefix of length $(B - i)$, a middle part of length $(n - B)$, and a suffix of length i, where the middle part is aligned with a block border (see Figure 1).

The algorithm then starts by computing the $(1 + \varepsilon)$-approximation of the Hamming distance between the middle part and the text with the help of the super-sketches. If the Hamming distance is large, the algorithm can simply discard the prefix and suffix parts. Otherwise, the algorithm also needs to approximate the Hamming distance between the prefix or the suffix of the pattern and the text. To approximate the Hamming distance between the prefix of the pattern and the text the idea is to use the information Alice transfers to

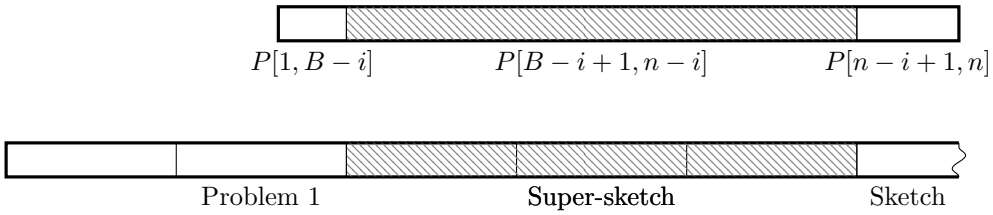


Figure 1 To estimate the Hamming distances at a particular position the algorithm uses a data structure containing the information Alice transfers to Bob in our solution for Problem 1 for the prefix $P[1, B-i]$, a super-sketch for the middle part $P[B-i+1, n-i]$, and a sketch for the suffix $P[n-i+1, n]$.

Bob in our solution for Problem 1. For the suffix, the algorithm will use the sketch of the part of the block between its start and the current alignment.

3 Communication complexity

In this section we show upper bounds for communication complexities of Problems 1 and 2.

3.1 Problem 1

We start by showing an upper bound for the communication complexity of Problem 1. Remember that in this problem we have two players, Alice and Bob. Alice knows the first half of the text T and the pattern P, and Bob knows the second half of the text T and the pattern P. We will show that the communication complexity of this problem is $\mathcal{O}(\varepsilon^{-4}\log^2 n)$.

Let us first explain what Alice sends to Bob. For simplicity, we denote $k = 6/\varepsilon$. First, Alice selects $q = \lfloor \log_k n \rfloor$ positions $n \geq i_1 \geq i_2 \geq \ldots \geq i_q \geq 1$ such that the Hamming distance between $T[i_j, n]$ and the prefix $P[1, n-i_j+1]$ is at most k^{j+1}. She does this in turn starting from $j = 1$ and selecting the leftmost possible position for each j. Alice then sends to Bob $\mathcal{O}(k^2 \cdot \varepsilon^{-2}\log n) = \mathcal{O}(\varepsilon^{-4}\log n)$ bits of information for each j. She starts by dividing $T[i_j, n]$ into k^2 blocks such that the Hamming distance between each block and the corresponding substring of the pattern is at most k^{j-1}. If $n \geq b_1 > b_2 > \ldots > b_{k^2} = i_j$ are the borders of the blocks, she sends Bob $b_1, b_2, \ldots, b_{k^2} = i_j$ and the $(1+\varepsilon/6)$-approximate sketches of $T[b_\ell, n]$ for all $\ell \in [1, k^2]$. In total, Alice sends to Bob $\mathcal{O}(\varepsilon^{-4}\log^2 n)$ bits of information.

To see how Bob can use this information, consider a particular position i. At this position $P[1, n-i+1]$ is aligned with Alice's half of the text, whereas $P[n-i+2, n]$ is aligned with Bob's half of the text. As Bob knows the pattern, he can compute the exact Hamming distance between $P[n-i+2, n]$ and his half of the text with no additional information. We now go on to explain how he can estimate the Hamming distance h between $P[1, n-i+1]$ and Alice's half of the text.

Bob starts by locating the position i_j that is closest to i from the left, and the block $T[b_{\ell+1}, b_\ell]$ of $T[i_j, n]$ containing i (see Figure 2). The border b_ℓ divides the pattern into two parts, P_1 and P_2. Let h_1 be the Hamming distance between P_1 and the text, and h_2 be the Hamming distance between P_2 and the text, $h_1 + h_2 = h$. To find a $(1+\varepsilon)$-approximation h_2' of h_2, Bob uses the sketch of $T[b_\ell, n]$. He cannot use sketches to estimate h_1 as P_1 is not aligned with the block $T[b_{\ell+1}, b_\ell]$, but he knows that there are only a few mismatches between $T[b_\ell, b_{\ell+1}]$ and the pattern aligned at the position i_j. So he estimates h_1 by computing

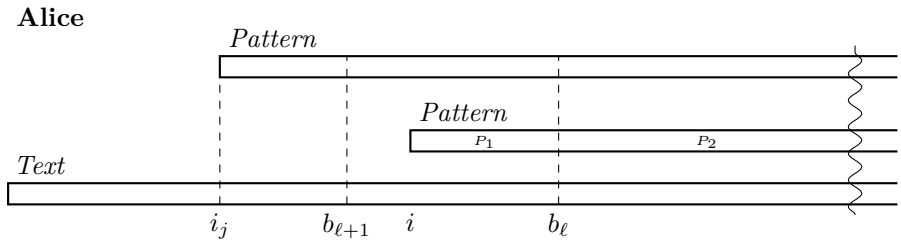


Figure 2 Figure shows Alice's half of the text and the rightmost position $i_j < i$. Dashed lines show block borders for $T[i_j, n]$. Borders $b_{\ell+1}$ and b_ℓ are the closest to i from the left and from the right respectively. The border b_ℓ divides the pattern into two parts P_1 and P_2. To estimate the Hamming distance h_1 between P_1 and T, Bob uses the pattern aligned at i_j. To estimate the Hamming distance h_2 between P_2 and T, he uses the sketch of $T[b_\ell, n]$.

the Hamming distance h'_1 between P_1 and the pattern aligned at the position i_j. The next lemma shows that Bob can output $h' = (h'_1 + h'_2)/(1 - \varepsilon/3)$ as a $(1+\varepsilon)$-approximation of h.

▶ **Lemma 8.** *h' is a $(1+\varepsilon)$-approximation of h.*

Proof. Remember that h'_1 is the Hamming distance between P_1 and the pattern aligned at the position i_j, and h_1 is the Hamming distance between P_1 and the text. The Hamming distance between the pattern aligned at the position i_j and $T[b_{\ell+1}, b_\ell]$ is at most k^{j-1}. Therefore,

$$h_1 - k^{j-1} \le h'_1 \le h_1 + k^{j-1}$$

On the other hand, h'_2 is a $(1+\varepsilon/6)$-approximation of h_2. Hence,

$$h_1 + h_2 - k^{j-1} \le h'_1 + h'_2 \le h_1 + k^{j-1} + (1+\varepsilon/6) \cdot h_2 \,.$$

We now substitute $h = h_1 + h_2$ and estimate $h_2 \le h$ to obtain

$$h - k^{j-1} \le h'_1 + h'_2 \le (1+\varepsilon/6) \cdot h + k^{j-1} \,.$$

Finally, by our choice of i_j we have $h \ge k^{j+1}$, and therefore

$$(1-\varepsilon/3) \cdot h \le (1-\varepsilon/6) \cdot h \le h'_1 + h'_2 \le (1+\varepsilon/3) \cdot h \,.$$

Dividing all parts of this inequality by $(1-\varepsilon/3)$, we obtain

$$h \le h' = (h'_1 + h'_2)/(1-\varepsilon/3) \le \frac{1+\varepsilon/3}{1-\varepsilon/3}\, h \le (1+\varepsilon) \cdot h \,. \qquad \blacktriangleleft$$

3.2 Problem 2

In this section we show an upper bound for the communication complexity of Problem 2. Remember that in this problem we have three players, Bob, Charlie, and Alice. Bob knows the first half of the text T, Charlie knows the second half of the text T, and Alice knows the pattern P. We will show that the communication complexity of this problem is $\mathcal{O}(\varepsilon^{-2}\sqrt{n}\log n)$.

Let the block size $B = \sqrt{n}$ and the threshold $\tau = 2\varepsilon^{-1}\sqrt{n}$. We start by explaining what the players send to each other. Alice sends to Bob the following information:

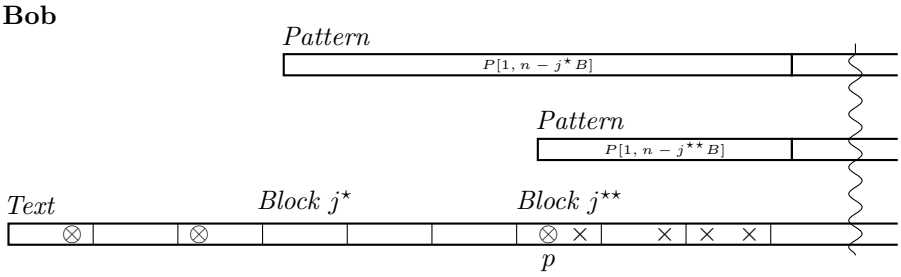


Figure 3 The figure shows Bob's half of the text. Crosses show alignments where the Hamming distance is at most τ. $P[1, n - j^\star B]$ is the longest prefix with $\tau(2+\varepsilon)$-period smaller than B. Block $j^{\star\star} \geq j^\star$ is the first block containing a cross. Bob sends to Charlie sketches of text suffixes starting at blocks' borders, locations of all encircled crosses, and the last block.

1. $(1+\varepsilon/2)$-approximate sketches of suffixes $P[i, n]$ for all $i \in [1, B]$ (Charlie will use them to estimate large Hamming distances);
2. $(1+\varepsilon/2)$-approximate sketches of prefixes $P[1, n - jB]$ for all $j \in [1, n/B]$ (Bob will use them to find alignments with small Hamming distances);
3. The ℓ-RLE encoding of the longest prefix $P[1, n - j^\star B]$ with $(2+\varepsilon)\tau$-period ℓ smaller than B (Bob will use it to describe the text).

Overall Alice sends $\mathcal{O}((n/B + B) \cdot \varepsilon^{-2} \log n + ((2+\varepsilon)\tau + B) \cdot \log n) = \mathcal{O}(\varepsilon^{-2}\sqrt{n} \log n)$ bits of information.

Bob starts by forwarding the information he received from Alice to Charlie. Bob also sends him $(1+\varepsilon/2)$-approximate sketches of all suffixes $T[jB, n]$. Next, for each $j < j^\star$ Bob uses the sketch of $P[1, n - jB]$ to find $(1+\varepsilon/2)$-approximations of Hamming distances in a block j. (Remember that Bob knows $T[1, n]$ and can compute a sketch for any its substring.) If the approximate value of the Hamming distance for some alignment is smaller than $(1+\varepsilon/2)\tau$, he sends it to Charlie. Note that there is at most one such alignment in a block. Indeed, if we have two such alignments in the block, then the Hamming distance between the patterns at these alignments is at most $(2+\varepsilon)\tau$, which is a contradiction with the $(2+\varepsilon)\tau$-period being larger than B. Moreover, Bob will not miss any alignment with the Hamming distance smaller than τ.

After that, Bob decodes $P[1, n - j^\star B]$ from its ℓ-RLE encoding and computes the Hamming distances between $P[1, n - jB]$ and the text for all alignments in blocks $j \geq j^\star$ precisely. He finds the first block $j^{\star\star} \geq j^\star$ where there is an alignment of $P[1, n - j^{\star\star} B]$ with the Hamming distance at most τ. Bob sends Charlie the starting position p of this alignment and the positions of the mismatches. Finally, he sends Charlie all bits of the last block of his half of the text. Overall, Bob sends to Charlie $\mathcal{O}(\varepsilon^{-2}\sqrt{n} \log n + (\varepsilon^{-2} \log n + \log n) \cdot (n/B) + \tau) = \mathcal{O}(\varepsilon^{-2}\sqrt{n} \log n)$ bits of information.

We now explain how Charlie computes the Hamming distances. If the Hamming distance at a particular alignment i in a block $j < j^{\star\star}$ is smaller than τ, then Charlie already knows its approximate value. If it is bigger than τ, then Charlie computes its approximation using the sketch of the longest suffix $P[jB - i, n]$ of P aligned with a block border, the sketch of $T[(j+1)B, n]$, and $T[n+1, 2n]$. Let h be the Hamming distance between the text and the pattern at the alignment i and let h' be the Hamming distance between $T[(j+1)B, i+n-1]$ and $P[jB - i, n]$.

▶ **Lemma 9.** $h \leq h' + B \leq (1+\varepsilon) \cdot h$.

Proof. The left inequality is trivial. To prove the right one, remember that $\tau \leq h$, which implies $B = (\varepsilon/2)\tau \leq (\varepsilon/2) \cdot h$. ◀

We now go on to the remaining blocks. The Hamming distances at alignments $i < p$ in the block $j^{\star\star}$ are bigger than τ and Charlie can find their approximation in the way described above. Charlie then decodes $P[1, n - j^{\star}B]$ and recovers $T[p, n]$ by fixing the at most $\sqrt{n}$ mismatches between $P[1, n - j^{\star}B]$ and $T[p, p + n - j^{\star\star}B + 1]$ and appending the last $p - (j^{\star\star} - 1)B$ symbols of T (Remember that Charlie knows all symbols of the last block of $T[1, n]$). Using $T[p, n]$, $T[n + 1, 2n]$, and the sketch of P, he can approximate the Hamming distances for all alignments to the right of p.

4 Streaming algorithm

We now show a streaming algorithm for Problem 3. In this problem we are asked to output a $(1 + \varepsilon)$-approximation of the Hamming distance between the pattern and the text at each alignment, and we do not assume that we store a copy of the pattern or of the text. For $\varepsilon < 1/2$, the algorithm uses $\mathcal{O}(\varepsilon^{-3}\sqrt{n}\log^{1.5} n)$ bits of space and its running time is $\mathcal{O}(\varepsilon^{-2}\log n)$ per arriving symbol. For simplicity, we will set $k = 1/\varepsilon > 2$ for the rest of this section.

Let $B = k\sqrt{n}$. The algorithm starts by selecting a $9k^2 \times B$ matrix M and a vector $(\sigma_1, \sigma_2, \ldots, \sigma_{n/B-1})$ of i.u.d. ± 1 random variables. The algorithm then preprocesses the pattern P. It remembers the first B symbols of P, as well as a super-sketch of each $(n - B)$-length substring of P. To compute the super-sketches the algorithm divides a substring into $(n/B - 1)$ blocks of length B, computes their sketches using M as described in Section 2, and then sums the sketches multiplying them by σ_i. The algorithm also computes sketches of the last B suffixes of P. The sketch of a suffix $P[n - i + 1, n]$ is defined to be equal to $M \cdot S_i$, where $S_i = P[n - i + 1, n]\, 0^{B-i}$. Finally, for each $i \in [1, B]$ and for each $j \in [0, \log_{1+\varepsilon} n]$ it stores the maximal length of pattern's prefix such that the Hamming distance between this prefix aligned at position i and the pattern is at most $(1 + \varepsilon)^j$, which takes $\mathcal{O}(\varepsilon^{-1}B\log^2 n)$ bits since $\log_{1+\varepsilon} n = \mathcal{O}(\varepsilon^{-1}\log n)$.

4.1 Text processing

The algorithm processes the text in non-overlapping blocks of length B. For each of the last n/B blocks the algorithm maintains its sketch and a data structure containing the information Alice transfers to Bob in our solution for Problem 1.

Let us start by explaining how the algorithm maintains the sketches. At the starting index of each block it initialises the block's sketch with a zero vector of length $9k^2$. When the j^{th} symbol of the block arrives, the algorithm adds the product of the j^{th} column of M and the symbol to the sketch in $\mathcal{O}(9k^2)$ time. While reading the block the algorithm also computes the super-sketch of the $(n - B)$-length substring consisting of the $n/B - 1$ most recent blocks. Recall that the super-sketch is defined to be equal to the sum of the blocks' sketches multiplied by the variables σ_i. The total time needed for computing the sum is $\mathcal{O}(9k^2 n/B)$. The algorithm de-amortises this time over the block executing $\Omega(9k^2 n/B^2)$ steps per arriving symbol.

For each block the algorithm maintains a data structure containing the information Alice transmits to Bob in our solution for Problem 1. The algorithm starts computing the data structure when it has received the entire block. It then computes the Hamming distance between prefixes $P[1], P[1, 2], \ldots, P[1, B]$ as being aligned at the right border of the block

and the block by running the fast Fourier transform algorithm on $P[1, B]$ and the block appended with B zeros, which takes $\mathcal{O}(B)$ space and $\mathcal{O}(B \log B)$ time in total [14]. The algorithm then finds $i_1, i_2, \ldots, i_q$, where $q = \lceil \log_k B \rceil$ as defined in Problem 1 and for each i_j it computes the borders and the sketches of the blocks, where the sketches are defined with the help of the matrix M. Remember that the algorithm stores the block and the first B symbols of the pattern, so this could be done in a naive way, using symbol-by-symbol comparison. Finally, the algorithm builds binary search trees on $i_1, i_2, \ldots, i_q$ and the block borders for each i_j to allow fast access to the information. The total construction time of the data structure is $\mathcal{O}((B + k^2) \cdot \log n)$. Note that the data structure will only be used $n/B - 1 \geq 2$ blocks later, so we can de-amortise the construction time over the succeeding block executing $\Omega((1 + k^2/B) \cdot \log n)$ steps of the construction process per symbol. The data structure occupies $\mathcal{O}(k^4 \log^2 n)$ bits of space.

4.2 Hamming distance

To compute the Hamming distance at an alignment i, the algorithm divides the pattern into three parts: a prefix of length $(B - i)$, a middle part of length $(n - B)$, and a suffix of length i, where the middle part is aligned with a block border. The algorithm then starts by computing the square N of the norm of the difference between the super-sketches of the middle part and the corresponding text substring. Both super-sketches are already known as the middle part is an $(n - B)$-length substring of the pattern and we store its super-sketch explicitly, while the super-sketch of the text substring was computed at the end of the preceding block. As both sketches have length $9k^2$, it takes $\mathcal{O}(9k^2)$ time. Next, the algorithm computes the Hamming distance H_s between the sketch of the suffix of the pattern and the part of the text block seen so far. This again takes $\mathcal{O}(9k^2)$ time. Finally, the algorithm computes an approximation H_p of the Hamming distance between the prefix and the text as described in Problem 1. With the help of the binary search trees, $i_j, b_{\ell+1}$ and b_ℓ can be found in $\mathcal{O}(\log \log n + \log \log k^2)$ time. Recall that b_ℓ divides the prefix into two parts. The Hamming distance between the second part of the prefix and the text can be approximated in $\mathcal{O}(9k^2)$ time with the help of the sketches as in Problem 1, but it is not possible to use symbol-by-symbol comparison for the first part as this would take too much time. Instead, the algorithm does binary search on the prefixes' lengths it calculated during the preprocessing step which allows him to find $(1 + \varepsilon)$-approximation of the Hamming distance in $\mathcal{O}(\log \log_{1+\varepsilon} n)$ time. It then outputs $H_p + H_m + H_s$, where $H_m = \varepsilon^2 N / 9(1 - \varepsilon/3)$.

4.3 Analysis

The running time of the algorithm is $\mathcal{O}(\varepsilon^{-2} \log n)$ per arriving symbol. The space used is $\mathcal{O}(\varepsilon^{-3} \sqrt{n} \log^2 n)$ bits. We now need to show that $H_p + H_m + H_s$ is a $(1 + \varepsilon)$-approximation of the Hamming distance with constant probability. It suffices to show that H_m is a $(1 + \varepsilon)$-approximation of the Hamming distance between the middle part of the pattern and the text. Consider two binary strings t and p of length $(n - B)$. Let sk_t and sk_p be their super-sketches of length $9k^2$ calculated with the help of M and σ_i and let $N = \|sk_t - sk_p\|_2^2$ and $\tilde{H} = \tilde{\varepsilon}^2 N$, where $\tilde{\varepsilon} = \varepsilon/3$. We will show that $\tilde{H}$ is a good approximation of the Hamming distance between t and p. Recall that t and p are binary, and therefore the Hamming distance between them is equal to $\|t - p\|_2^2$.

▶ **Lemma 10.** *With constant probability* $(1 - \tilde{\varepsilon}) \cdot \|t - p\|_2^2 \leq \tilde{H} \leq (1 + \tilde{\varepsilon}) \cdot \|t - p\|_2^2$.

Proof. Let t_i and p_i, $i \in [1, n/B-1]$, be the blocks of t and p of length B. We have

$$\mathbb{E}\left[\tilde{H}\right] = \tilde{\varepsilon}^2 \cdot \mathbb{E}\left[\left\|\sum_i \sigma_i M \cdot (t_i - p_i)\right\|_2^2\right] = \tilde{\varepsilon}^2 \sum_j \mathbb{E}\left[\left(\sum_i \sigma_i M_j \cdot (t_i - p_i)\right)^2\right]$$

where M_j is the j^{th} row of M. As all rows of M are identically distributed, we have $\mathbb{E}\left[(\sum_i \sigma_i M_j \cdot (t_i - p_i))^2\right] = \mathbb{E}\left[(\sum_i \sigma_i M_1 \cdot (t_i - p_i))^2\right]$ for all j, which is equal to $\|t-p\|_2^2$ as if at least one of the inequalities $i_1 = i_2$ or $j_1 = j_2$ does not hold, then the variables $\sigma_{i_1} M_1[j_1]$ and $\sigma_{i_2} M_1[j_2]$ are independent and the expectation of $\sigma_{i_1}\sigma_{i_2} M_1[j_1] M_1[j_2]$ is equal to zero, and otherwise it is equal to one. So finally we have $\mathbb{E}\left[\tilde{H}\right] = \|t-p\|_2^2$.

We now compute the variance of H. We again use the fact that the rows of M are independent and identically distributed.

$$\mathrm{Var}\left[\tilde{H}\right] = \tilde{\varepsilon}^2 \cdot \mathrm{Var}\left[\left(\sum_i \sigma_i M_1 \cdot (t_i - p_i)\right)^2\right] \leq \tilde{\varepsilon}^2 \cdot \mathbb{E}\left[\left(\sum_i \sigma_i M_1 \cdot (t_i - p_i)\right)^4\right].$$

By Khintchine's inequality there exists a universal constant $c > 0$ such that

$$\mathrm{Var}\left[\tilde{H}\right] \leq c\,\tilde{\varepsilon}^2 \cdot \mathbb{E}\left[\left(\sum_i \sigma_i M_1 \cdot (t_i - p_i)\right)^2\right]^2 \leq c\,\tilde{\varepsilon}^2 \cdot \|t-p\|_2^4 .$$

The claim then follows by Chebyshev's inequality. ◀

Let now $H = \varepsilon^2 N/9(1-\varepsilon/3) = \tilde{H}/(1-\varepsilon/3)$. The probability H is a $(1+\varepsilon)$-approximation of the Hamming distance between t and p is at least the probability $\tilde{H}$ is in $[(1-\varepsilon/3)\cdot \|t-p\|_2^2, (1-\varepsilon/3)(1+\varepsilon)\cdot\|t-p\|_2^2]$, which in turn can be estimated from below as

$$\Pr\left[\tilde{H} \in [(1-\varepsilon/3)\cdot\|t-p\|_2^2, (1+\varepsilon/3)\cdot\|t-p\|_2^2]\right] \geq 1 - 1/c \text{ (Lemma 10.)}$$

To justify the last transition note that $(1-\varepsilon/3)(1+\varepsilon) \geq (1+\varepsilon/3)$ for all $\varepsilon < 1$. From above it follows that with constant probabilities H_p, H_m, and H_s are $(1+\varepsilon)$-approximations of the Hamming distances for the prefix, the middle part, and the suffix of the pattern respectively. We note that the probabilities can be made arbitrarily small by Chebyshev's inequality if we run a constant number of independent instances of the algorithm in parallel and output the sum of the medians of the values H_p, H_m, H_s. Correctness of the algorithm follows by the union bound.

Acknowledgements. We are grateful to T.S. Jayram and Paul Beame for helpful and inspiring conversations about the problems in this paper and to Ely Porat for introducing the original streaming pattern matching problem to us and for explaining how to solve Problem 3 in $\mathcal{O}(n^{2/3}poly(1/\varepsilon)))$ space. We were also informed that Ely Porat had independently developed a solution that uses $\mathcal{O}(\sqrt{n}/\varepsilon^2)$ space and for each alignment with Hamming distance H outputs some integer in the interval $[(1-\varepsilon)\cdot H - 1/2\sqrt{n}, (1+\varepsilon)\cdot H + 1/2\sqrt{n}]$ [24].

References

1 Karl Abrahamson. Generalized string matching. *SIAM Journal on Computing*, 16(6):1039–1051, 1987.

2 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC'00: Proc. 28th Annual ACM Symp. Theory of Computing*, pages 20–29. ACM, 1996.

3 Amihood Amir, Yonatan Aumann, Gary Benson, Avivit Levy, Ohad Lipsky, Ely Porat, Steven Skiena, and Uzi Vishne. Pattern matching with address errors: Rearrangement distances. *Journal of Computer System Sciences*, 75(6):359–370, 2009.

4 Amihood Amir, Yonatan Aumann, Oren Kapah, Avivit Levy, and Ely Porat. Approximate string matching with address bit errors. In *CPM'08: Proc. 19th Annual Symp. on Combinatorial Pattern Matching*, pages 118–129, 2008.

5 Amihood Amir, Yonatan Aumann, Moshe Lewenstein, and Ely Porat. Function matching. *SIAM Journal on Computing*, 35(5):1007–1022, 2006.

6 Amihood Amir, Richard Cole, Ramesh Hariharan, Moshe Lewenstein, and Ely Porat. Overlap matching. *Information and Computation*, 181(1):57–74, 2003.

7 Amihood Amir, Estrella Eisenberg, and Ely Porat. Swap and mismatch edit distance. *Algorithmica*, 45(1):109–120, 2006.

8 Amihood Amir, Martin Farach, and S. Muthu Muthukrishnan. Alphabet dependence in parameterized matching. *Information Processing Letters*, 49(3):111–115, 1994.

9 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. *SIAM Journal on Computing*, 41(5):1299–1317, 2012.

10 Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana A. Starikovskaya. The k-mismatch problem revisited. In *SODA'16: Proc. 27th ACM-SIAM Symp. on Discrete Algorithms*, pages 2039–2052, 2016.

11 Raphaël Clifford and Benjamin Sach. Pseudo-realtime pattern matching: Closing the gap. In *CPM'10: Proc. 21st Annual Symp. on Combinatorial Pattern Matching*, pages 101–111, 2010.

12 Graham Cormode, Mayur Datar, Piotr Indyk, and S. Muthukrishnan. Comparing data streams using Hamming norms (how to zero in). *IEEE Trans. on Knowl. and Data Eng.*, 15(3):529–540, 2003.

13 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002.

14 M. Fischer and M. Paterson. String matching and other products. In *Proc. 7th SIAM-AMS Complexity of Comp.*, pages 113–125, 1974.

15 Wei Huang, Yaoyun Shi, Shengyu Zhang, and Yufan Zhu. The communication complexity of the Hamming distance problem. *Information Processing Letters*, 99(4):149–153, 2006.

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

17 Markus Jalsenius, Benny Porat, and Benjamin Sach. Parameterized matching in the streaming model. In *STACS'13: Proc. 30th Annual Symp. on Theoretical Aspects of Computer Science*, pages 400–411, 2013. `arXiv:1109.5269`.

18 Thathachar S. Jayram and David P. Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms (TALG)*, 9(3):26, 2013.

19 William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Proc. of the Conference in Modern Analysis and Probability*, volume 26 of *Contemporary Mathematics*, pages 189–206. American Mathematical Society, 1984.

20 H. Karloff. Fast algorithms for approximately counting mismatches. *Information Processing Letters*, 48(2):53–60, 1993.

21 Tsvi Kopelowitz and Ely Porat. Breaking the variance: Approximating the Hamming distance in $1/\epsilon$ time per alignment. In *FOCS'15: Proc. 56th Annual Symp. Foundations of Computer Science*, pages 601–613, 2015.

22 S. Rao Kosaraju. Efficient string matching. Manuscript, 1987.

23 Gad M. Landau and Uzi Vishkin. Fast string matching with k differences. *Journal of Computer System Sciences*, 37(1):63–78, 1988.

24 Ely Porat. Personal communication, 2016.

Price of Competition and Dueling Games*

Sina Dehghani[1], MohammadTaghi Hajiaghayi[2], Hamid Mahini[3], and Saeed Seddighin[4]

1 **University of Maryland, College Park, USA**
dehghani@cs.umd.edu
2 **University of Maryland, College Park, USA**
hajiagha@cs.umd.edu
3 **University of Maryland, College Park, USA**
mahini@cs.umd.edu
4 **University of Maryland, College Park, USA**
saeedrez@cs.umd.edu

Abstract

We study competition in a general framework introduced by Immorlica, Kalai, Lucier, Moitra, Postlewaite, and Tennenholtz and answer their main open question. Immorlica et al. considered classic optimization problems in terms of competition and introduced a general class of games called dueling games. They model this competition as a zero-sum game, where two players are competing for a user's satisfaction. In their main and most natural game, the ranking duel, a user requests a webpage by submitting a query and players output an ordering over all possible webpages based on the submitted query. The user tends to choose the ordering which displays her requested webpage in a higher rank. The goal of both players is to maximize the probability that her ordering beats that of her opponent and gets the user's attention. Immorlica et al. show this game directs both players to provide suboptimal search results. However, they leave the following as their main open question: "does competition between algorithms improve or degrade expected performance?" (see the introduction for more quotes) In this paper, we resolve this question for the ranking duel and a more general class of dueling games.

More precisely, we study the quality of orderings in a competition between two players. This game is a zero-sum game, and thus any Nash equilibrium of the game can be described by minimax strategies. Let the value of the user for an ordering be a function of the position of her requested item in the corresponding ordering, and the social welfare for an ordering be the expected value of the corresponding ordering for the user. We propose the price of competition which is the ratio of the social welfare for the worst minimax strategy to the social welfare obtained by a social planner. Finding the price of competition is another approach to obtain structural results of Nash equilibria. We use this criterion for analyzing the quality of orderings in the ranking duel. Although Immorlica et al. show that the competition leads to suboptimal strategies, we prove the quality of minimax results is surprisingly close to that of the optimum solution. In particular, via a novel factor-revealing LP for computing price of anarchy, we prove if the value of the user for an ordering is a linear function of its position, then the price of competition is at least 0.612 and bounded above by 0.833. Moreover we consider the cost minimization version of the problem. We prove, the social cost of the worst minimax strategy is at most 3 times the optimal social cost.

Last but not least, we go beyond linear valuation functions and capture the main challenge for bounding the price of competition for any arbitrary valuation function. We present a principle which states that the lower bound for the price of competition for all 0-1 valuation functions is the same as the lower bound for the price of competition for all possible valuation functions. It is

* Supported in part by NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 21; pp. 21:1–21:14

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

worth mentioning that this principle not only works for the ranking duel but also for all dueling games. This principle says, in any dueling game, the most challenging part of bounding the price of competition is finding a lower bound for 0-1 valuation functions. We leverage this principle to show that the price of competition is at least 0.25 for the generalized ranking duel.

1998 ACM Subject Classification F. Theory of computation

Keywords and phrases POC, POA, Dueling games, Nash equilibria, sponsored search

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.21

1 Introduction

The conventional wisdom is that competition among suppliers will increase social welfare by providing consumers with competitive prices, high-quality products, and a wide range of options. A classic example is the Bertrand competition [10] where suppliers compete in price to incentivize consumers to buy from them and as a result the market price decreases to the point that it matches the marginal cost of production. Indeed there are many theoretical and empirical studies for supporting this belief in the economic literature (see, e.g., [29, 25, 24, 2]). However while in many markets the competition steers businesses to optimize their solutions for consumers, there are competitive markets in which businesses do not offer the best option to consumers. An interesting example for describing this situation is a dueling game, namely, a zero-sum game where two players compete to attract users. Immorlica, Kalai, Lucier, Moitra, Postlewaite, and Tennenholtz [19] showed surprisingly if players are aimed to beat their opponents in a dueling game, they may offer users suboptimal results. However, they raised this question regarding the efficiency of the competition as the authors write, *"Perhaps more importantly, one could ask about performance loss inherent when players choose their algorithms competitively instead of using the (single-player) optimal algorithm. In other words, what is the price of anarchy*[1] *of a given duel? ... Our main open question is (open question 1): does competition between algorithms improve or degrade expected performance?"* As we describe below, we study this open question for a set of dueling games and in particular for the ranking duel which is an appropriate representative of dueling games due to Immorlica et al. [19].

Dueling games. A *dueling game* $\mathcal{G}$ is a zero-sum game where two players compete for the attention of a user [2]. In a dueling game both players try to beat the other player and offer a better option with a higher value to the user. In particular, while the user's request is unknown to both players and they only have access to probability distribution p, the goal for each player is to maximize the probability that her offer is better than her opponent's offer. This framework falls within a general and natural class of ranking or social context games [7, 11], where each player plays a base game separately and then ultimate payoffs are determined by both their own outcomes and the outcomes of others. Immorlica et al. argue that this class of games models a variety of scenarios of competitions between algorithm designers, such as, competition between search engines (who must choose how to rank search

[1] Indeed Immorlica et al. [19] use the term of the *price of anarchy* in their aforementioned open question for the same concept of the *price of competition* in this paper.

[2] One can see the user as a population of users with the same behavior.

results), or competition between hiring managers (who must choose from a pool of candidates in the style of the secretary problem).

To be more precise a dueling game is defined by 4-tuple $\mathcal{G} = (\Omega, p, S, v)$, where Ω is the set of all possible requests from the user, p is a probability distribution over set Ω i.e., p_ω is the probability of requesting $\omega \in \Omega$ by the user, S is the set of all possible pure strategies for both players, and $v_\omega(s)$ is the value of pure strategy $s \in S$ for the user upon request $\omega \in \Omega$. Note that v is usually considered to be the valuation of the players, but in this paper valuation function v denotes the value for the user. While a mixed strategy is a probability distribution over all possible pure strategies in S, we write the value of mixed strategy $\mathbf{x}$ as $v_\omega(\mathbf{x}) = E_{s\sim\mathbf{x}}[v_\omega(s)]$.

A social planner is often interested in choosing a strategy which maximizes the social welfare, even though it may be a bad strategy in the competition between players. This means the social welfare maximizer strategy may not appear in any Nash equilibrium of the game, and thus the competition between players results in a suboptimal outcome for the users. Knowing the fact that Nash equilibria of a dueling game can be formed by suboptimal strategies, the following question seems to be an important question to ask regarding the inefficiency of this competition:

> What is the social welfare of any Nash equilibrium in a dueling game in comparison to the social welfare of the optimal strategy?

Price of competition. As aforementioned while in so many cases the competition motivates businesses to optimize their solutions for consumers, there are competitive markets and in particular dueling games of Immorlica et al. [19], in which businesses do not offer the best option to consumers. We define *price of competition* in this paper to capture this phenomenon.

First we note that since dueling games are two-player zero-sum games, Nash equilibria of these games are characterized by minimax strategies. Therefore, one can measure the inefficiency of any Nash equilibrium by comparing the welfare of any minimax strategy, in a game of competition between two players, with the welfare achieved by a social welfare maximizer. We are now ready to define the following criterion for measuring the quality of minimax strategies in a dueling game.

▶ **Definition 1. Price of competition (PoC)** is the ratio between the social welfare of the worst minimax strategy and the social welfare of the best possible strategy.

The proposed concept of the price of competition has the same spirit as the concept of the price of anarchy, and both concepts try to measure the inefficiency of Nash equilibria quantitatively. The price of anarchy, introduced by the seminal work of Koutsoupias and Papadimitriou [23], is a well-known concept in game theory that measures the ratio of the social welfare of the worst Nash equilibrium to the optimal social welfare. Although these two concepts are defined to capture properties of Nash equilibria, they are meaningfully different. In the price of anarchy, the social welfare is defined as the expected utility of all players in an equilibrium "outcome"[3] which is always zero for any zero-sum game. However, in the price of competition, the social welfare is the expected utility of the user (which is not a player) in a minimax "strategy". In fact, the price of competition is aimed to analyze the impact of the competition between players on an external user.

[3] Which is essentially the same as the sum of utilities of all players.

Since the price of competition captures the inefficiency of minimax strategies in two-player zero-sum games and all Nash equilibria of any two-player zero-sum game can be described by the set of minimax strategies, we believe the price of competition sheds new light on the structural analysis of Nash equilibria in two-player zero-sum games. Indeed as Alon, Demaine, Hajiaghayi, and Leighton [5] mention understanding the structure of Nash equilibiria, and not just the price of anarchy, is very important in general and thus our work is exactly toward this direction.

Due to the space constraints, all the missing proofs are provided in the appendices.

1.1 Our results

Ranking duel: To define the ranking duel more precisely, consider a ranking duel with two players. When a user submits a query to a player, she is basically searching a webpage which is unknown to the player. The player only has a prior knowledge about the requested webpage, i.e., for each webpage the probability that this webpage is requested by the user is known. The strategy of each player is an ordering for displaying webpages. When the requested webpage is realized, the player which puts this webpage in a higher rank gets the user attention, and thus wins the competition. The goal of each player is to maximize the probability of winning the competition. In this situation, a social planner who wants to minimize the expected rank of the requested webpage lists webpages in a decreasing order of their probabilities. However, this strategy may lose the competition to another strategy.[4]

We first investigate the quality of minimax strategies and prove that surprisingly the social welfare of any minimax strategy is not far from that of the optimal solution; it is 0.612 of the optimal solution for the linear valuation functions and 0.25 of the optimal for any arbitrary valuation function.

▶ **Theorem 1.** *Consider an instance of the raking duel. If the valuation function is a non-negative linear function of the rank, the price of competition is at least* 0.612 *for* $|\Omega| \geq 10$*, and at most* 0.833*.*

Our proof needs a careful understanding of properties for minimax strategies and has three main steps. First, we prove nice structural properties of minimax strategies. This step is the main step toward bounding the price of competition and gives an insight into properties of the polytope of minimax strategies. For example for every two webpages ω_1 and ω_2 with $p_{\omega_1} > p_{\omega_2}$, we prove there is a lower bound on the probability that any minimax strategy ranks webpage ω_1 before webpage ω_2. In the next step, we leverage these properties to write a factor-revealing mathematical program for bounding PoC. At last, we find a linear program where the set of its feasible solutions is a superset of the set of feasible solutions of the former mathematical program. We find the optimal solution of this linear program to formally prove the theorem for $|\Omega| \geq 10$. Moreover, we write a computer program to find the optimal solution of the corresponding linear program and show the price of competition is at least 0.637 for $|\Omega| \geq 100$ (which is slightly better the case that $\Omega \geq 10$). To the best of our knowledge, we are the first to use factor-revealing techniques to bound the inefficiency of equilibria.

[4] For example consider a situation when the user submits a query and she is interested in webpages w_1, w_2, and w_3 with probabilities 0.35, 0.33 and 0.32 respectively. In this situation the social planner ranks webpage w_i at position i, for $i = 1, 2, 3$. However, if a player plays based on this strategy, her opponent puts webpages w_2, w_3, and w_1 at positions 1, 2, and 3 respectively, and thus wins the competition when the user requests webpages w_2 or w_3. This means the social planner strategy loses the competition with probability 0.65.

Afterwards, we consider the cost minimization version of the ranking duel and prove a constant upper bound for the social cost of the game using the same technique of Theorem 1. Note that the only difference between the cost minimization and welfare maximization of a dueling game is that function v is a cost function rather than a valuation function, and once a webpage is searched the winner of the cost minimization game is the player who provides a solution with a lower cost. Moreover, we define the PoC_{cost} of the ranking duel as the ratio between the minimax strategy with the highest cost and the strategy with the least cost. In the following theorem we show that $\text{PoC}_{cost} \leq 3$.

▶ **Theorem 2.** *For a ranking duel with a linear cost function, we have* $PoC_{cost} \leq 3$.

It is worth mentioning that the structural properties of minimax strategies do not depend on the valuation function, and thus the polytope of minimax strategies remains unchanged for every valuation function which is a decreasing (increasing) function of rank in the welfare maximization (cost minimization) variant of the game. Therefore, we leverage structural properties of the polytope of minimax strategies, which is presented in Theorem 1, for proving Theorem 2 and in general one can apply our techniques for characterizing the polytope of minimax strategies for an arbitrary valuation function. Nevertheless, writing the factor-revealing mathematical program totally depends on the linearity of the valuation function.

General valuation functions: There are situations where the value of the user is not a linear function of rank. For example, consider a user that only cares about the top search results and will be satisfied if and only if her requested webpage is ranked higher than a certain threshold. We investigate the efficiency of minimax strategies for any non-negative non-linear valuation function. Moreover, we go beyond the ranking duel and consider other dueling games, in the pioneering work of [19]. While bounding the social welfare for arbitrary valuation functions and general dueling games seems to be challenging, we present a general principle to capture the main challenge of this problem. The proposed principle has the same spirit as the classic 0-1 principle in the sorting network which states: "a sorting network will sort any given input if and only if it sorts any given 0-1 input [13]." The following principle has the same message and shows if one can bound the social welfare for any 0-1 valuation function, the same bound holds for any arbitrary valuation function. This means the main challenge for bounding the social welfare is to bound it for 0-1 valuation functions. The main idea for proving Theorem 3 is to decompose any valuation function into 0-1 valuation functions.

▶ **Theorem 3.** *0-1 Principle: Consider a dueling game. If the price of competition is greater than* α *when the social welfare is defined based on any 0-1 valuation function, then it is greater than* α *when the social welfare is defined based on any valuation function.*

One can leverage this principle to analyze the efficiency of competition in any dueling game. For example, we show that the price of competition in the ranking duel is at least 0.25 for an arbitrary valuation function.

▶ **Theorem 4.** *The price of competition is at least* 0.25 *for the ranking duel, when the social welfare is defined based on an arbitrary valuation function.*

In the proof of Theorem 4, based on the 0-1 principle, we first consider the problem with pseudo-valuation functions in which the value of each position is either 0 or 1. We consider $\mathbf{x}^*$ as the minimax strategy with the least social welfare and construct a response strategy $\mathbf{x}'_i$ for the second player for every $1 \leq i \leq n$ as follows:

- Draw a permutation randomly based on strategy $\mathbf{x}^*$. If the value of the position of the i-th webpage is 1 then play that permutation. Otherwise, swap the position of the i-th webpage with one of the positions with value 1 at random and play the new permutation.

Next, we use the fact that minimax strategy $\mathbf{x}^*$ does not lose to strategy $\mathbf{x}'_i$ for proving a set of inequalities which later on helps us to bound the price of competition. Finally, we use the 0-1 principle to show that this lower bound holds for all possible valuation functions.

This principle also helps us to provide upper bounds on the price of competition when one considers a general valuation function. For example we show that the PoC of the following game introduced by Immorlica et al. [19] cannot be bounded by any constant value:

Binary search duel: The binary search duel is a dueling game where each player chooses a binary search tree over the set of all possible requests Ω. When the user's request $\omega \in \Omega$ is realized, the value for each strategy is defined based on the depth of request ω in the corresponding binary search tree.

▶ **Theorem 5.** *The price of competition is $\mathcal{O}(\frac{1}{|\Omega|})$ for the binary search duel, when the social welfare is defined based on an arbitrary valuation function.*

In order to construct bad instances for these duels, we design a valuation function which is 1 for low depths, 0 for high depths, and a small positive value ϵ in between. We show the price of competition is less than any given number $\beta > 0$ for the binary search duel by constructing an instance of the binary search duel with $|\Omega| = \Theta(\frac{1}{\beta})$.

1.2 Related work

Immorlica et al. [19] are the first who considered the concept of dueling games. They present dueling games in the context of dueling algorithms, where two competitive algorithms try to maximize the probability of outperforming their opponent for an unknown stochastic input. While we employ the same model in this paper, our goal completely differs from that of Immorlica et al. [19]. Immorlica et al. [19] present polynomial-time algorithm for finding a minimax strategy of a dueling game when the polytope of minimax strategies can be represented by a polynomial number of linear constraints. Knowing the fact that the polytope of minimax strategies of any ranking duel has polynomially many facets, they propose a polynomial-time algorithm for finding a minimax strategy of ranking duels. This method was later generalized by [3] to solve the Colonel Blotto game. Immorlica et al. [19] leave the problem of analyzing the social welfare of competitive algorithms as their main open question. In this paper, we do not deal with the computational complexity of finding minimax strategies, but we focus on answering the posted open question and analyze the social welfare of minimax strategies for a given duel.

As we are interested in quantifying the inefficiency of Nash equilibria, our proposed concept of the price of competition has the same flavor as the concept of the price of anarchy [23, 26]. The price of anarchy is commonly used for quantifying the inefficiency of a system which is constructed by selfish agents. For example, it has been used to analyze the inefficiency of Nash equilibria in congestion games [27, 12], network creation games [16, 14, 4, 5], and selfish scheduling games [6, 20]. (See, e.g. [26] for more examples).

Kempe and Lucier [21] recently study the impact of competition on the social welfare in a competitive sponsored search market. In their model, which is a departure from the model of Immorlica et al. [19], search engines again compete to obtain more users. A user's request is defined by a set S of webpages which is unknown to search engines, and the user is satisfied if and only if at least one of webpages in S is ranked in a better position than a given threshold t. The strategy of each search engine is an ordering over all possible webpages.

At last, the user chooses a search engine based on a selection rule which is a function of probability of being satisfied by each search engine. Kempe and Lucier [21] prove that if search engines extract utility from satisfied users or the search engine selection rule is convex, then the social welfare of the game is at least half of the optimum social welfare. Moreover, they show if the utility of search engines is driven from all customers and the search engine selection rule is concave, then the social welfare of the game is bounded away from that of the optimum solution by a factor of $\Omega(n)$, where n is the number of all possible webpages. We would like to note that our model is a general model for studying all dueling games which is exactly the same as the model of Immorlica et al. [19], and is significantly different from that of Kempe and Lucier [21].

There is a line of research that study a competition between advertisers in sponsored search auctions [1, 9, 17, 15, 22]. These works analyze the revenue of a single search engine in various settings regarding users' behavior and the business model of advertisers. However, in ranking duel we investigate a competition between players who provide orderings rather than advertisers.

There is a rich literature in economics that explains product differentiation in competitive markets. While producing similar products is supported by classical models such as the Hotelling model [18], Aspremont, Gabszewicz, and Thisse [8] argue that competitive producers may improve their revenue by producing different products. See, e.g., [29, 25, 24] for details on this literature. The same phenomenon can be seen in the sponsored search market, e.g., Telang, Rajan, and Mukhopadhyay [28] show low-quality search engines may extract revenue from the sponsored search market.

2 Model

2.1 Dueling games

In dueling game $\mathcal{G}$ both players try to beat the other player and offer a better value in the competition. Assume players A and B play pure strategies s_A and s_B respectively, and event ω has occurred. In this situation, player A wins the competition if and only if $v_\omega(s_A) > v_\omega(s_B)$, and thus the utility of player A given event ω can be written as follows:

$$u^A_\omega(s_A, s_B) = \begin{cases} +1 & \text{if } v_\omega(s_A) > v_\omega(s_B) \\ 0 & \text{if } v_\omega(s_A) = v_\omega(s_B) \\ -1 & \text{if } v_\omega(s_A) < v_\omega(s_B) \end{cases}$$

Now consider a situation where players A and B play mixed strategies $\mathbf{x}$ and $\mathbf{y}$ respectively and event ω has occurred. The utility of player A is the probability that player A wins the competition minus the probability that player B wins the competition and can be defined as follows:

$$u^A_\omega(\mathbf{x}, \mathbf{y}) = Pr_{\substack{s_A \sim \mathbf{x} \\ s_B \sim \mathbf{y}}}[v_\omega(s_A) > v_\omega(s_B)] - Pr_{\substack{s_A \sim \mathbf{x} \\ s_B \sim \mathbf{y}}}[v_\omega(s_A) < v_\omega(s_B)]$$

Finally the overall utility of player A is $u^A(\mathbf{x}, \mathbf{y}) = \sum_\omega p_\omega u^A_\omega(\mathbf{x}, \mathbf{y})$. Since dueling game $\mathcal{G}$ is a zero-sum game the utility of player B is the negation of the utility of player A for each ω, i.e., $u^B_\omega(\mathbf{x}, \mathbf{y}) = -u^A_\omega(\mathbf{x}, \mathbf{y})$ and thus $u^B(\mathbf{x}, \mathbf{y}) = -u^A(\mathbf{x}, \mathbf{y})$.

▶ **Definition 2** (Minimax strategy). Strategy $\mathbf{x}$ of player A is minimax if $\mathbf{x} \in \text{argmax}_{\mathbf{x}'}\{\min_{\mathbf{y}}\{u^A(\mathbf{x}', \mathbf{y})\}\}$. Similarly, Strategy $\mathbf{y}$ of player B is minimax if $\mathbf{y} \in \text{argmax}_{\mathbf{y}'}\{\min_{\mathbf{x}}\{u^B(\mathbf{x}, \mathbf{y}')\}\}$.

Based on the definition of dueling games and the fact that the set of all possible pure strategies for both players is S, we can conclude the outcome of both players in any Nash equilibrium is 0 and moreover the set of minimax strategies of both players coincide. We define the set of minimax strategies by $\mathcal{M}$.

▶ **Definition 3** (Social welfare)**.** Consider dueling game $\mathcal{G} = (\Omega, p, S, v)$. The social welfare of pure strategy s is the expected value of this strategy over all possible events and can be written as $\mathrm{SW}(s) = \sum_\omega p_\omega v_\omega(s)$. The social welfare of mixed strategy $\mathbf{x}$ is $\mathrm{SW}(\mathbf{x}) = E_{s\sim\mathbf{x}}[\mathrm{SW}(s)]$.

In this paper, we are interested to study the social welfare of the game in equilibria. Note that the customer locks into one of the players in long term. On the other hand, both players only try to offer the customer a better option than the other one, and thus play a minimax strategy in the competition. These cause inefficiency in the game. Here we define a new criterion to measure this inefficiency in the game.

▶ **Definition 4** (Price of competition)**.** The price of competition is the ratio of the worst minimax strategy to the optimal solution which is:

$$\frac{\min_{\mathbf{x}\in\mathcal{M}} \mathrm{SW}(\mathbf{x})}{\max_{\mathbf{x}} \mathrm{SW}(\mathbf{x})} = \frac{\min_{\mathbf{x}\in\mathcal{M}} \mathrm{SW}(\mathbf{x})}{\max_{s\in S} \mathrm{SW}(s)}.$$

Similar to the welfare maximization model, we consider the cost minimization model in which players try to beat the opponent by offering a lower cost to the user. In particular we have a cost function c, such that $c_\omega(s)$ denotes the cost of strategy s and event ω. Hence, the utility of player A would be defined as

$$u^A_\omega(\mathbf{x}, \mathbf{y}) = Pr_{\substack{s_A\sim\mathbf{x}\\ s_B\sim\mathbf{y}}}[c_\omega(s_A) < c_\omega(s_B)] - Pr_{\substack{s_A\sim\mathbf{x}\\ s_B\sim\mathbf{y}}}[c_\omega(s_A) > c_\omega(s_B)].$$

Similarly we define the social cost $\mathrm{SC}(s) = \sum_\omega p_\omega c_\omega(s)$ for a pure strategy s and $\mathrm{SC}(\mathbf{x}) = E_{s\sim\mathbf{x}}[\mathrm{SC}(s)]$ for a mixed strategy $\mathbf{x}$. Finally the price of competition in cost minimization version is defined as

$$\frac{\max_{\mathbf{x}\in\mathcal{M}} \mathrm{SC}(\mathbf{x})}{\min_{\mathbf{x}} \mathrm{SC}(\mathbf{x})} = \frac{\max_{\mathbf{x}\in\mathcal{M}} \mathrm{SC}(\mathbf{x})}{\min_{s\in S} \mathrm{SC}(s)}.$$

2.2 Ranking duel

Ranking duel is a dueling game where $\Omega = \{1, \cdots, n\}$ is the set of n webpages which can be requested by a user. In this game, the set of pure strategies S is equal to the set of all possible permutations over Ω, i.e., each player outputs an ordering of webpages for the user. We denote each pure strategy of the ranking duel by π (instead of s) where $\pi(\omega)$ is the rank of webpage ω. The valuation function v of a raking duel can be defined based on function $f : \{1, \cdots, n\} \to R^+ \cup \{0\}$ as $v_\omega(\pi) = f(\pi(\omega))$. Consider mixed strategy $\mathbf{x}$ where $\mathbf{x}_\pi$ is the probability that strategy $\mathbf{x}$ outputs permutation π. The *social welfare* of strategy $\mathbf{x}$ can be defined as:

$$\mathrm{SW}(\mathbf{x}) = \sum_\omega \sum_\pi p_\omega \mathbf{x}_\pi f(\pi(\omega)). \tag{1}$$

3 Price of competition in the linear ranking duel

3.1 Welfare maximization ranking duel

In this section we give bounds for the PoC in the ranking duel when the valuation function is non-negative and linear, in other words $f(i) = c(n - i) + d$, where $c, d \geq 0$.

First we formulate the social welfare of strategy $\mathbf{x}$ and the optimal social welfare. Without loss of generality in this section we assume $p_1 \geq p_2 \geq \ldots \geq p_n$. Let $Pr_{\pi\sim\mathbf{x}}[\pi(a) = i]$ denote the probability that in a randomly drawn permutation π from strategy $\mathbf{x}$, the rank of webpage a is i. Similarly let $Pr_{\pi\sim\mathbf{x}}[\pi(a) < \pi(b)]$ denote the probability that in a randomly drawn permutation π from strategy $\mathbf{x}$, webpage a comes before webpage b.

▶ **Proposition 5.** *In a ranking duel with valuation function f and n webpages, the social welfare of a strategy $\mathbf{x}$ is $SW_f(\mathbf{x}) = \sum_{a=1}^{n}\sum_{i=1}^{n} p_a Pr_{\pi\sim\mathbf{x}}[\pi(a) = i]f(i)$.*

Let OPT be the strategy with the maximum social welfare. Hence SW(OPT) is formulated as follows.

▶ **Proposition 6.** *In a ranking duel with valuation function f and n webpages, the optimal social welfare is $SW_f(OPT) = \sum_{a=1}^{n} p_a f(a)$.*

Lemma 7 shows that for any minimax strategy $\mathbf{x}$ and any linear function $f(i) = c(n-i)+d$ with $c, d \geq 0$, the PoC is no less than the case in which $f(i) = n - i$.

▶ **Lemma 7.** *For valuation functions $f(i) = n - i$, $f'(i) = c(n - i) + d$ with $c, d \geq 0$, and any strategy $\mathbf{x}$, $\frac{SW_f(\mathbf{x})}{SW_f(OPT)} \leq \frac{SW_{f'}(\mathbf{x})}{SW_{f'}(OPT)}$.*

Thus any lower bound for the PoC with $f(i) = n - i$, is also a lower bound for the PoC with any other linear valuation function. Therefore, from now on we assume $f(i) = n - i$, and use SW($\mathbf{x}$) and SW(OPT) instead of $\text{SW}_f(\mathbf{x})$ and SW_f(OPT), respectively. Hence $\text{SW(OPT)} = \sum_{a=1}^{n} p_a(n - a)$. Now we try to compute SW($\mathbf{x}$) from a different perspective.

▶ **Proposition 8.** *In a ranking duel with n webpages, the social welfare of strategy $\mathbf{x}$ is*

$$SW(\mathbf{x}) = \sum_{a=1}^{n}\sum_{b=a+1}^{n} p_a Pr_{\pi\sim\mathbf{x}}[\pi(a) < \pi(b)] + p_b Pr_{\pi\sim\mathbf{x}}[\pi(b) < \pi(a)].$$

Intuitively by Proposition 8 we can compute the social welfare of a strategy by comparing the ranks of every pairs of webpages. Therefore we define $h_{ab}(\mathbf{x})$ to be the amount that the pair of webpages a and b adds to the social welfare in strategy $\mathbf{x}$, i.e. $h_{ab}(\mathbf{x}) = p_a Pr_{\pi\sim\mathbf{x}}[\pi(a) < \pi(b)] + p_b Pr_{\pi\sim\mathbf{x}}[\pi(b) < \pi(a)]$. Thus we can rewrite Proposition 8 as $\text{SW}(\mathbf{x}) = \sum_{a=1}^{n}\sum_{b=a+1}^{n} h_{ab}(\mathbf{x})$. Hence for every strategy $\mathbf{x}$,

$$\frac{\text{SW}(\mathbf{x})}{\text{SW(OPT)}} = \frac{\sum_{a=1}^{n}\sum_{b=a+1}^{n} h_{ab}(\mathbf{x})}{\sum_{a=1}^{n} p_a(n-a)}.$$

In Lemma 9 we provide our main tool for bounding the price of competition in the linear ranking duel.

▶ **Lemma 9.** *Given a strategy $\mathbf{x}$, if there exist an integer k such that $2 \leq k \leq n$ and for all k different indices $i_1 < i_2 < \ldots < i_k$,*

$$\frac{\sum_{a=1}^{k}\sum_{b=a+1}^{k} h_{i_a i_b}(\mathbf{x})}{\sum_{a=1}^{k} p_{i_a}(k-a)} \geq \alpha,$$

then $\frac{SW(\mathbf{x})}{SW(OPT)} \geq \alpha$.

Now our goal is to provide a lower bound for α when $\mathbf{x}$ is a minimax strategy. In order to do that, first we provide some structural properties of the minimax strategies. Leveraging

these properties we write a mathematical program with k variables p_a and $\binom{k}{2}$ variables h_{ab}. Finally, we provide a factor-revealing linear program to obtain a close lower bound for α in the corresponding mathematical program.

In Lemmas 10, 11, 13, and Proposition 12 we provide the structural properties of the minimax strategies.

▶ **Lemma 10.** *Let* $\mathbf{x}$ *be a minimax strategy and* a *and* b *be two webpages such that* $p_a \geq p_b$. *Let* π_{ba} *be any permutation in the support of* $\mathbf{x}$ *in which* b *precedes* a. *Let* $i < j$ *be the respective position of* a *and* b *in* π_{ba}, *then strategy* $\mathbf{x}$ *must satisfy,*

$$Pr_{\pi\sim\mathbf{x}}[i < \pi(b) \leq j] + Pr_{\pi\sim\mathbf{x}}[i \leq \pi(b) < j] \geq \frac{p_a}{p_b}(Pr_{\pi\sim\mathbf{x}}[i < \pi(a) \leq j] + Pr_{\pi\sim\mathbf{x}}[i \leq \pi(a) < j]).$$

Intuitively Lemma 10 shows that if $p_a \geq p_b$ and there is a permutation in which b comes before a, then the probability that $\mathbf{x}$ ranks b in interval $[i, j]$ (counting the non-endpoint elements twice) is greater than the probability that $\mathbf{x}$ ranks a in this interval by a factor of $\frac{p_a}{p_b}$. Otherwise, by swapping the rank of a and b we can achieve a strategy that beats $\mathbf{x}$.

▶ **Lemma 11.** *Let* $\mathbf{x}$ *be a minimax strategy and* $\mathbf{x}_\pi$ *be the probability that strategy* $\mathbf{x}$ *plays permutation* π. *For every pair of webpages* a *and* b *with* $p_a \geq p_b$, *we have*

$$Pr_{\pi\sim\mathbf{x}}[\pi(a) < \pi(b)] \geq (\frac{p_a}{2p_b} - 1) Pr_{\pi\sim\mathbf{x}}[\pi(b) < \pi(a)]. \tag{2}$$

Briefly, in the proof of Lemma 11 we propose an algorithm to find a set of permutations Π in $\mathbf{x}$, such that 1) for each $\pi \in \Pi$, b comes before a, 2) for each permutation π' in $\mathbf{x}$ in which b comes before a, there is a permutation $\pi \in \Pi$, such that $\pi(b) \leq \pi'(a) \leq \pi(a)$, and 3) the interval of the ranks of b and a are distinct, i.e. for two permutations $\pi, \pi' \in \Pi, [\pi(b), \pi(a)] \cap [\pi'(b), \pi'(a)] = \emptyset$. We apply the inequality in Lemma 10 for all permutations in Π to achieve Lemma 11.

In Proposition 12 and Lemma 13 we provide lower bounds for $h_{ab}(\mathbf{x})$ when $\mathbf{x}$ is a minimax strategy. Hence we can use these lower bounds in the proposed mathematical program to achieve a lower bound for the PoC.

▶ **Proposition 12.** *For minimax strategy* $\mathbf{x}$ *and webpages* a *and* b *such that* $p_a \geq p_b$, $h_{ab}(\mathbf{x}) \geq p_b$.

▶ **Lemma 13.** *For minimax strategy* $\mathbf{x}$ *and webpages* a *and* b *such that* $p_a \geq p_b$, $h_{ab}(\mathbf{x}) \geq p_a - 2p_b + \frac{2p_b^2}{p_a}$.

Leveraging the properties of the minimax strategies we write MP 3. In MP 3, Constraints 5 and 6 force p_a's to satisfy the probability constraints. Using Proposition 12, Constraint 7 forces h_{ab} to be no less than p_b and due to Lemma 13, Constraint 7 forces h_{ab} to be no less than $p_a - 2p_b + \frac{2p_b^2}{p_a}$. By Lemma 9, α in Constraint 4 gives a lower bound for the PoC.

$$\text{minimize} \quad \alpha \tag{3}$$

$$\text{subject to} \quad \alpha = \frac{\sum_{a=1}^{k}\sum_{b=a+1}^{k} h_{ab}}{\sum_{a=1}^{k} p_a(k-a)} \tag{4}$$

$$p_a \geq 0 \qquad \forall 1 \leq a \leq k \tag{5}$$

$$\sum_{1\leq a\leq k} p_a \leq 1 \tag{6}$$

$$h_{ab} \geq p_b \qquad \forall 1 \leq a < b \leq k \tag{7}$$

$$h_{ab} \geq p_a - 2p_b + \frac{2p_b^2}{p_a} \qquad \forall 1 \leq a < b \leq k \tag{8}$$

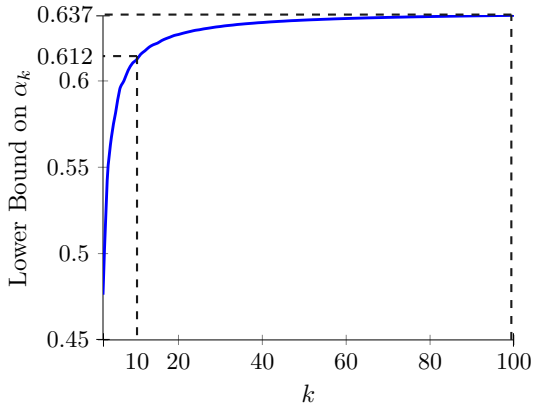


Figure 1 Lower bound on the solution of MP 3. While we formally prove $\alpha_{10} \geq 0.612$, this figure shows lower bounds on α_k for $2 \leq k \leq 100$ found by a computer program. Note that, by Lemma 14 the PoC of the ranking duel with linear valuation function is at least α_k for all $n \geq k$.

For each k, let α_k be the optimal value of the objective function in MP 3.

▶ **Lemma 14.** *α_k is a lower bound for the PoC of the linear ranking duel where $n \geq k$.*

In Theorem 15 we formally prove $\alpha_{10} \geq 0.612$, which results in PoC ≥ 0.612 for any ranking duel with $n \geq 10$ webpages. Moreover, we write a computer program to find α_k for $2 \leq k \leq 100$ (see Figure 1).

▶ **Theorem 15.** *For a linear ranking duel with $n \geq 10$ webpages, PoC ≥ 0.612.*

The proof idea is as follows. We design a linear program from MP 3. Constraints 4 and 8 in MP 3 are not linear, thus we first try to replace Constraint 8 by three linear constraints. Afterwards, intuitively we scale the probabilities such that $\sum_{a=1}^{k} p_a(k-a)$ equals 1, hence we can have a linear constraint instead of Constraint 4. Then we show each feasible solution for MP 3 is also a feasible solution for the achieved linear program. Finally by finding a feasible solution for the dual of the corresponding LP, we provide a lower bound for α in the primal LP, which is a lower bound for the optimal solution of MP 3 as well.

4 General framework

In this section we present a general framework for analyzing the price of competition in dueling games. Proving lower bounds for the price of competition in dueling games highly depends on the valuation functions and it becomes more challenging when the valuation functions are complex. However, the behavior of minimax strategies only depends on the comparison of the valuation functions rather than actual values. We leverage this fact to provide Theorem 16 which enables us to prove bounds for the price of competition without concerning the complexities of the valuation functions. We refer to this theorem as the *0-1 principle*.

Let (Ω, p, S, v) be a dueling game and α be a non-negative real number. We define the trigger function $v_\omega^\alpha(s)$ for a pure strategy s in the following way:

$$v_\omega^\alpha(s) = \begin{cases} 1 & \text{if } v_\omega(s) \geq \alpha \\ 0 & \text{if } v_\omega(s) < \alpha \end{cases}$$

Moreover, we define the pseudo-welfare function $\mathrm{SW}^\alpha(s)$ as the summation of the values of the trigger functions when a player is playing strategy s with respect to α, $\mathrm{SW}^\alpha(s) = \sum_{\omega\in\Omega} p_\omega v^\alpha_\omega(s)$. Furthermore, the pseudo-welfare function for a mixed strategy $\mathbf{x}$ is defined as $\mathrm{SW}^\alpha(\mathbf{x}) = E_{s\sim\mathbf{x}}[\mathrm{SW}^\alpha(s)]$. Let PoC^α be the pseudo-welfare of the minimax strategy with the least social welfare over SW(OPT) which can be formulated by $\mathrm{PoC}^\alpha = \frac{\mathrm{SW}^\alpha(\mathbf{x}^*)}{\mathrm{SW}^\alpha(\mathrm{OPT})}$, where $\mathbf{x}^*$ is the minimax strategy with the least social welfare and OPT is the strategy with highest social welfare. Note that optimal and minimax strategies are determined regardless of the pseudo-welfare function. For simplicity, we consider $\mathrm{PoC}^\alpha = 1$ when $\mathrm{SW}^\alpha(\mathrm{OPT}) = 0$. In the following we show that the PoC of every dueling game is bounded by $\min_{\alpha\geq 0}\{\mathrm{PoC}^\alpha\}$.

▶ **Theorem 16** (0-1 principle). *For every dueling game we have* $PoC \geq \min_{\alpha\geq 0}\{PoC^\alpha\}$.

In the following subsections we show how we can apply the 0-1 principle to dueling games in order to present lower bounds for the PoC. In Subsection 4.1 we show that the PoC of the ranking duel is at least $\frac{1}{4}$ regardless of the valuation function. Note that, for every α, one could design a valuation function in such a way that $|v^\alpha_\omega(\mathbf{x}) - v_\omega(\mathbf{x})| \leq \epsilon$ while the optimal and minimax strategies remain the same. Therefore, we have the lowest PoC when the range of the valuation function is $[0, \epsilon] \cup [1, 1 + \epsilon]$.

4.1 Ranking duel with general valuation function

Recall that in the ranking duel each position of the permutation has a valuation $f(i)$, each pure strategy of the players is a permutation of webpages $\pi = \langle \pi^{-1}(1), \pi^{-1}(2), \pi^{-1}(3), \ldots, \pi^{-1}(n)\rangle$, and $\Omega = \{1, 2, \ldots, n\}$ is the set of elements of uncertainty. For a webpage $\omega \in \Omega$, $v_\omega(\pi) = f(\pi(\omega))$, where $\pi(\omega)$ is the rank of ω in π. In the following, we use the 0-1 principle to show that the PoC of the ranking duel with an arbitrary valuation function is at least $\frac{1}{4}$.

▶ **Theorem 17.** *The PoC of the ranking duel is at least* $\frac{1}{4}$.

4.2 Binary search duel with general valuation function

In this subsection we study the binary search duel and show that the PoC of this game can be $\Omega(\frac{1}{n})$. In this game $\Omega = \{1, 2, \ldots, n\}$ and each pure strategy of the players is a binary tree such that its in-order traversal visits the elements of Ω in the sorted order. Moreover, $v_\omega(s)$ is determined by $f(d_s(\omega))$ where $d_s(\omega)$ denotes the depth of element ω in the binary search tree corresponding to s and $f : \mathbb{N} \to \mathbb{R}_{\geq 0}$ is a decreasing function.

▶ **Theorem 18.** *For every* $\beta > 0$ *there is an instance of the binary search duel with* $|\Omega| = \theta(\frac{1}{\beta})$ *and* $PoC \leq \beta$.

Acknowledgment. We would like to gratefully thank Rakesh Vohra and Brendan Lucier for their insightful discussions.

References

1 Gagan Aggarwal, Jon Feldman, S. Muthukrishnan, and Martin Pál. Sponsored search auctions with markovian users. In *WINE*, pages 621–628. 2008.

2 AmirMahdi Ahmadinejad, Sina Dehghani, MohammadTaghi Hajiaghayi, Hamid Mahini, Saeed Seddighin, and Sadra Yazdanbod. Forming external behaviors by leveraging internal opinions. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 1849–1857. IEEE, 2015.

3 Mahdi Ahmadinejad, Sina Dehghani, MohammadTaghi Hajiaghayi, Brendan Lucier, Hamid Mahini, and Saeed Seddighin. From duels to battefields: Computing equilibria of blotto and other games. *AAAI 2016*, 2016.

4 Susanne Albers, Stefan Eilts, Eyal Even-Dar, Yishay Mansour, and Liam Roditty. On nash equilibria for a network creation game. In *SODA*, pages 89–98, 2006.

5 Noga Alon, Erik D. Demaine, Mohammad T. Hajiaghayi, and Tom Leighton. Basic network creation games. *SIAM Journal on Discrete Mathematics*, 27(2):656–668, 2013.

6 Nir Andelman, Michal Feldman, and Yishay Mansour. Strong price of anarchy. In *SODA*, pages 189–198, 2007.

7 Itai Ashlagi, Piotr Krysta, and Moshe Tennenholtz. Social context games. In *Internet and Network Economics*, pages 675–683. Springer, 2008.

8 Claude Aspremont, J. Jaskold Gabszewicz, and J.-F. Thisse. On hotelling's" stability in competition". *Econometrica: Journal of the Econometric Society*, pages 1145–1150, 1979.

9 Susan Athey and Glenn Ellison. Position auctions with consumer search. Technical Report 15253, National Bureau of Economic Research, 2009.

10 J. Bertrand. Book review of théorie mathématique de la richesse sociale and of recherches sur les principes mathématiques de la théorie des richesses. *Journal de Savants*, 67:499–508, 1883.

11 Felix Brandt, Felix Fischer, Paul Harrenstein, and Yoav Shoham. Ranking games. *Artificial Intelligence*, 173(2):221–239, 2009.

12 George Christodoulou and Elias Koutsoupias. The price of anarchy of finite congestion games. In *STOC*, pages 67–73, 2005.

13 Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

14 Erik D Demaine, MohammadTaghi Hajiaghayi, Hamid Mahini, and Morteza Zadimoghaddam. The price of anarchy in network creation games. In *PODC*, pages 292–298, 2007.

15 Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

16 Alex Fabrikant, Ankur Luthra, Elitza Maneva, Christos H. Papadimitriou, and Scott Shenker. On a network creation game. In *PODC*, pages 347–351, 2003.

17 Anindya Ghose and Sha Yang. An empirical analysis of search engine advertising: Sponsored search in electronic markets. *Management Science*, 55(10):1605–1622, 2009.

18 Harold Hotelling. Stability in competition. *The Economic Journal*, 39(153):41–57, 1929.

19 Nicole Immorlica, Adam Tauman Kalai, Brendan Lucier, Ankur Moitra, Andrew Postlewaite, and Moshe Tennenholtz. Dueling algorithms. In *STOC*, pages 215–224, 2011.

20 Nicole Immorlica, Li Erran Li, Vahab S. Mirrokni, and Andreas S. Schulz. Coordination mechanisms for selfish scheduling. *Theoretical Computer Science*, 410(17):1589–1598, 2009.

21 David Kempe and Brendan Lucier. User satisfaction in competitive sponsored search. In *WWW*, pages 699–710, 2014.

22 David Kempe and Mohammad Mahdian. A cascade model for externalities in sponsored search. In *Internet and Network Economics*, pages 585–596. 2008.

23 Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *STACS*, pages 404–413, 1999.

24 David M Kreps. *A course in microeconomic theory*. Harvester Wheatsheaf New York, 1990.

25 Andreu Mas-Colell, Michael Dennis Whinston, and Jerry R. Green. *Microeconomic theory*. Oxford university press New York, 1995.

26 Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

27 Tim Roughgarden. *Selfish routing and the price of anarchy*, volume 174. MIT press Cambridge, 2005.

28 Rahul Telang, Uday Rajan, and Tridas Mukhopadhyay. The market structure for internet search engines. *Journal of Management Information Systems*, 21(2):137–160, 2004.

29 Hal R Varian and Jack Repcheck. *Intermediate microeconomics: a modern approach*. WW Norton & Company New York, NY, 8th edition, 2010.

Popular Half-Integral Matchings

Telikepalli Kavitha*

Tata Institute of Fundamental Research, Mumbai, India
kavitha@tcs.tifr.res.in

Abstract

In an instance $G = (A \cup B, E)$ of the stable marriage problem with strict and possibly incomplete preference lists, a matching M is popular if there is no matching M' where the vertices that prefer M' to M outnumber those that prefer M to M'. All stable matchings are popular and there is a simple linear time algorithm to compute a maximum-size popular matching. More generally, what we seek is a *min-cost* popular matching where we assume there is a cost function $c : E \to \mathbb{Q}$. However there is no polynomial time algorithm currently known for solving this problem. Here we consider the following generalization of a popular matching called a popular *half-integral* matching: this is a fractional matching $\vec{x} = (M_1 + M_2)/2$, where M_1 and M_2 are the 0-1 edge incidence vectors of matchings in G, such that $\vec{x}$ satisfies popularity constraints. We show that every popular half-integral matching is equivalent to a stable matching in a larger graph G^*. This allows us to solve the min-cost popular half-integral matching problem in polynomial time.

1998 ACM Subject Classification G.2.2 Graph algorithms, G.1.6 Linear programming

Keywords and phrases bipartite graphs, stable matchings, fractional matchings, polytopes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.22

1 Introduction

Let $G = (A \cup B, E)$ be an instance of the stable marriage problem on n vertices and m edges. Each vertex has a strict preference list ranking its neighbors. A matching M is stable if M admits no *blocking edge*, i.e., an edge (a, b) such that both a and b prefer each other to their respective assignments in M. The existence of stable matchings in G and the Gale-Shapley algorithm [7] to find one are classical results in graph algorithms.

Stability is a very strict condition and here we consider a relaxation of this called *popularity*. This notion was introduced by Gärdenfors [9] in 1975. We say a vertex $u \in A \cup B$ *prefers* matching M to matching M' if u is matched in M and unmatched in M' or it is matched in both and $M(u)$ ranks better than $M'(u)$ in u's preference list. For any two matchings M and M' in G, let $\phi(M, M')$ be the number of vertices that prefer M to M'.

▶ **Definition 1.** A matching M is *popular* if $\phi(M, M') \geq \phi(M', M)$ for every matching M' in G, i.e., $\Delta(M, M') \geq 0$ where $\Delta(M, M') = \phi(M, M') - \phi(M', M)$.

Every stable matching is popular [9]. In fact, it is known that every stable matching is a minimum-size popular matching [10]. In applications such as matching students to projects or applicants to posts, it may be useful to consider a weaker notion (such as popularity) than the total absence of blocking edges for the sake of obtaining larger-sized matchings. Popularity provides "global stability" since a popular matching never loses an election to another matching; by relaxing stability to popularity, we have a larger pool of candidate matchings to choose from in such an application.

* Part of this work was done during a visit to IIT Delhi.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 22; pp. 22:1–22:13


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

When there is a cost function $c : E \to \mathbb{Q}$, what we seek is a min-cost popular matching. There are several polynomial time algorithms known [11, 5, 6, 16, 14, 15] for computing a min-cost stable matching in G. However, while a maximum-size popular matching can be computed in linear time [12], no polynomial time algorithm is currently known for computing a min-cost popular matching in an instance $G = (A \cup B, E)$ with strict preference lists, except when preference lists are complete [4].

A fractional matching $\vec{p}$ is a convex combination of matchings, i.e., $\vec{p} = \sum_i p_i \cdot I(M_i)$ where $\sum_i p_i = 1$, $p_i \geq 0$ for all i, M_i's are matchings in G, and $I(M)$ is the 0-1 edge incidence vector of M. The fractional matching $\vec{p}$ is *popular* if $\Delta(\vec{p}, M) \geq 0$ for all matchings M in G where $\Delta(\vec{p}, M) = \sum_i p_i \cdot \Delta(M_i, M)$ (see Definition 1). It follows by linearity that if $\vec{p}$ is a popular fractional matching then $\Delta(\vec{p}, \vec{q}) \geq 0$ for all fractional matchings $\vec{q}$.

Let $\mathcal{P}$ be the polytope defined by the constraints that $\vec{p}$ belongs to the matching polytope of G and $\Delta(\vec{p}, M) \geq 0$ for all matchings M in G. A simple description of $\mathcal{P}$ was given in [13]. Thus a min-cost popular *fractional* matching can be computed in polynomial time.

Our results and techniques. Our main result is a polynomial time algorithm to compute a min-cost popular *half-integral* matching in G. A popular half-integral matching is a vector $\vec{x} \in \{0, \frac{1}{2}, 1\}^m \cap \mathcal{P}$. For any two popular matchings M_1 and M_2 in G, the half-integral matching $(I(M_1) + I(M_2))/2$ is popular. However not every popular half-integral matching is a convex combination of popular matchings – we show such an example in Section 2. Thus if $\mathcal{Q}$ is the convex hull of popular half-integral matchings in G, then $\mathcal{Q}$ need not be integral.

We show that every extreme point of $\mathcal{Q}$ is a stable matching in a new (larger) graph G^* that we construct here. Thus the min-cost popular half-integral matching problem in G becomes the min-cost stable matching problem in G^* which can be solved in polynomial time. This also gives us a simple description of the polytope $\mathcal{Q}$ via the stable matching polytope of G^* (i.e., the convex hull of stable matchings in G^*).

The main tool that we use here is the description of the polytope $\mathcal{P}$ from [13]. We first show that every stable matching S in the new graph G^* can be mapped to a half-integral matching in G whose incidence vector belongs to $\mathcal{P}$. We then show that every extreme point $\vec{p}$ of the convex hull $\mathcal{Q}$ of popular half-integral matchings in G can be realized as a stable matching in G^*. We use the fact that $\vec{p} \in \mathcal{P}$ along with the fact that G is bipartite to show a "helpful witness" $(\alpha_u)_{u \in A \cup B} \in \{\pm 1, 0\}^n$. This witness will guide us in building a stable matching S in G^* that corresponds to $\vec{p}$.

A graph G', similar to the graph G^* used here, was recently used in [4] to show that any stable matching in G' maps to a maximum-size popular matching M in G. However every maximum-size popular matching in G need not be obtained as a stable matching in G'. In the special case when preference lists are complete (i.e., G is $K_{|A|,|B|}$), all popular matchings in G can be realized as stable matchings in G'. The method used in [4] is similar to the method used in previous algorithms to compute maximum-size popular matchings [10, 12] – these show that there is no *popularity-improving* alternating path or cycle with respect to the matching returned. In contrast, our technique here is based on linear programming.

A min-cost popular half-integral popular matching has applications – consider the problem of assigning projects to students where each project can be split into two half-projects. Each half-project can be assigned to a distinct student and a student can be assigned two half-projects. A min-cost popular half-integral matching is a feasible assignment here that is popular and has the least cost. While fractional matchings, in general, may not be feasible in typical applications, half-integral matchings are more natural and suitable to applications.

Background. Algorithms for computing popular matchings [1] were first considered in the one-sided preference lists model where it is only vertices in A that have preferences and cast votes while vertices in B have no preferences. Popular matchings need not always exist in this model, however it was shown in [13] that popular fractional matchings always exist and using the description of $\mathcal{P}$, such a fractional matching can be found in polynomial time (via linear programming).

In the two-sided preference lists model, when preference lists have ties, $G = (A \cup B, E)$ need not always admit a popular matching and it is known that determining if G admits a popular matching or not is an NP-complete problem [2, 3]. When preference lists are strict, every stable matching is popular. The min-cost stable matching problem in an instance $G = (A \cup B, E)$ with strict preference lists is well-studied and descriptions of the stable matching polytope were given by Vande Vate [16], Rothblum [14], and Teo and Sethuraman [15].

We discuss preliminaries in Section 2. Section 3 describes the graph G^* and shows that every stable matching in G^* is a popular half-integral matching in G. Section 4 shows how every popular half-integral matching in G that is an extreme point of $\mathcal{Q}$ (the popular half-integral matching polytope) can be obtained as a stable matching in G^*.

2 Preliminaries

For any vertex $u \in A \cup B$ and neighbors v and w, we will use the following function to show u's preference for v vs w: $\mathsf{vote}_u(v, w) = 1$ if u prefers v to w, it is -1 if u prefers w to v, else (i.e., when $v = w$) it is 0. We will be using this function in the description of the popular fractional matching polytope $\mathcal{P}$.

Recall that a popular fractional matching is a point $\vec{x} = (x_e)_{e \in E}$ in the matching polytope of G such that $\Delta(\vec{x}, M) \geq 0$ for all matchings M in G. It will be convenient to assume that each vertex $u \in A \cup B$ is completely matched in every fractional matching $\vec{x}$ in G. So we will revise $\vec{x}$ so that each vertex u gets matched to an artificial last-resort neighbor $\ell(u)$ (which is placed at the bottom of u's preference list) with weight $(1 - \sum_{e \in E(u)} x_e)$, where the sum is over all the edges e incident on u.

For convenience, we will continue to use $\vec{x}$ to denote the revised $\vec{x}$ in $[0, 1]^{m+n}$. We use $\tilde{E}$ to denote the edge set $E \cup \{(u, \ell(u)) : u \in A \cup B\}$ and $\tilde{E}(u)$ is the set of edges in $\tilde{E}$ that are incident on u. The following simple description of $\mathcal{P}$ was given in [13]. In the constraints below, a variable α_u is associated with each $u \in A \cup B$ and not to last-resort neighbors.

$$\alpha_a + \alpha_b \geq \sum_{(a,b') \in \tilde{E}(a)} x_{(a,b')} \cdot \mathsf{vote}_a(b, b') + \sum_{(a',b) \in \tilde{E}(b)} x_{(a',b)} \cdot \mathsf{vote}_b(a, a') \quad \forall (a, b) \in \tilde{E}$$

$$\sum_{u \in A \cup B} \alpha_u = 0 \quad \text{and} \quad \sum_{e \in \tilde{E}(u)} x_e = 1 \quad \forall u \in A \cup B \quad \text{and} \quad x_e \geq 0 \quad \forall e \in \tilde{E}.$$

The constraints above arise as the dual to the maximum weight matching problem in the graph $\tilde{G}_x$ which is G augmented with last-resort neighbors and with edge set $\tilde{E}$, where the weight of an edge (a, b) is $\sum_{(a,b') \in \tilde{E}(a)} x_{(a,b')} \cdot \mathsf{vote}_a(b, b') + \sum_{(a',b) \in \tilde{E}(b)} x_{(a',b)} \cdot \mathsf{vote}_b(a, a')$. The constraint $\sum_{u \in A \cup B} \alpha_u = 0$ is equivalent to saying that the maximum weight of a matching in $\tilde{G}_x$ is 0, in other words, $\vec{x}$ is popular. We refer the reader to Section 3 of [13] for all the details.

For any fractional matching $\vec{x}$, if there exists $\vec{\alpha} = (\alpha_u)_{u \in A \cup B}$ such that $\vec{x}$ and $\vec{\alpha}$ satisfy the above constraints, then we say $\vec{x} \in \mathcal{P}$. The vector $\vec{\alpha}$ will be called a *witness* to $\vec{x}$'s popularity.

a_0	v_1			
a_1	b_1	v_1		
a_2	b_1	b_2		
u_1	v_1	v_2	b_0	
u_2	v_2	b_2	v_1	

b_0	u_1			
b_1	a_2	a_1		
b_2	a_2	u_2		
v_1	u_2	a_1	u_1	a_0
v_2	u_1	u_2		

Figure 1 The above table describes the preference lists of all the men and women in G. Here a_0 has a single neighbor v_1 while a_1's top choice is b_1, second choice is v_1 and so on for each vertex.

$\mathcal{P}$ is not integral. We now show an example of a graph G and a fractional matching $\vec{p} \in \mathcal{P}$, however $\vec{p}$ is not a convex combination of popular matchings. Let $A = \{a_0, a_1, a_2, u_1, u_2\}$, $B = \{b_0, b_1, b_2, v_1, v_2\}$, and the preference lists of vertices are described in Figure 1.

Consider the half-integral matching $\vec{p}$ which has $p_{(a_1,b_1)} = p_{(a_2,b_2)} = 1$ and $p_e = \frac{1}{2}$ for $e \in \{(u_1, v_1), (u_2, v_2), (u_1, v_2), (u_2, v_1)\}$. For any other edge e, we have $p_e = 0$. This fractional matching belongs to $\mathcal{P}$ by using the following α values: $\alpha_{a_0} = \alpha_{b_0} = 0$; $\alpha_{a_2} = \alpha_{b_1} = 1$; $\alpha_{a_1} = \alpha_{b_2} = -1$; and $\alpha_w = 0$ for $w \in \{u_1, u_2, v_1, v_2\}$.

There is only one way to express $\vec{p}$ as a convex combination of integral matchings, that is, $\vec{p} = (I(M_1) + I(M_2))/2$, where $M_1 = \{(a_1, b_1), (a_2, b_2), (u_1, v_1), (u_2, v_2)\}$ and $M_2 = \{(a_1, b_1), (a_2, b_2), (u_1, v_2), (u_2, v_1)\}$. We show below that neither M_1 nor M_2 is popular.

The matching $M_1' = \{(u_1, b_0), (a_1, v_1), (a_2, b_1), (u_2, v_2)\}$ is more popular than M_1 and the matching $M_2' = \{(a_0, v_1), (u_2, b_2), (a_2, b_1), (u_1, v_2)\}$ is more popular than M_2. Thus $\vec{p}$ is not in the convex hull of popular matchings in G.

The graph G'. Our input is a graph $G = (A \cup B, E)$ on n vertices and m edges. Note that there are no last-resort neighbors here – they were added only for the formulation of the polytope $\mathcal{P}$. Vertices in A and in B are usually referred to as men and women, respectively, and we follow the same convention here.

The construction of the following graph $G' = (A' \cup B', E')$, based on G, was shown in [4]. The set A' has two copies a_0 and a_1 of each man $a \in A$, the men in $\{a_0 : a \in A\}$ are called *level 0 men* of G' and those in $\{a_1 : a \in A\}$ are called *level 1 men* of G'. The set B' consists of all the women in B along with *dummy vertices* $\cup_{a \in A}\{d(a)\}$, where there is one dummy vertex per man in A. The preference lists of the vertices are as follows:

- each level 0 man a_0 has the same preference list as the corresponding man a in G except that the dummy vertex $d(a)$ occurs as his *least* preferred neighbor at the bottom of his preference list
- each level 1 man a_1 has the same preference list as the corresponding man a in G except that the dummy vertex $d(a)$ occurs as his *most* preferred neighbor at the top of his preference list
- each dummy vertex $d(a)$ has a_0 and a_1 as its neighbors: top choice is a_0, followed by a_1
- every woman $b \in B$ has the following preference list in G': all her level 1 neighbors (in the same order of preference as in G) followed by all her level 0 neighbors (in the same order of preference as in G).

We will be using this graph G' here; in fact, we will have two such graphs G' and G'' combining to form our new graph G^*. The graph G'' is analogous to the graph G' except that the roles of men and women (and also that of levels 0 and 1) are swapped here.

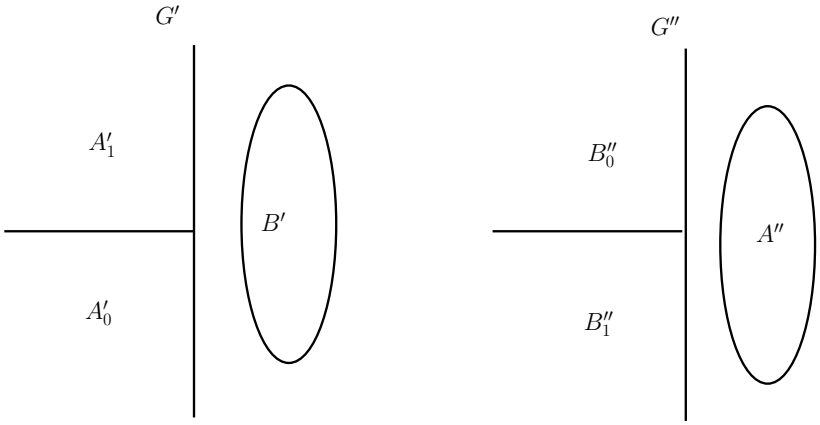


Figure 2 The graph G' on the left and the graph G'' on the right in G^*. For $i = 0, 1$, we use A'_i to refer to level i men in G' and we use B''_i to refer to level i women in G''.

3 The graph G^*

We define the graph G^* as follows: G^* consists of two vertex-disjoint subgraphs G' and G'' (see Figure 2). The graph G' was described in Section 2.

In the graph $G'' = (B'' \cup A'', E'')$, women are on the left side of G'' and men are on the right side – the set B'' has two copies b_0 and b_1 of each woman $b \in B$, the women in $\{b_0 : b \in B\}$ are called *level 0 women* of G'' and those in $\{b_1 : b \in B\}$ are called *level 1 women* of G''.

The set A'' consists of all the men in A along with new dummy vertices $\cup_{b \in B}\{d(b)\}$, where there is one dummy vertex per woman in B. The preference lists of the vertices are as follows:

- each level 0 woman b_0 has the same preference list as the corresponding woman b in G except that the dummy vertex $d(b)$ occurs as her *most* preferred neighbor at the top of her preference list
- each level 1 woman b_1 has the same preference list as the corresponding woman b in G except that the dummy vertex $d(b)$ occurs as her *least* preferred neighbor at the bottom of her preference list
- each dummy vertex $d(b)$ has only b_0 and b_1 as its neighbors: its top choice is b_1, followed by b_0
- every man $a \in A$ has the following preference list in G'': all his level 0 neighbors (in the same order of preference as in G) followed by all his level 1 neighbors (in the same order of preference as in G).

We want all stable matchings in G^* to be perfect matchings – note that all level 0 men in G' and all level 1 women in G'' will be matched in any stable matching in G^* since they are top-choice neighbors for their respective dummy neighbors. However the same cannot be said about level 1 men in G' and level 0 women in G''.

In order to take care of these vertices, we add the following "self-loop" edges to G^*: the edge (a_1, a) for each man a in A, where $a_1 \in A'_1$ and $a \in A''$, and the edge (b_0, b) for each woman b in B, where $b_0 \in B''_0$ and $b \in B'$. The vertex $a_1 \in A'_1$ regards $a \in A''$ as his worst ranked neighbor and similarly, $b_0 \in B''_0$ regards $b \in B'$ as her worst ranked neighbor.

For any man $a \in A''$, the vertex a_1 is in the middle of his preference list, sandwiched between all his level 0 neighbors and all his level 1 neighbors as shown in (1) below. More

precisely, a_1 is sandwiched between b''_0 and b'_1, where $b' > \cdots > b''$ is a's preference list in G. Thus b''_0 is a's worst level 0 neighbor and b'_1 is a's best level 1 neighbor.

$$a:\ b'_0 > \cdots > b''_0 > \underline{a_1} > b'_1 > \cdots > b''_1; \qquad b:\ a'_1 > \cdots > a''_1 > \underline{b_0} > a'_0 > \cdots > a''_0. \quad (1)$$

Similarly, for any woman $b \in B'$, the vertex b_0 is in the middle of her preference list, sandwiched between all her level 1 neighbors and all her level 0 neighbors as shown in (1). More precisely, b_0 is sandwiched between a''_1 and a'_0, where $a' > \cdots > a''$ is b's preference list in G. Using the fact that all stable matchings in G^* match the same set of vertices [8], it can be shown that every stable matching in G^* is perfect.

The function f. We now define a function f : {stable matchings in G^*} $\to$ {half-integral matchings in G}. Observe that every stable matching in G^* has to match all dummy vertices since each of these is a top-choice neighbor for someone. Thus out of a_0 and a_1 in A', only one is matched to a non-dummy neighbor and similarly, out of b_0 and b_1 in B'', only one is matched to a non-dummy neighbor.

Let S be any stable matching in G^*. By removing all self-loops that occur in S and those edges in S that contain a dummy vertex, the resulting matching is the union of two matchings S' and S'' in G. We define $f(S)$ to be $(I(S') + I(S''))/2$, where $I(M) \in \{0,1\}^m$ is the 0-1 edge incidence vector of M. So $f(S)$ is a valid half-integral matching in G.

▶ **Theorem 2.** *For any stable matching S in G^*, the half-integral matching $f(S)$ is popular in G.*

Proof. We are given a stable matching S in G^*. Recall that we pruned all edges that contain a dummy vertex and all self-loops from S to define $f(S)$. We now prune all dummy vertices, their partners in S, and self-loops from G^* also – let H^* denote the pruned graph G^*. Let H' denote the pruned subgraph G' and let H'' denote the pruned subgraph G''.

The men in the graph H' consist of one copy of each $a \in A$ – some of these are in level 0 and the rest are in level 1. The women in H' are exactly those in B. The women in H'' consist of one copy of each $b \in B$ – some of these are in level 0 and the rest are in level 1. The men in H'' are exactly those in A. Thus H' and H'' are two copies of the graph G.

Let S' be the pruned matching (resulting from S) restricted to H' and let S'' be the pruned matching (resulting from S) restricted to H''. Let $\tilde{A}'_i$ denote the set of level i men in H', for $i = 0, 1$ (see Figure 3). Let $\tilde{B}'_i$ consist of women matched in S' to men in $\tilde{A}'_i$, for $i = 0, 1$. Women unmatched in S' are added to $\tilde{B}'_1$.

Similarly, $\tilde{B}''_i$ consists of level i women in the H'' part of H^* and $\tilde{A}''_i$ denotes the set of men matched in S'' to women in $\tilde{B}''_i$, for $i = 0, 1$. Men unmatched in S'' are added to $\tilde{A}''_0$.

For each edge $e = (a, b) \in H'$, define the function $w'(e)$ as follows: $w'(e) = \mathsf{vote}_a(b, S'(a)) + \mathsf{vote}_b(a, S'(b))$. If $S'(u)$ is undefined for any vertex u, then $\mathsf{vote}_u(v, S'(u)) = 1$ for any neighbor v of u since every vertex prefers being matched to being unmatched. Note that if $(a, b) \in S'$ then $w'(e) = 0$.

Similarly, for each edge $e = (a, b) \in H''$, define the function $w''(e)$ as follows: $w''(e) = \mathsf{vote}_a(b, S''(a)) + \mathsf{vote}_b(a, S''(b))$. For any vertex u that is unmatched in S'', we take $\mathsf{vote}_u(v, S''(u)) = 1$, for any neighbor v of u. Note that $w'(e)$ and $w''(e)$ always take values in $\{-2, 0, 2\}$. Due to the stability of the matching S in G^*, the following observations hold:

- Every edge $e \in \tilde{A}'_1 \times \tilde{B}'_0$ has to satisfy $w'(e) = -2$. Similarly, every edge $e \in \tilde{A}''_1 \times \tilde{B}''_0$ has to satisfy $w''(e) = -2$.

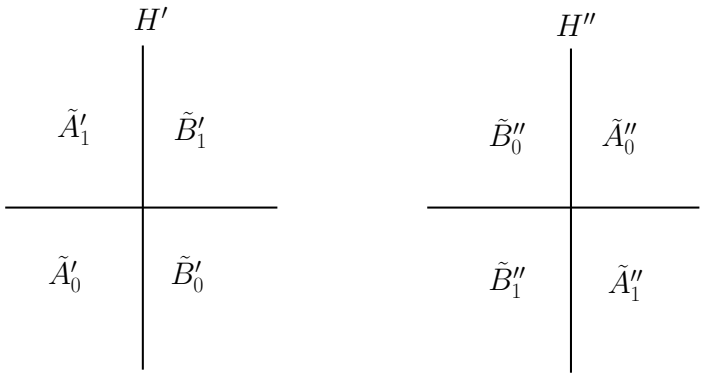


Figure 3 The graph H' on the left and the graph H'' on the right in the graph H^*.

Consider an edge (a_1, b) in $\tilde{A}'_1 \times \tilde{B}'_0$. It follows from the definition of preference lists of women in G' that the woman b prefers a_1 (a level 1 man) to her partner $S'(b)$ (a level 0 man). Since S is stable, it follows that a_1 prefers his partner $S'(a_1)$ to b. Moreover, a_0 prefers b to $S'(a_0) = d(a)$, since $d(a)$ is a_0's last choice. Thus b prefers her partner $S'(b)$ to a_0. So $\mathsf{vote}_a(b, S'(a)) = \mathsf{vote}_b(a, S'(b)) = -1$. A similar proof holds for any edge $e \in \tilde{A}''_1 \times \tilde{B}''_0$.

- Every edge e such that $w'(e) = 2$ has to be in $\tilde{A}'_0 \times \tilde{B}'_1$. Similarly, every edge e such that $w''(e) = 2$ has to be in $\tilde{A}''_0 \times \tilde{B}''_1$.

If e is an edge in H' such that $w'(e) = 2$, then $e \notin \tilde{A}'_i \times \tilde{B}'_i$ (for $i = 0, 1$) as such an edge would block S. We have already seen that any edge $e \in \tilde{A}'_1 \times \tilde{B}'_0$ satisfies $w'(e) = -2$. Thus any edge e such that $w'(e) = 2$ has to be in $\tilde{A}'_0 \times \tilde{B}'_1$. We can similarly show that any edge e in H'' such that $w''(e) = 2$ has to be in $\tilde{A}''_0 \times \tilde{B}''_1$.

We will now show that $f(S) \in \mathcal{P}$ by assigning appropriate α_u values for all u in $A \cup B$. We first define α'_u and α''_u:

- let $\alpha'_u = -1$ if $u \in \tilde{A}'_1 \cup \tilde{B}'_0$ and let $\alpha'_u = 1$ if $u \in \tilde{A}'_0 \cup \tilde{B}'_1$.
- let $\alpha''_u = -1$ if $u \in \tilde{A}''_1 \cup \tilde{B}''_0$ and let $\alpha''_u = 1$ if $u \in \tilde{A}''_0 \cup \tilde{B}''_1$.

The following is an immediate corollary of the above observations and the definitions of α'_u and α''_u: $\alpha'_a + \alpha'_b \geq w'(a, b)$ and $\alpha''_a + \alpha''_b \geq w''(a, b)$ for all edges (a, b). Also for any vertex u that is unmatched in S' and S'', we have $\alpha'_u + \alpha''_u = 0$.

Define $\alpha_u = (\alpha'_u + \alpha''_u)/2$ for all $u \in A \cup B$. Observe that $\sum_{u:A\cup B} \alpha_u = 0$. The above constraints imply that $(\alpha_u)_{u\in A\cup B}$ and the incidence vector of $f(S)$ satisfy the constraints of the polytope $\mathcal{P}$. Thus $f(S)$ is a popular half-integral matching. ◀

4 Constructing a stable matching in G^*

We showed in the previous section that f maps stable matchings in G^* to popular half-integral matchings in G. In fact, $f(S)$ is what we will call a *full* half-integral matching, i.e., for every vertex $u \in A \cup B$, either u is fully matched in $f(S)$ or it is fully unmatched in $f(S)$. Let $\vec{p} \in \{0, \frac{1}{2}, 1\}^m$ be a full half-integral matching that is popular. Since $\vec{p} \in \mathcal{P}$, there exists a witness $(\alpha_u)_{u\in A\cup B}$ to $\vec{p}$'s popularity. The following lemma will be useful to us.

▶ **Lemma 3.** *There exists a witness $(\alpha_u)_{u\in A\cup B}$ to $\vec{p}$'s popularity such that $\alpha_u \in \{\pm 1, 0\}$, for each $u \in A \cup B$.*

Proof. In order to show such a witness, we will consider the following linear program:

$$\text{minimize} \sum_{u \in A \cup B} \alpha_u \qquad \text{(LP1)}$$

subject to

$$\alpha_a + \alpha_b \;\geq\; \textstyle\sum_{(a,b') \in \tilde{E}(a)} p_{(a,b')} \cdot \mathsf{vote}_a(b, b') + \sum_{(a',b) \in \tilde{E}(b)} p_{(a',b)} \cdot \mathsf{vote}_b(a, a') \;\forall\, (a,b) \in \tilde{E}$$

Recall that $\tilde{E}$ is the set $E \cup \{(u, \ell(u)) : u \in A \cup B\}$, where $\ell(u)$ is the artificial last-resort neighbor of vertex u. In the above constraints, let us denote the right hand side quantity corresponding to edge e by $\mathsf{value}_p(e)$. Since $\vec{p}$ is a full half-integral matching, it is easy to see that $\mathsf{value}_p(e)$ is integral for all edges e.

Consider the polyhedron defined by the above constraints $N \cdot \vec{\alpha} \geq \vec{c}$, where N is the above $(m+n) \times n$ constraint matrix, $\vec{\alpha}$ is the column of unknowns α_u, for $u \in A \cup B$, and $\vec{c}$ is the column vector of $\mathsf{value}_p(\cdot)$ values. The top $m \times n$ sub-matrix of N is the edge-vertex incidence matrix U of the graph G and the bottom $n \times n$ matrix is the identity matrix I. Since the graph G is bipartite, the matrix U is totally unimodular and hence the matrix N is totally unimodular. Since $\vec{c}$ is an integral vector, it follows that all the vertices of $N \cdot \vec{\alpha} \geq \vec{c}$ are integral.

Thus there is an integral optimal solution to (LP1), call it $\vec{\alpha}^*$. We need to now show that $\vec{\alpha}^* \in \{\pm 1, 0\}^n$. It follows from the constraints corresponding to the edges $(u, \ell(u))$ that $\alpha^*_u \geq -1$ if u is matched in $\vec{p}$ and $\alpha^*_u \geq 0$ for u unmatched in $\vec{p}$. We now show the following claim.

▶ **Claim 4.** *Let $e = (a, b)$ be any edge such that $p_e > 0$. Then the constraint in (LP1) corresponding to e is tight, i.e., $\alpha^*_a + \alpha^*_b = \mathsf{value}_p(e)$.*

Proof. Consider the dual program of (LP1): it is the maximum weight matching problem in the graph G augmented with last-resort neighbors and with edge set $\tilde{E}$, where the weight of edge e is $\mathsf{value}_p(e)$. A maximum weight matching in this graph has weight 0 (because $\vec{p}$ is popular). Since $\Delta(\vec{p}, \vec{p}) = 0$, the fractional matching $\vec{p}$ is an optimal dual solution. It follows from complementary slackness conditions that if $p_{(a,b)} > 0$, then the constraint in (LP1) for edge (a, b) is tight. ◀

Observe that for any vertex u, there has to be an edge e incident on it with $p_e > 0$ and either $\mathsf{value}_p(e) = 0$ or $\mathsf{value}_p(e) = -1$ (the edge e between u and its worse partner v in $\vec{p}$). Using Claim 4 and the fact that $\alpha^*_v \geq -1$, we can now conclude that $\alpha^*_u \leq 1$. ◀

We will use the above lemma to show the following theorem in this section.

▶ **Theorem 5.** *Let $\vec{p} \in \{0, \frac{1}{2}, 1\}^m$ be a full half-integral matching that is popular. Then $\vec{p} = f(S)$ for some stable matching S in G^*.*

We will now build a stable matching S in the graph G^* such that $f(S) = \vec{p}$. For every edge $e = (a, b)$ such that $p_e > 0$, we need to decide which of the edges $(a_0, b), (a_1, b), (b_0, a), (b_1, a)$ will get included in S. In order to make this decision, we will build a graph H^*. The graph H^* consists of two copies H' and H'' of the input graph G.

Every vertex $u \in A \cup B$ gets assigned a level, denoted by $\mathsf{level}'(u)$, in H'. For $a \in A$, $\mathsf{level}'(a) = i$ fixes $a_i \in \{a_0, a_1\}$ to be the one that will be matched to a woman (i.e., a non-dummy vertex) in S. For $b \in B$, we say $\mathsf{level}'(b) = i$ to fix b getting matched to some level i man in H'. We will say u is in level i in H' to mean $\mathsf{level}'(u) = i$.

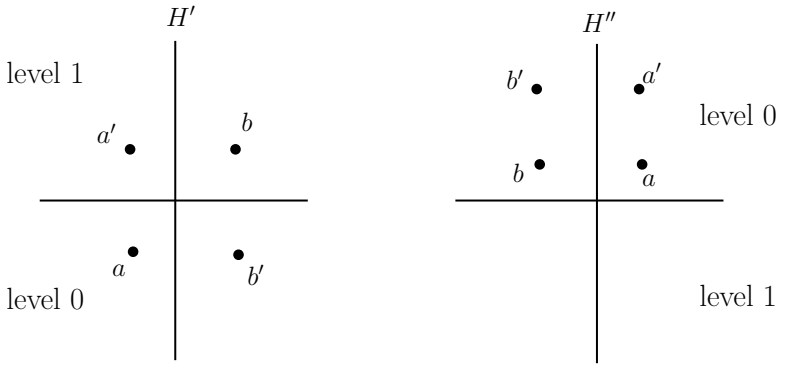


Figure 4 Since $\alpha^*_a = 1$ and $\alpha^*_{b'} = -1$, we have $\mathsf{level}'(a) = \mathsf{level}'(b') = 0$ and similarly, $\mathsf{level}''(a) = \mathsf{level}''(b') = 0$. Since $\alpha^*_b = \alpha^*_{a'} = 0$, we have $\mathsf{level}'(b) = \mathsf{level}'(a') = 1$ and similarly, $\mathsf{level}''(b) = \mathsf{level}''(a') = 0$. So we place a and b' in level 0 in both H and H' and we place a' and b in level 1 in H' and in level 0 in H''.

Similarly, every vertex $u \in A \cup B$ gets assigned a level, denoted by $\mathsf{level}''(u)$, in H''. For $b \in B$, $\mathsf{level}''(b) = j$ fixes $b_j \in \{b_0, b_1\}$ to be the one that will be matched to a man (i.e., a non-dummy vertex) in S. For $a \in A$, we say $\mathsf{level}''(a) = j$ to fix a getting matched to some level j woman in H''. We will say u is in level j in H'' to mean $\mathsf{level}''(u) = j$.

Since $\vec{p}$ is a full half-integral matching that is popular, we know from Lemma 3 that there exists a witness $\vec{\alpha}^* = (\alpha^*_u)_{u \in A \cup B}$ in $\{-1, 0, 1\}^n$ to the popularity of $\vec{p}$. We will use $\vec{\alpha}^*$ to fix $\mathsf{level}'(u)$ and $\mathsf{level}''(u)$ for each vertex u as follows.

- $\alpha^*_u = -1$: If $u \in A$ then $\mathsf{level}'(u) = \mathsf{level}''(u) = 1$. If $u \in B$ then $\mathsf{level}'(u) = \mathsf{level}''(u) = 0$.
- $\alpha^*_u = 1$: If $u \in A$ then $\mathsf{level}'(u) = \mathsf{level}''(u) = 0$. If $u \in B$ then $\mathsf{level}'(u) = \mathsf{level}''(u) = 1$.
- $\alpha^*_u = 0$: For all $u \in A \cup B$, $\mathsf{level}'(u) = 1$ and $\mathsf{level}''(u) = 0$.

As an example, consider the 4-cycle G on 2 men a, a' and 2 women b, b' where both a and a' prefer b to b' and both b and b' prefer a to a'. Let $\vec{p}$ be the half-integral matching with $p_e = 1/2$ for each edge e. This is popular and $\alpha^*_a = 1$, $\alpha^*_b = \alpha^*_{a'} = 0$, and $\alpha^*_{b'} = -1$ is a witness to $\vec{p}$'s popularity. Figure 4 shows how these vertices get placed in H' and in H''.

For any vertex u, let v and v' be its neighbors in G such that $\vec{p}$ has positive support on (u, v) and (u, v'). We will refer to v and v' as *partners* of u in $\vec{p}$. We need to show that either (i) $\mathsf{level}'(u) = \mathsf{level}'(v)$ and $\mathsf{level}''(u) = \mathsf{level}''(v')$, or (ii) $\mathsf{level}'(u) = \mathsf{level}'(v')$ and $\mathsf{level}''(u) = \mathsf{level}''(v)$. In other words, we need to show that u, v are level-compatible in one of H', H'' and u, v' are level-compatible in the other graph in H', H''.

We will now show that our allocation of levels to men and women based on their α^*-values ensures this. If $v = v'$ then $p_{(u,v)} = 1$ and the (tight) constraint for edge (u, v) in the description of $\mathcal{P}$ is $\alpha^*_u + \alpha^*_v = 0$. Thus (α^*_u, α^*_v) has to be one of $(1, -1), (0, 0), (-1, 1)$: in all three cases we have level-compatibility in both H' and H''. The following lemma shows that even when u has two distinct partners v and v' in $\vec{p}$, there is level-compatibility.

▶ **Lemma 6.** *Every vertex that has two distinct partners in $\vec{p}$ is level-compatible in H' with one partner and is level-compatible in H'' with another partner.*

Proof. We will show this lemma for any vertex $b \in B$. An analogous proof holds for any vertex in A. Let $a \neq a'$ be the partners of b in $\vec{p}$ and let b prefer a to a'. We know that $p_{(a,b)} = p_{(a',b)} = 1/2$. Since $\vec{p}$ is a full half-integral matching, a (similarly, a') has another neighbor $r(a)$ (resp., $r(a')$) with positive support in $\vec{p}$. We have four cases depending on how a and a' rank b versus $r(a)$ and $r(a')$, respectively.

1. If both a and a' prefer $r(a)$ and $r(a')$ respectively to b, then $\mathsf{value}_p(a,b) = -\frac{1}{2}+\frac{1}{2} = 0$ and $\mathsf{value}_p(a',b) = -\frac{1}{2}-\frac{1}{2} = -1$. By Claim 4, we know that $\alpha^*_a + \alpha^*_b = 0$ and $\alpha^*_{a'} + \alpha^*_b = -1$. So $(\alpha^*_a, \alpha^*_b, \alpha^*_{a'})$ is either $(1,-1,0)$ or $(0,0,-1)$.
   - In the former case $\mathsf{level}'(a) = \mathsf{level}'(b) = 0$ and $\mathsf{level}''(a') = \mathsf{level}''(b) = 0$.
   - In the latter case $\mathsf{level}'(a') = \mathsf{level}'(b) = 1$ and $\mathsf{level}''(a) = \mathsf{level}''(b) = 0$.
2. If both a and a' prefer b to $r(a)$ and $r(a')$ respectively, then $\mathsf{value}_p(a,b) = \frac{1}{2}+\frac{1}{2} = 1$ and $\mathsf{value}_p(a',b) = \frac{1}{2}-\frac{1}{2} = 0$. By Claim 4, we know that $\alpha^*_a + \alpha^*_b = 1$ and $\alpha^*_{a'} + \alpha^*_b = 0$. So $(\alpha^*_a, \alpha^*_b, \alpha^*_{a'})$ is either $(0,1,-1)$ or $(1,0,0)$.
   - In the former case $\mathsf{level}'(a) = \mathsf{level}'(b) = 1$ and $\mathsf{level}''(a') = \mathsf{level}''(b) = 1$.
   - In the latter case $\mathsf{level}'(a') = \mathsf{level}'(b) = 1$ and $\mathsf{level}''(a) = \mathsf{level}''(b) = 0$.
3. If a prefers b to $r(a)$ while a' prefers $r(a')$ to b, then $\mathsf{value}_p(a,b) = \frac{1}{2}+\frac{1}{2} = 1$ and $\mathsf{value}_p(a',b) = -\frac{1}{2}-\frac{1}{2} = -1$. By Claim 4, we know that $\alpha^*_a + \alpha^*_b = 1$ and $\alpha^*_{a'} + \alpha^*_b = -1$. So $(\alpha^*_a, \alpha^*_b, \alpha^*_{a'})$ is $(1,0,-1)$.
   - Here $\mathsf{level}'(a') = \mathsf{level}'(b) = 1$ and $\mathsf{level}''(a) = \mathsf{level}''(b) = 0$.
4. If a prefers $r(a)$ to b while a' prefers b to $r(a')$, then $\mathsf{value}_p(a,b) = -\frac{1}{2}+\frac{1}{2} = 0$ and $\mathsf{value}_p(a',b) = \frac{1}{2}-\frac{1}{2} = 0$. By Claim 4, we know that $\alpha^*_a + \alpha^*_b = 0$ and $\alpha^*_{a'} + \alpha^*_b = 0$. So $(\alpha^*_a, \alpha^*_b, \alpha^*_{a'})$ is $(1,-1,1)$ or $(0,0,0)$ or $(-1,1,-1)$.
   - In the first case, all three vertices a, b, and a' are in level 0 in both H' and H''.
   - In the second case, all three vertices are in level 1 in H' and in level 0 in H''.
   - In the third case, all three vertices are in level 1 in both H' and H''. ◀

For any vertex u with partners v and v' in $\vec{p}$, where u prefers v to v', we call v the *better partner* of u and v' the *worse partner* of u. If $p_{(a,b)} = 1$ for some edge (a,b), then we regard a as both the better partner and the worse partner of b.

We are now ready to describe the construction of our matching S. We give the following two pairing rules for any $b \in B$ (let a be b's better partner and a' be b's worse partner):

1. if $\alpha^*_b \in \{\pm 1\}$ then pair b with a in H' and with a' in H''.
2. if $\alpha^*_b = 0$ then pair b with a' in H' and with a in H''.

More precisely, if $\alpha^*_b = -1$ then we include (a_0, b) and (b_0, a') in S; if $\alpha^*_b = 1$ then we include (a_1, b) and (b_1, a') in S; and if $\alpha^*_b = 0$ then we include (a'_1, b) and (b_0, a) in S.

Note that the above rules for pairing vertices follow from the proof of Lemma 6. A woman b with $\alpha^*_b = -1$ (similarly, $\alpha^*_b = 1$) is level-compatible with her better partner in level 0 (resp., level 1) in H' and with her worse partner in level 0 (resp., level 1) in H''. Similarly, if $\alpha^*_b = 0$ then b is level-compatible with her worse partner in level 1 in H' and with her better partner in level 0 in H''.

Thus level-compatibility unambiguously fixes for us in which of H, H' a vertex gets paired with which partner till we are left with a set T of vertices forming a cycle: each vertex in T has both its partners in T, and all these vertices are in the same level in both H' and H''. We again know from the proof of Lemma 6 that this happens only when $(\alpha^*_a, \alpha^*_b) \in \{(1,-1),(0,0),(-1,1)\}$ for each edge (a,b) in this cycle. The cycle can be resolved as per the two rules above (which is what our algorithm for constructing S does). Thus rule 1 and rule 2 given above always work.

As the last step, we add the dummy vertices to H' and H''. We also add the *inactive* men and women (the ones who will get matched to dummy vertices in S). We now add to S the edges $(a_j, d(a))$ for all inactive men a_j and similarly, the edges $(b_j, d(b))$ for all inactive women b_j. We also add self-loops to match each unmatched vertex with its copy on the other side, i.e., we add the edges (a_1, a) for each $a \in A$ that is unmatched in $\vec{p}$ and the edges (b_0, b)

for each $b \in B$ that is unmatched in $\vec{p}$. Thus the final matching S is a perfect matching in the graph G^* and it follows from the construction of S that $f(S) = \vec{p}$.

In order to prove that the matching S is stable in G^*, we show in Lemmas 7 and 8 that S has no blocking edge in G'. We can similarly show that S admits no blocking edge in G''. Regarding the other edges in G^*, no self-loop (a_1, a) or (b_0, b) can be a blocking edge since a is the least preferred neighbor of a_1 and similarly, b is the least preferred neighbor of b_0. Similarly, since the dummy vertex $d(a)$ is the least preferred neighbor of a_0 and since a_1 is the least preferred neighbor of $d(a)$, no edge $(a_i, d(a))$ can block S. It is the same with edges $(b_i, d(b))$, for $i = 0, 1$. Hence S is a stable matching in G^* and Theorem 5 follows.

▶ **Lemma 7.** *Let $a \in A$ be in level 0 in H' and b be any neighbor of a in G. Neither edge (a_0, b) nor edge (a_1, b) in G' can block S.*

Proof. The following are the three cases that we need to consider here and show that none is a blocking edge to S:

1. the edge (a_1, b),
2. the edge (a_0, b) where b is in level 1 in H',
3. the edge (a_0, b) where b is in level 0 in H'.

Consider Case 1. Since a is in level 0 in H', the vertex a_1 is matched to $d(a)$ in S. Since $d(a)$ is a_1's most preferred neighbor, it follows that the edge (a_1, b) cannot block S for any neighbor b.

Consider Case 2. The woman b is in level 1 and this implies that $S(b)$ is a level 1 vertex in H'. Since b prefers any level 1 neighbor to a level 0 neighbor in G', it follows that b prefers $S(b)$ to a_0, thus (a_0, b) cannot block S.

Consider Case 3. Since both a and b are in level 0 in H', we have $\alpha^*_a = 1$ and $\alpha^*_b = -1$. These α^*-values and $p_{(a,b)}$ satisfy the constraint corresponding to edge (a, b) in the description of the popular matching polytope $\mathcal{P}$. Thus we have $0 \geq \mathsf{value}_p(a, b)$, where $\mathsf{value}_p(a, b)$ is the right hand side of the constraint for (a, b) in $\mathcal{P}$. The following sub-cases can occur here (since $\mathsf{value}_p(a, b) \leq 0$):

- **(i)** both the partners of a are better than b or both the partners of b are better than a
- **(ii)** $p_{(a,b)} = 1/2$ and either a regards its other partner better than b or vice-versa
- **(iii)** a has one partner better than b and the other worse than b and similarly, b has one partner better than a and the other worse than a

Sub-case (i) is straightforward and it is easy to see that (a_0, b) does not block S here. In sub-cases (ii) and (iii), we know that a woman b with $\alpha^*_b = -1$ gets matched to her better partner in H'. Thus in sub-case (ii) either b is matched to a (if a is b's better partner) or to a partner that b prefers to a. Similarly, in sub-case (iii), b gets matched to a neighbor that she prefers to a, thus (a_0, b) does not block S in any of these cases. This completes the proof of Lemma 7. ◀

▶ **Lemma 8.** *Let $a \in A$ be in level 1 in H' and b be any neighbor of a in G. Neither edge (a_0, b) nor edge (a_1, b) in G' can block S.*

Proof. The following are the three cases that we need to consider here and show that none is a blocking edge to S:

1. the edge (a_0, b),
2. the edge (a_1, b) where b is in level 0 in H',
3. the edge (a_1, b) where b is in level 1 in H'.

Consider Case 1. When b is in level 1 in H', she is matched to a level 1 man; since b prefers any level 1 neighbor to a level 0 neighbor in G', it follows that b prefers $S(b)$ to a_0, thus (a_0, b) cannot block S.

Let us consider the case when b is in level 0 in H'. So $\alpha_b^* = -1$. Since a is in level 1 in H', we have either $\alpha_a^* = -1$ or $\alpha_a^* = 0$. So $\mathsf{value}_p(a, b) \leq -1$. Hence b prefers her better partner to a and since b satisfies $\alpha_b^* = -1$, she gets matched to her better partner in H'. Thus (a_0, b) does not block S.

Consider Case 2. We will show that a_1 prefers his partner $S(a_1)$ to b. Either (i) $\alpha_a^* = -1$ in which case $\mathsf{value}_p(a, b) \leq -2$ or (ii) $\alpha_a^* = 0$ in which case $\mathsf{value}_p(a, b) \leq -1$.

In case (i), $\mathsf{vote}_a(b, S(a_1)) = -1$ and so a prefers $S(a_1)$ to b. In case (ii), $\mathsf{vote}_a(b, S(a_1)) \leq 0$ and so a prefers his better partner in $\vec{p}$ to b. It follows from the proof of Lemma 6 that if $\alpha_a^* = 0$, then the man a is matched to his better partner in H'. Thus (a_1, b) does not block S in either case.

Consider Case 3. There are four sub-cases here based on possible values of (α_a^*, α_b^*): (i) $(\alpha_a^*, \alpha_b^*) = (-1, 1)$, (ii) $(\alpha_a^*, \alpha_b^*) = (-1, 0)$, (iii) $(\alpha_a^*, \alpha_b^*) = (0, 1)$, and (iv) $(\alpha_a^*, \alpha_b^*) = (0, 0)$.

- Cases (i) and (iv) are analogous to case 3 in the proof of Lemma 7 since $\mathsf{value}_p(a, b)$ is at most 0 in both these cases and a similar proof holds here for both these cases.
- In case (ii) above, we have $\mathsf{value}_p(a, b) \leq -1$. So either (I) a prefers both his partners in $\vec{p}$ to b or vice-versa, in which case (a_1, b) does not block S or (II) $p_{(a,b)} = 1/2$ and both a and b prefer their other partners in $\vec{p}$ to each other, in which case $(a_1, b) \in S$.
- In case (iii) above, we know that both a and b get paired to their respective better partners in H' (since $\alpha_a^* = 0$ and $\alpha_b^* = 1$). We have $\mathsf{value}_p(a, b) \leq 1$ here. So either (I) a prefers its better partner in $\vec{p}$ to b or vice-versa (in which case (a_1, b) does not block S) or (II) $p_{(a,b)} = 1/2$ and both a and b prefer each other to their other partners in $\vec{p}$, in which case $(a_1, b) \in S$. Thus (a_1, b) does not block S in any of these cases. ◀

Thus we have shown that f is a surjective map from the set of stable matchings in G^* to the set of full half-integral matchings in G that are popular. It can be shown that if $\vec{p}$ is a popular half-integral matching that is *not* full, then the edge incidence vector of $\vec{p}$ is a convex combination of the edge incidence vectors of popular half-integral matchings that are full. Hence the extreme points of the convex hull $\mathcal{Q}$ of popular half-integral matchings are the full ones. Thus the description of $\mathcal{Q}$ can be obtained in a straightforward manner from the description of the stable matching polytope of G^*.

We have shown the following theorem.

▶ **Theorem 9.** *A min-cost popular half-integral matching in $G = (A \cup B, E)$ with strict preference lists and cost function $c : E \to \mathbb{Q}$ can be computed in polynomial time.*

Conclusions. We gave a simple description of the convex hull of popular half-integral matchings in a stable marriage instance $G = (A \cup B, E)$ with strict preference lists. This allowed us to solve the min-cost popular half-integral matching problem in G in polynomial time. The main open problem here is to settle the complexity of the min-cost popular matching in G.

Acknowledgments. Thanks to Naveen Garg for useful discussions on this problem. Thanks to the reviewers for their helpful comments.

References

1. D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.
2. P. Biró, R. W. Irving, and D. F. Manlove. Popular matchings in the marriage and roommates problems. In *Proceedings of 7th International Conference on Algorithms and Complexity (CIAC)*, pages 97–108, 2010.
3. Á. Cseh, C.-C. Huang, and T. Kavitha. Popular matchings with two-sided preferences and one-sided ties. In *Proceedings of 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 367–379, 2015.
4. Á. Cseh and T. Kavitha. Popular edges and dominant matchings. To appear in the Proceedings of the 18th Conference on Integer Programming and Combinatorial Optimization (IPCO), 2016.
5. T. Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45:233–284, 1992.
6. T. Feder. Network flow and 2-satisfiability. *Algorithmica*, 11(3):291–319, 1994.
7. D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.
8. D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11:223–232, 1985.
9. P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Sciences*, 20:166–173, 1975.
10. C.-C. Huang and T. Kavitha. Popular matchings in the stable marriage problem. *Information and Computation*, 222:180–194, 2013.
11. R. W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM*, 38(3):532–543, 1987.
12. T. Kavitha. A size-popularity tradeoff in the stable marriage problem. *SIAM Journal on Computing*, 43(1):52–71, 2014.
13. T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412:2679–2690, 2011.
14. U. Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54:57–67, 1992.
15. C.-P. Teo and J. Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23(4):874–891, 1998.
16. J. H. Vande Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153, 1989.

Voronoi Choice Games*

Meena Boppana[†1], Rani Hod[‡2], Michael Mitzenmacher[§3], and Tom Morgan[¶4]

1 **Harvard University, Cambridge, MA, USA**
boppana@college.harvard.edu
2 **Department of Mathematics, Harvard University, Cambridge, MA, USA**
rani.hod@gmail.com
3 **SEAS, Harvard University, Cambridge, MA, USA**
michaelm@eecs.harvard.edu
4 **SEAS, Harvard University, Cambridge, MA, USA**
tdmorgan@seas.harvard.edu

Abstract

We study novel variations of Voronoi games and associated random processes that we call *Voronoi choice games*. These games provide a rich framework for studying questions regarding the power of small numbers of choices in multi-player, competitive scenarios, and they further lead to many interesting, non-trivial random processes that appear worthy of study.

As an example of the type of problem we study, suppose a group of n miners (or players) are staking land claims through the following process: each miner has m associated points independently and uniformly distributed on an underlying space (such as the unit circle, the unit square, or the unit torus), so the kth miner will have associated points $p_{k1}, p_{k2}, \ldots, p_{km}$. We generally here think of m as being a small constant, such as 2. Each miner chooses one of these points as the base point for their claim. Each miner obtains mining rights for the area of the square that is closest to their chosen base; that is, they obtain the Voronoi cell corresponding to their chosen point in the Voronoi diagram of the n chosen points. Each player's goal is simply to maximize the amount of land under their control. What can we say about the players' strategy and the equilibria of such games?

In our main result, we derive bounds on the expected number of pure Nash equilibria for a variation of the 1-dimensional game on the circle where a player owns the arc starting from their point and moving clockwise to the next point. This result uses interesting properties of random arc lengths on circles, and demonstrates the challenges in analyzing these kinds of problems. We also provide several other related results. In particular, for the 1-dimensional game on the circle, we show that a pure Nash equilibrium always exists when each player owns the part of the circle nearest to their point, but it is NP-hard to determine whether a pure Nash equilibrium exists in the variant when each player owns the arc starting from their point clockwise to the next point. This last result, in part, motivates our examination of the random setting.

1998 ACM Subject Classification F.2.m Miscellaneous

Keywords and phrases Voronoi games, correlated equilibria, power of two choices, Hotelling model

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.23

* A full version of the paper is available on the arXiv [5].

† Meena Boppana was supported in part by a PRISE summer research fellowship.
‡ Rani Hod was supported by the Center of Mathematical Sciences and Applications at Harvard University.
§ Michael Mitzenmacher was supported in part by NSF grants CCF-1320321, CNS-1228598, IIS-0964473, and CCF-0915922; part of his work was done while visiting Microsoft Research, New England.
¶ Tom Morgan was supported in part by NSF grants CCF-1320231 and CCF-0915922.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 23; pp. 23:1–23:13

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

1 Introduction

Consider the following prototypical problem: a group of miners are staking land claims. The kth miner – or player – has m associated points $p_{k1}, p_{k2}, \ldots, p_{km}$ in the unit torus (which is the unit square with wraparound at the boundaries, providing symmetry). Each miner via some process will choose exactly one of their m points as the base for their claim. The resulting n points yield a Voronoi diagram, and each miner obtains their corresponding Voronoi cell. Each player's goal is simply to maximize the amount of land under their control. We wish to study player behavior in this and similar games, focusing on equilibria.

As another application, political candidates can often be mapped according to their political views into a small-dimensional space; e.g., American candidates are often viewed as being points in a two-dimensional space, measuring how liberal/conservative they are on economic issues in one dimension and social issues on the other. Suppose parties must choose a candidate simultaneously, and their probability of winning is increasing in the area of the political space closest to their point. Again, the goal in this case is to maximize the corresponding area in a Voronoi diagram.

There are numerous variations one can construct from this setting. Most naturally, if the players are (lazy) security guards instead of miners, who have to patrol the area closest to their chosen base, their goal might be to minimize the area under their purview. Other alternatives stem from variations such as whether player choices are simultaneous or sequential, how the points for players are chosen, the underlying metric space, the type of equilibrium sought, and the utility function used to evaluate the final outcome.

However, the variations share the following fundamental features. There are n players, with the kth player having m_k associated points in some metric space. (We will focus on $m_k = m$ for a fixed m for all players.) Each player will have to choose to adopt one of their available points. A Voronoi diagram is then constructed, and each player is then associated with the corresponding area in the diagram. We refer to this general setting as *Voronoi choice games*. We discuss below how Voronoi choice games differ from similar recent work, but the key point is in the problems we study different players have different available choices; this asymmetry creates new problems and requires distinct methods.

We are particularly interested in the setting where each player's points are chosen uniformly at random from the underlying space. While uniform random points are not motivated by practice, the framework leads to an interesting and, from the standpoint of probabilistic analysis and geometry, very natural class of games. Our work suggests many potential connections, to work on Voronoi diagrams for random point sets, and to work on balanced allocations (or "the power of two choices"), where choice is used to improve load balancing. Moreover, looking at the setting of uniform random points gives us the opportunity to understand the nature of these games at a high level; specifically, do most instances have no pure Nash equilibrium, or could they have exponentially many possible pure Nash equilibria?

In general, however, we find that results for these types of problems seem very challenging. In our main result, we limit ourselves to the setting where each player has m associated points chosen uniformly at random from the unit circle, and each player owns the arc starting from their point clockwise to the next point – that is, the distance is unidirectional around the circle. We derive bounds on the expected number of pure Nash equilibria. Even in this simple setting, our result is quite technical, requiring a careful analysis based on interesting properties of distributions of random arcs on a circle. This appears, however, to be the "easiest" interesting version of the problem; currently, higher-dimensional Voronoi diagrams

are beyond our reach. However, our work suggests that further results are likely to involve interesting mathematics.

The random case of this specific version of the problem is also motivated by the following results. We show it is NP-hard to determine whether a pure Nash equilibrium exists when a player owns the arc starting from their point clockwise to the next point for $m \geq 4$, nearly resolving the worst case. Further, for the different setting when a player owns an arc of the circle corresponding to the standard Voronoi diagrams, that is a player own all points nearest to their point, we show a Nash equilibrium always exists (as long as all the possible choices for the players are distinct).

While other similar Voronoi game models have been introduced previously, our primary novelty is to introduce this natural type of asymmetric "choice" into these types of games. We believe this addition provides a rich framework with many interesting combinatorial, geometric, and game theoretic problems, as we describe throughout the paper. As such, we leave many natural open questions.

1.1 Related Work

The classical foundations for problems of this type can be found in the work by Hotelling [11], who studied the setting of two vendors who had to determine where to place their businesses along a line, corresponding to the main street in a town, with the assumption of uniformly distributed customers who would walk to the nearer vendor. Hotelling games have been considered for example in work on regret minimization and the price of anarchy, where the model studied players choosing points on a general graph instead of on the line as in the original model [4]. Recent work has also shown that for a Hotelling game on a given graph, once there are sufficiently many players a pure Nash equilibrium always exists [8]. A useful survey on economic location-based models is provided by Gabszewicz and Thisse [9].

Other variations of Voronoi games have appeared in the literature. More recent work refers to these generally as *competitive location games*; see for example [7, 14, 18, 13], which discuss Voronoi games on graphs, for additional references.

Our setting appears different from previous work, in that it focuses on players with *limited* sets of choices that *vary* among the players. Our starting point was aiming to build connections between Voronoi games and random processes based on "the power of two choices" [3, 15, 6, 1]. While in our games, each player has a limited (typically constant) set of distinct points to choose from, in previous work generally *all players* could choose from *any point* in the universe of possible choices. In economic terms, in relation to the Hotelling model, our work models that different businesses may have available a limited number of differing locations where they may establish their business. For example, businesses may have optioned the right to set up a franchise at specific locations in advance, and must then choose which location to actually build. While they could know the options available to other competing franchises, they may have to decide where to build without knowing the choices made by competitors. In other situations, it may be possible for franchises to move (at some cost) to an alternative location. We emphasize that our model is very different than previously studied symmetric versions of the game; we do not recover earlier results, and earlier results do not appear to apply once asymmetry is introduced.

1.2 Models

Before beginning, we explain the general class of games we are interested in. We refer to the following as the *k-D Simultaneous Voronoi Game*:

- Each of the n players has m associated points from the k-dimensional unit torus $[0,1]^k$. We assume that all players know about all of the possible points that can be chosen by every player (it is a game of complete information).
- The n players must simultaneously choose one of their m associated points.
- A Voronoi diagram is constructed for the n chosen points, and each player receives utility equal to the volume of its point's Voronoi cell in the maximization variation of the game. (In the minimization version, the utility could be the negation of the corresponding volume.)

The easiest version to think about is the 1-D version; each player chooses from m points on the unit circle, and after their choice they own an arc of the circle corresponding to all points closest to their chosen point. If each player tries to maximize their arc length, then the utility of a player is the length of their arc. (Or, if each player tries to minimize their arc length, the negation of the arc length is the utility.) On the unit circle, there is another variant that we refer to as the *One Way 1-D Simultaneous Voronoi Game*, in which a player owns the arc starting from their point and continuing in a clockwise direction until the next chosen point. Such a variation is quite natural in one dimension; it corresponds to assigning a "direction" to the unit circle. This variation is chiefly motivated by our connections to the power-of-two choices. In particular, it resembles the distributed hashing scheme of [6] in which peers correspond to points on a circle and keys are mapped to the closest peer in one direction along the circle.

Our contributions include highlighting differences between the 1-D problem and the One Way 1-D problem, showing that in this case a small difference in the model subtlety leads to large differences in the behavior with respect to equilibria. Indeed, as we explain, we believe the One Way 1-D problem potentially offers more insight into the behavior of the k-D Simultaneous Voronoi Game for $k \geq 2$ with respect to pure Nash equilibria.

We focus on analyzing the equilibria of these games. The most common equilibrium to study is the Nash equilibrium [16], in which each player has a random distribution on strategies such that no player can improve their expected utility by changing their distribution. While Nash's results imply the Voronoi games above all have Nash equilibria, we do not determine the complexity of finding Nash equilibria for these games; this is left as an open question. We here focus on pure Nash equilibrium. A pure Nash equilibrium is a Nash equilibrium in which each player's distribution has a support of size one. In other words, each player picks a single strategy to play and, given the other players' strategies, no player can improve their utility by choosing a different strategy. Unlike the Nash and correlated equilibria, a pure Nash equilibrium is not guaranteed to exist.

Pure Nash equilibria can be viewed as a setting where each player can choose to switch to any of their adopted points at any time. The question is then what are the stable states, where no player individually has the incentive to switch their adopted point. These stable states correspond to pure Nash equilibria, and may not even exist. A natural question is whether simple local dynamics – such as myopic best response, where at each time step some subset of players decides whether or not to switch the point it has adopted – reach a stable state quickly. To motivate our study of the random case, we examine the computational complexity of determining the existence of stable states in the 1-D Simultaneous Voronoi Game and One Way 1-D Simultaneous Voronoi Game in Section 2. For the former, we show (making use of known techniques) that a pure Nash equilibrium always exists; for the latter, we show that determining whether a pure Nash equilibrium exists is NP-complete.

In Section 3 we consider the existence of a pure Nash equilibrium for the Randomized One Way 1-D Simultaneous Voronoi Game, where each players possible choices for points

are selected uniformly at random from the unit circle. Here we bound the expected number of pure Nash equilibria, through a careful analysis based on properties of distributions of random arcs on a circle.

We also note that we have some results for another type of equilibrium, known as the correlated equilibrium. Whereas Nash equilibria have the players independently choosing their strategies, a correlated equilibrium allows the players' random distributions to be correlated (for example, by an external party). The stability requirement is then that given knowledge only of the overall distribution of outcomes and their own randomly chosen strategy, a player cannot improve their expected utility by deviating from their given strategy distribution [2]. Since a Nash equilibrium is a special case of a correlated equilibrium, a correlated equilibrium for the above games must exist. We discuss the computational complexity of finding a correlated equilibrium for the k-D Simultaneous Voronoi Game in Section 4.

We provide additional results in the full paper [5], including an empirical investigation of the probability that myopic best response will find a stable state in the Randomized One Way 1-D Simultaneous Voronoi Game and Randomized 2-D Simultaneous Voronoi Game, and several related conjectures related to the Randomized One Way 1-D Simultaneous Voronoi Game.

2 Pure Nash Equilibria

In this section, we show a fundamental difference between the One Way 1-D Simultaneous Voronoi Game and the 1-D Simultaneous Voronoi Game. Recall that for these problems each of the n players has a choice of m points on the unit circle; all players simultaneously choose one of their m points. The utility for the One Way variation of given player is equal to the distance to the nearest chosen point clockwise from its chosen point, while for the standard variation the utility is the size of the Voronoi cell (in this case, an arc).

We show the standard Voronoi variation always has at least one pure Nash equilibrium (for any number of choices per player), while it is NP-hard to determine if the maximization version of the One Way variation has a pure Nash equilibrium. We also suggest the implications of these results for the higher dimensional setting.

2.1 Existence of Pure Nash Equilibria in the 1-D Simultaneous Voronoi Game

In the argument that follows we assume the choices of points are distinct. The analysis can be easily modified for the case where multiple players can choose the same point if ownership of that point is determined by a fixed preference order (and other players have zero utility). However, if players choosing the same point share utility, then the theorem does not hold, as shown in [14].

▶ **Theorem 1.** *A pure Nash equilibrium for the maximization and minimization versions of the 1-D Simultaneous Voronoi Game exists for any set of points.*

Proof. We follow an approach utilized in [7, Lemma 4]. We define a natural total ordering on multi-sets of numbers A and B. For any two such multi-sets A and B, if $|A| < |B|$ we have $A \succ B$. When $|A| = |B|$, we have $A \succ B$ if $\max A > \max B$. If $\max A = \max B$, then let A' be A with one copy of the value $\max A$ removed, and similarly for B'; then $A \succ B$ also when $\max A = \max B$ and $A' \succ B'$.

Now consider a collection of choices in the 1-D Simultaneous Voronoi Game, and let $A = \{a_0, a_1, \ldots, a_{n-1}\}$ be the corresponding arc lengths, starting from some chosen point

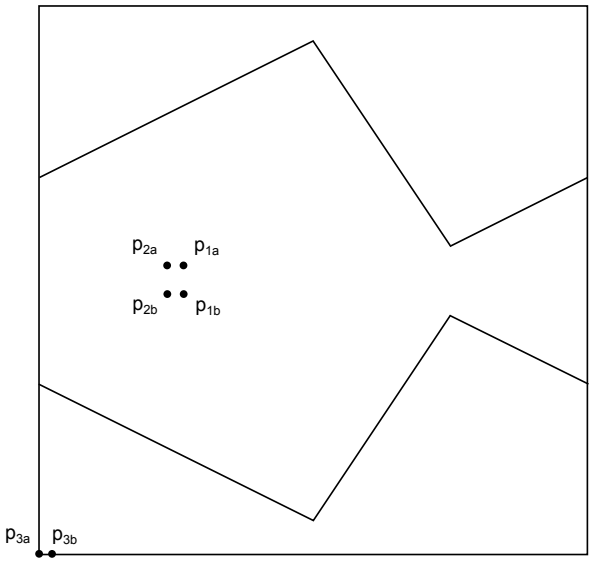


Figure 1 This is an example of a 2-D Simultaneous Voronoi Game in which no pure Nash equilibria exists. Player 1 has points $(p_{1a}, p_{1b}) = ((1/4 + \varepsilon^2, 1/2 + \varepsilon), (1/4 + \varepsilon^2, 1/2 - \varepsilon))$, player 2 has points $(p_{2a}, p_{2b}) = ((1/4 - \varepsilon^2, 1/2 + \varepsilon), (1/4 - \varepsilon^2, 1/2 - \varepsilon))$, and player 3 has points $(p_{3a}, p_{3b}) = ((0, 0), (\varepsilon, 0))$. The Voronoi diagram shown is constructed from the points $(1/4, 1/2)$ and $(0, 0)$.

and then in clockwise order around the circle, induced by the choice of points. Each player's payoff is given by a value of the form $(a_i + a_{i+1})/2$ for a suitable value of i. (One player's payoff is $(a_{n-1} + a_0)/2$.) Consider the maximization version of the game. If some player has a move that improves their utility, let them make that move. Without loss of generality, suppose this player's payoff was given by $(a_i + a_{i+1})/2$, and it moves somewhere on the arc with length a_j. Note this means $a_j > a_i + a_{i+1}$. We see that the arc lengths A' are those of A but with a_i, a_{i+1}, and a_j replaced by $a_i + a_{i+1}, x$, and y where $x + y = a_j$. Hence $A \succ A'$, so after a finite number of moves, this version of myopic best response converges to a pure Nash equilibrium.

The argument for the minimization variation is analogous. ◀

We note that we leave as an open question to determine a bound on the number of steps a myopic best response approach would take to reach a pure Nash equilibrium; in particular, we do not yet know if the approach of Theorem 1 yields a pure Nash equilibrium in a polynomial number of steps.

We might have hoped that the above technique could allow us to show that for the 2-D Simultaneous Voronoi Game (and higher dimensions) that a pure Nash equilibrium exists. Unfortunately, that is not the case. One can readily find choices of 2 points for each of 3 players where no pure Nash equilibrium exists for both the maximization and minimization version of the problem. We have generated many such examples randomly, computing the Voronoi diagrams for the eight resulting configurations. One example in two dimensions is depicted in Figure 1. In this example, player 1 has choices $((1/4 + \varepsilon^2, 1/2 + \varepsilon), (1/4 + \varepsilon^2, 1/2 - \varepsilon))$, player 2 has choices $((1/4 - \varepsilon^2, 1/2 + \varepsilon), (1/4 - \varepsilon^2, 1/2 - \varepsilon))$, and player 3 has choices $((0, 0), (\varepsilon, 0))$ for sufficiently small $\varepsilon > 0$. The idea here is that player 3's choice is irrelevant, and players 1 and 2 are dividing up the Voronoi cell owned by point $(1/4, 1/2)$ in the Voronoi diagram of the points $(1/4, 1/2)$ and $(0, 0)$. If both player 1 and 2 choose their first point, or both choose their second point, then they divide the cell with a vertical line, which due to the geometry

of the cell favors player 1. However, if one of them chooses their first point and the other chooses their second point then they divide the cell with a roughly horizontal line, which gives each of them roughly half. Thus, there is no choice of points for which neither wants to deviate. This example applies equally in higher dimensions (by making higher dimensional coordinates all zero).

This example can be extended to show that for any number of players n, there are settings of points for the players so no pure Nash equilibrium exists, showing that this setting differs from previous work on symmetric Hotelling games on graphs, where it has been shown that when there are sufficiently many players a pure Nash equilibrium always exists [8]. Specifically, use the same example but for players $4, \ldots, n$ both of their choices will be in an epsilon-small neighborhood of $(0, 0)$. We note that such examples do not disprove the possibility that a pure Nash equilibrium exists with high probability if the points are chosen randomly.

Given that the 1-D Simultaneous Voronoi Game appears to have a very special structure (in terms of the existence of pure Nash equilibrium) that differs from k-D Simultaneous Voronoi Game for $k \geq 2$, it is natural to seek a 1-D variant that might shed more insight into the behavior in higher dimensions. This motivates us to look at the One Way 1-D Simultaneous Voronoi Game.

2.2 NP-Hardness of the One Way 1-D Simultaneous Voronoi Game

In contrast to the result of the previous subsection, we prove the NP-hardness of the maximization version of the One Way 1-D Simultaneous Voronoi Game whenever each player has $m \geq 4$ choices. We leave as an open question to find a reduction for $m = 2$ or 3, as well as for the minimization version.

▶ **Theorem 2** (For proof see [5]). *The problem of determining if a pure Nash equilibrium (PNE) exists in the maximization version of a One Way 1-D Simultaneous Voronoi Game is NP-hard for $m \geq 4$.*

We conjecture that determining if a pure Nash equilibrium exists in the k-D Simultaneous Voronoi Game is NP-hard for $k \geq 2$; however, we suspect that building the corresponding gadgets will prove technically challenging.

3 Random Voronoi Games

We now consider random variations of the Voronoi games we have considered, where each player's available choices are chosen uniformly at random from the underlying universe. While it is not clear such a model corresponds to any specific real-world scenario, such random problems are intrinsically interesting combinatorially and in relation to other similar studied problems. For example, in the context of load-balancing in distributed peer-to-peer systems, the authors of [6] study a model where one begins with a Voronoi diagram on N points (chosen uniformly at random from the universe, say the unit torus) corresponding to N servers. Then M agents sequentially enter; each is assigned k random points from the universe; and each agent chooses the one of its k points that lies in the Voronoi cell with the smallest number of agents, or *load* for that server. Other "power-of-choice" problems, such as the Achlioptas process [1], have spurred new understanding of phenomena such as explosive percolation.

Given our hardness results, a natural question is whether the One Way 1-D Simultaneous Voronoi Game has (or does not have) a pure Nash equilibrium with high probability in the

random variation. While we have not proven this result, we have proven bounds on the expected number of pure Nash equilibria in the random setting that are interesting in their own right and nearly answer this question. In particular, our careful analysis builds on the interesting relationship between random arcs on a circle and weighted sums of exponentially distributed random variables.

The following is our main result.

▶ **Theorem 3.** *The expected number of PNE for the maximization version of the One Way 1-D Simultaneous Voronoi Game is at most m, and at least $0.19^{m-1}m$.*

Interestingly, our bounds depend on the number of choices m, not the number of players. Unfortunately, this means that we cannot use these expectation bounds directly to show that, for example, the probability of a PNE existing is exponentially small in n. But the bounds provide insight by showing that pure Nash equilibria are typically few in number in the random case.

In proving Theorem 3, we need the following well-known property of exponential random variables. Let $X_1, \ldots, X_n$ be i.i.d. exponential random variables. Let $S_n = \sum_{j=1}^{n} X_j$. Recall that S_n is said to have a *gamma distribution*, and we use facts about the gamma distribution later in our analysis. Similarly, S_{n-1}/S_n is said to have a *beta distribution*, and we use facts about beta distributions as well. For example,

▶ **Lemma 4.** *$S_n, \frac{S_1}{S_2}, \frac{S_2}{S_3}, \ldots, \frac{S_{n-1}}{S_n}$ are all mutually independent.*

See e.g. [10] for its discussion of exponential random variables. In particular, the X_i can be viewed as the gaps in the arrivals of a Poisson process; the $n-1$ arrivals before the last are uniformly distributed on the interval $[0, S_n]$, from which one can derive the lemma above.

Another important fact we use is the following:

▶ **Fact 5.** *We have* $\mathrm{E}\left[(S_j/S_{j+1})^t\right] = \prod_{i=0}^{t-1} \frac{j+i}{j+1+i} = \frac{j}{j+t}$, *since* $S_j/S_{j+1} \sim \mathit{Beta}\,(j,1)$.

We note that in some of our arguments, the ordering of the variables becomes reversed, and we consider sums of the form $\sum_{j=n-i+1}^{n} X_j$. Of course this has the same distribution as $\sum_{j=1}^{i} X_i$, and the corresponding version of Lemma 4 holds. Where convenient, we therefore refer to $\sum_{j=n-i+1}^{n} X_j = S_i$ where there is no ambiguity as to the desired meaning.

Proof of Theorem 3. We begin by using linearity of expectations to write the expected number of PNE in terms of the probability that each player choosing their first choice will yield a stable configuration.

$$\begin{aligned}\mathrm{E}[\#\text{ PNE}] &= m^n \cdot \Pr[\text{first choices are stable}] \\ &= m^n \cdot \mathrm{E}[\Pr[\text{first choices are stable} \mid \text{position of first choices}]].\end{aligned}$$

Partition the circle into arcs according to the players' first choice points. Let A_i be the length of the ith smallest arc. As shown in [10], the A_i are distributed jointly as

$$A_i \sim \frac{1}{S_n} \sum_{j=n-i+1}^{n} \frac{X_j}{j}$$

where again the X_j are i.i.d. exponential random variables of mean 1 and $S_n = \sum_{j=1}^{n} X_j$. We say that an arc is stable if the player whose point starts the arc (going clockwise) does not wish to deviate to any of their other points. Given the position of the first choices, the

probability each arc is stable depends only on the other choices available to the player that owns the arc, and hence the stability of the arcs are independent. Therefore

$$\begin{aligned}&\Pr[\text{first choices are stable} \mid \text{position of first choices}]\\ &= \prod_{i=1}^{n} \Pr[i\text{th smallest arc is stable} \mid \text{position of first choices}].\end{aligned}$$

If the arc is the ith smallest, then it will be stable if the other choices fall in the same arc, one of the $i-1$ smaller arcs, or the front A_i-length portion of the $(n-i)$ larger arcs – except in the latter two cases, we must take into account that if a choice falls immediately backward into the arc directly counterclockwise of the current arc, then the arc is not stable. We therefore have the following calculation:

$$\begin{aligned}&\Pr[i\text{th smallest arc is stable} \mid \text{position of first choices}]\\ &= \left((n-i)A_i + \sum_{j=1}^{i} A_j - \min(A_i, \text{length of arc before i})\right)^{m-1}\\ &\le \left((n-i)A_i + \sum_{j=1}^{i} A_j\right)^{m-1}\\ &= S_n^{-(m-1)} \left((n-i)\sum_{j=n-i+1}^{n} \frac{X_j}{j} + \sum_{j=1}^{i}\sum_{k=n-j+1}^{n} \frac{X_k}{k}\right)^{m-1}\\ &= S_n^{-(m-1)} \left((n-i)\sum_{j=n-i+1}^{n} \frac{X_j}{j} + \sum_{k=n-i+1}^{n} (k-n+i)\frac{X_k}{k}\right)^{m-1}\\ &= S_n^{-(m-1)} \left(\sum_{j=n-i+1}^{n} (n-i+j-n+i)\frac{X_j}{j}\right)^{m-1}\\ &= S_n^{-(m-1)} \left(\sum_{j=n-i+1}^{n} X_j\right)^{m-1}\\ &= \left(\frac{S_i}{S_n}\right)^{m-1}.\end{aligned}$$

Note we have used $\sum_{j=n-i+1}^{n} X_j = S_i$ for convenience. Our resulting bound has a surprisingly clean form in terms of the S_i.

Thus, by Lemma 4 and Fact 5:

$$\begin{aligned}\Pr[\text{first choices are stable}] &\le \mathrm{E}\left[\prod_{i=1}^{n}\left(\frac{S_i}{S_n}\right)^{m-1}\right]\\ &= \mathrm{E}\left[\prod_{i=1}^{n-1}\left(\frac{S_i}{S_{i+1}}\cdot\frac{S_{i+1}}{S_{i+2}}\cdot\ldots\cdot\frac{S_{n-1}}{S_n}\right)^{m-1}\right]\\ &= \mathrm{E}\left[\prod_{i=1}^{n-1}\left(\frac{S_i}{S_{i+1}}\right)^{i(m-1)}\right] = \prod_{i=1}^{n-1}\mathrm{E}\left[\left(\frac{S_i}{S_{i+1}}\right)^{i(m-1)}\right]\\ &= \prod_{i=1}^{n-1}\frac{i}{i+i\,(m-1)} = \frac{1}{m^{n-1}}.\end{aligned}$$

It then follows that

$$\mathrm{E}\left[\#\mathrm{PNE}\right] \leq m^n \cdot \frac{1}{m^{n-1}} = m.$$

We can similarly find a lower bound, although some additional technical work is required.

$$\begin{aligned}
&\Pr[i\text{th smallest arc is stable} \mid \text{position of first choices}] \\
&= \left((n-i)A_i + \sum_{j=1}^{i} A_j - \min(A_i, \text{length of arc before i})\right)^{m-1} \\
&\geq \left((n-i)A_i + \sum_{j=1}^{i} A_j - A_i\right)^{m-1} \\
&= \left((n-i-1)A_i + \sum_{j=1}^{i} A_j\right)^{m-1} \\
&= S_n^{-(m-1)} \left(\sum_{j=n-i+1}^{n} (n-i-1+j-n+i)\frac{X_j}{j}\right)^{m-1} \\
&= S_n^{-(m-1)} \left(\sum_{j=n-i+1}^{n} \frac{(j-1)X_j}{j}\right)^{m-1}
\end{aligned}$$

Hence

$$\Pr[\text{first choices are stable}] \geq \mathrm{E}\left[S_n^{-(m-1)n} \prod_{i=1}^{n} \left(\sum_{j=n-i+1}^{n} \frac{(j-1)X_j}{j}\right)^{m-1}\right].$$

A simple stochastic domination argument (provided in full version of the paper [5]) shows that the expectation on the right side decreases if, in each term in the product, we equalize the coefficient, so that instead of terms of the form $\frac{(j-1)X_j}{j}$, the coefficient for all terms of the sum in the ith term of the product is the average $c_i = \frac{1}{i}\sum_{j=n-i+1}^{n} \frac{j-1}{j}$. This gives

$$\begin{aligned}
\Pr[\text{first choices are stable}] &\geq \mathrm{E}\left[S_n^{-(m-1)n} \prod_{i=1}^{n} \left(\sum_{j=n-i+1}^{n} c_i X_j\right)^{m-1}\right] \\
&= \mathrm{E}\left[S_n^{-(m-1)n} \prod_{i=1}^{n} (c_i S_i)^{m-1}\right] \\
&= \left(\prod_{i=1}^{n} c_i^{m-1}\right) \mathrm{E}\left[\prod_{i=1}^{n} \left(\frac{S_i}{S_n}\right)^{m-1}\right].
\end{aligned}$$

Observe that this expectation is the same as the one we computed in the upper bound. Therefore

$$\mathrm{E}[\#\ \mathrm{PNE}] \geq \left(\prod_{i=1}^{n} c_i^{m-1}\right) m \geq 0.19^{m-1} m.$$

The proof of the final inequality is presented in the full version of the paper [5]. ◀

We note that similar calculations can be done for the minimization version, although we have not found a clean form for the upper bound. We can, however, state the following lower bound, showing the expected number of pure Nash equilibria is at least inverse polynomial in n for a fixed number of choices m.

▶ **Theorem 6** (For proof see [5]). *The expected number of PNE for the minimization version of the One Way 1-D Simultaneous Voronoi Game is at least* $\frac{m(m-1)}{(mn-1)n^{m-1}}$.

We have done several experiments regarding Random Voronoi games, which prove consistent with our theoretical results and suggest some interesting conjectures, particularly for the 2-D Simultaneous Voronoi Game. Chief among these conjectures is that the Random k-D Simultaneous Voronoi game has a pure Nash Equilibrium with probability approaching 1 as n grows. The complete discussion of these results can be found in the full version of the paper

4 Correlated Equilibria

Our goal in this section is to show that, for the k-D Simultaneous Voronoi Game, correlated equilibria can be found in polynomial time. We present the results for $k = 1, 2$, and 3. The results appear to extend to higher dimensions but the geometric details are technical; we note the time required to determine the correlated equilibrium appears to grow as $n^{O(k)}$. The results also apply to the One Way 1-D Simultaneous Voronoi Game.

▶ **Theorem 7.** *For $k = 1, 2$, and 3, and for a fixed m, there is a polynomial time algorithm for finding a correlated equilibrium in the k-D Simultaneous Voronoi Game.*

We appeal to [17] and [12], who present polynomial time algorithms for finding a correlated equilibrium of games *polynomial type*. (The running times for these algorithms are not specifically presented in the papers and appear rather large, but are still polynomial.) A game of polynomial type is one that can be represented in polynomial space such that given each player's strategy, their utilities can be computed in polynomial time. The k-D Simultaneous Voronoi Game is of polynomial type because it can be represented in $O(nmk)$ space by a list of each players point choices and, given the players' strategies, the utilities can be found by computing the Voronoi diagram of the chosen points.

▶ **Theorem 8** (Theorem 4.5, [12]). *Given a game of polynomial type and a polynomial time algorithm for computing the expected utility of a player under any product distribution on strategies, there exists a polynomial time algorithm for finding a correlated equilibrium in that game.*

Jiang et al. proved Theorem 8 by constructing a linear program with a variable for each of the 2^n possible strategy profiles. The LP's constraints are non-negativity, and the constraints requiring that the variables form a correlated equilibrium. They do not, however, enforce that the variables sum to one, or even at most one, and rather use the sum of these variables as the objective. Thus, since a correlated equilibrium is guaranteed to exist by Nash's Theorem, this LP is unbounded and its dual is infeasible. They then run the ellipsoid algorithm for a polynomial number of steps on the dual LP (this takes polynomial time, since the dual LP has only polynomially many variables). They argue that the intermediate steps of the ellipsoid algorithm can be used to construct product distributions of which there is a convex combination that is a valid correlated equilibrium, and which can be found with a second linear program.

The second linear program's separation oracle requires as a subroutine a polynomial time algorithm for computing the expected utility of a player given a product distribution over the strategies. (This requirement is referred to in [17] and [12] as the *polynomial expectation property*.) Our work is to demonstrate polynomial time algorithms for this subroutine. We note that it is not immediate that such an algorithm should exist, even when each player has only $m = 2$ choices, as the number of possible configurations is m^n. Hence, we cannot simply sum over all configurations when calculating the expectation. In [17] it is noted that for certain congestion games, these expectations can be computed using dynamic programming, essentially adding one player in at a time and updating accordingly. Our approach is similar in spirit, but requires taking advantage of the underlying geometry.

▶ **Lemma 9** (For proof see [5]). *Computing a player's expected utility under a product distribution on strategies in the k-D Simultaneous Voronoi Game takes* $O(n^{2k-1} \log n)$ *time for* $k \leq 3$.

The proof is trivial for one dimension. For two and three dimensions, the key idea is to partition the space into regions where the possible Voronoi cell boundaries do not cross.

5 Conclusion

We have introduced a new but we believe important set of variants on Voronoi games, where each player has a disjoint set of points to choose from. We believe these variations are motivated both by natural economic settings, and because of the possible connections to other "power-of-choice" processes in which participants choose from a limited set of random options.

In particular, we note that the Voronoi choice games we propose offer the chance to consider randomized versions of the problem, where the set of possible choices for each player is chosen uniformly over of the space. We have conjectured that the Random k-D Simultaneous Voronoi Game has a pure Nash equilibrium with high probability, based on a simulation study. While this is perhaps the most natural open question in this setting, there remain several other questions for both the simultaneous and sequential versions of Voronoi choice problems, in the worst case and with random point sets.

References

1 Dimitris Achlioptas, Raissa M D'Souza, and Joel Spencer. Explosive percolation in random networks. *Science*, 323(5920):1453–1455, 2009.

2 Robert J Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.

3 Yossi Azar, Andrei Z Broder, Anna R Karlin, and Eli Upfal. Balanced allocations. *SIAM journal on computing*, 29(1):180–200, 1999.

4 Avrim Blum, MohammadTaghi Hajiaghayi, Katrina Ligett, and Aaron Roth. Regret minimization and the price of total anarchy. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 373–382. ACM, 2008.

5 Meena Boppana, Rani Hod, Michael Mitzenmacher, and Tom Morgan. Voronoi choice games, 2016. `arXiv:1604.07084`.

6 John Byers, Jeffrey Considine, and Michael Mitzenmacher. Simple load balancing for distributed hash tables. In *Peer-to-peer Systems II*, pages 80–87. Springer, 2003.

7 Christoph Dürr and Nguyen Kim Thang. Nash equilibria in voronoi games on graphs. In *Proceedings of the 5th Annual European Symposium on Algorithms*, pages 17–28. Springer, 2007.

8 Gaëtan Fournier and Marco Scarsini. Hotelling games on networks: efficiency of equilibria. *Available at SSRN 2423345*, 2014.

9 J. J. Gabszewicz and J.-F. Thisse. Location. In *Handbook of Game Theory with Economic Applications*. Volume 1, Chapter 9. R. Aumann and S. Hart, editors. Elsevier Science Publishers, 1992.

10 Lars Holst. On the lengths of the pieces of a stick broken at random. *Journal of Applied Probability*, pages 623–634, 1980.

11 Harold Hotelling. Stability in competition. *The Economic Journal*, 39(153):41–57, 1929.

12 Albert Xin Jiang and Kevin Leyton-Brown. Polynomial-time computation of exact correlated equilibrium in compact games. *Games and Economic Behavior*, 21(1-2):183–202, 2013.

13 Masashi Kiyomi, Toshiki Saitoh, and Ryuhei Uehara. Voronoi game on a path. *IEICE TRANSACTIONS on Information and Systems*, 94(6):1185–1189, 2011.

14 Marios Mavronicolas, Burkhard Monien, Vicky G Papadopoulou, and Florian Schoppmann. Voronoi games on cycle graphs. In *Proceedings of Mathematical Foundations of Computer Science*, pages 503–514. Springer, 2008.

15 M Mitzenmacher, Andrea W Richa, and R Sitaraman. The power of two random choices: A survey of techniques and results. In *Handbook of Randomized Computing*, pages 255–312, 2000.

16 John Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

17 Christos H Papadimitriou and Tim Roughgarden. Computing correlated equilibria in multiplayer games. *Journal of the ACM*, 55(3):14, 2008.

18 Sachio Teramoto, Erik D Demaine, and Ryuhei Uehara. Voronoi game on graphs and its complexity. In *IEEE Symposium on Computational Intelligence and Games*, pages 265–271. IEEE, 2006.

The Complexity of Hex and the Jordan Curve Theorem*

Aviv Adler[1], Constantinos Daskalakis[2], and Erik D. Demaine[3]

1 **MIT CSAIL, Cambridge, USA**
`adlera@mit.edu`
2 **MIT CSAIL, Cambridge, USA**
`costis@mit.edu`
3 **MIT CSAIL, Cambridge, USA**
`edemaine@mit.edu`

Abstract

The Jordan curve theorem and Brouwer's fixed-point theorem are fundamental problems in topology. We study their computational relationship, showing that a stylized computational version of Jordan's theorem is PPAD-complete, and therefore in a sense computationally equivalent to Brouwer's theorem. As a corollary, our computational result implies that these two theorems directly imply each other mathematically, complementing Maehara's proof that Brouwer implies Jordan [10]. We then turn to the combinatorial game of Hex which is related to Jordan's theorem, and where the existence of a winner can be used to show Brouwer's theorem [6]. We establish that determining who won an (implicitly encoded) play of Hex is PSPACE-complete by adapting a reduction (due to Goldberg [7]) from Quantified Boolean Formula (QBF). As this problem is analogous to evaluating the output of a canonical path-following algorithm for finding a Brouwer fixed point – and which is known to be PSPACE-complete [8] – we thereby establish a connection between Brouwer, Jordan and Hex higher in the complexity hierarchy.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases Jordan, Brouwer, Hex, PPAD, PSPACE

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.24

1 Introduction

The Jordan curve theorem states that a simple closed curve C in $\mathbb{R}^2$ divides the plane into two connected components S_1, S_2 [9]. In particular, any continuous curve from a point $x \in S_1$ to a point $y \in S_2$ must intersect C. There are several known proofs of this basic topological fact, including one via Brouwer's fixed point theorem by Maehara [10]. In an earlier paper, Gale explores the equivalence between Brouwer's theorem and a theorem closely related to Jordan's, pertaining to the combinatorial game Hex [6]. In Hex, a rhomboidal board is partitioned into hexagonal tiles as in Figure 1, and two players claim tiles of the board until one of the players can connect the two opposite sides of the board that belong to him. The Hex theorem states that, once all tiles on the board are claimed, at least one of the two players has won.

In this paper we study the relationship of the Jordan, Brouwer and Hex theorems from both a computational and a mathematical standpoint. In particular, our goal is to show their 'equivalence' in both domains. We say that two theorems are *mathematically equivalent* if

* This work was partially supported by the NSF (grant CCF-1551875) and ONR (grant N00014-12-1-0999).

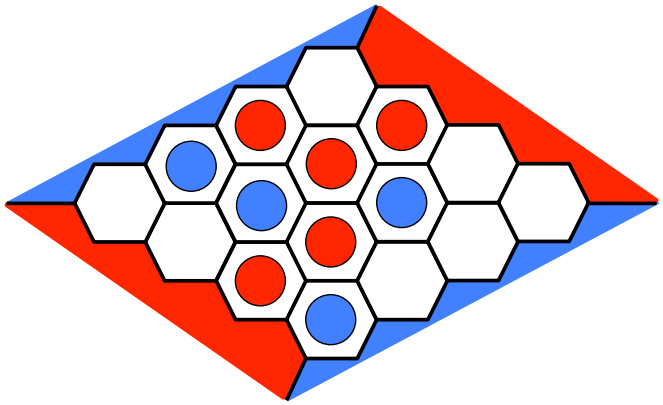


Figure 1 A sample Hex position, where Player 1 (red) has won.

(informally) they can be used to prove each other in an 'elementary' way, and *computationally equivalent* if the search problems with solutions guaranteed by them are complete in the same complexity class. This study is catalyzed by defining a stylized computational problem inspired by Jordan curve theorem as follows:

ZeroSurfaceCrossing
`Input:` Two circuits $F_1, F_2 : \{0,1\}^n \times \{0,1\}^n \to \{-2^m+1, \ldots, 2^m-1\}$, defining two continuous surfaces $f_1, f_2 : [0,1]^2 \to \mathbb{R}$ in $\mathbb{R}^3$ via interpolation.[a]

`Output:` A point (x, y) such that $f_1(x, y) = f_2(x, y) = 0$, or a violation of the following boundary conditions:

(1) $f_1(0, y) \geq 0$ and (2) $f_1(1, y) \leq 0$ for all y;

(3) $f_2(x, 0) \geq 0$ and (4) $f_2(x, 1) \leq 0$ for all x.

[a] The input to each circuit is interpreted in the obvious way as specifying a point $(x, y) \in [0,1]^2$, where both x and y are integer multiples of $1/(2^n - 1)$. Hence these circuits determine the values of f_1, f_2 at all such points. The values of f_1, f_2 at all other points are determined from these values via interpolation.

It is quite intuitive that, if Conditions 1–4 are satisfied, then the surfaces f_1 and f_2 should have an intersection on the zero plane. One way to show this is using the Jordan curve theorem; see Lemma 10. Moreover, given that f_1, f_2 are defined by circuits via interpolation, it is easy to see that the problem belongs to NP and thus in TFNP, the class of total problems in NP. The question is how it relates to other classes in TFNP [11]. We show the following:

▶ **Theorem 1.** ZeroSurfaceCrossing *is PPAD-complete.*

Given that Brouwer, the stylized computational problem of computing fixed points of continuous functions (defined formally in Section 3), is also PPAD-complete [11, 3, 5], Theorem 1 implies that ZeroSurfaceCrossing and Brouwer are computationally equivalent. Additionally, it helps establish the mathematical equivalence of the Brouwer and Jordan theorems. Maehera showed that Brouwer implies Jordan's theorem. Exploiting the proof of Theorem 1, we show the other direction, that Jordan implies Brouwer's theorem. Moreover, in view of Gale's proof that the Hex and Brouwer theorems are mathematically equivalent [6], our result also establishes that all three theorems are equivalent.

▶ **Proposition 2.** *The Jordan, Brouwer and Hex theorems are mathematically equivalent.*

1.1 Jordanian Action Inside PPAD

A close variant of the problem defined above is the curve crossing problem defined as follows.

CrossingCurves
`Input:` Two circuits $F_1, F_2 : \{0,1\}^n \to \{0, 2^{-m}, 2 \cdot 2^{-m}, \ldots, 1\}^2$, defining two continuous curves $f_1, f_2 : [0,1] \to [0,1]^2$ in $[0,1]^2$ via interpolation.[a]

`Output:` A pair $t_1, t_2 \in [0,1]$ such that $f_1(t_1) = f_2(t_2)$, or a violation of the following boundary conditions:

(1) $f_1(0) = (0,0)$; (2) $f_1(1) = (1,1)$; (3) $f_2(0) = (0,1)$ and (4) $f_2(1) = (1,0)$.

[a] The input to each circuit is interpreted in the obvious way as specifying an integer multiple of $1/(2^n-1)$, so that F_i defines the location of curve f_i for all inputs in $\{0, 1/(2^n-1), \ldots, 1\}$. The location of the curve at any $t \in [j/(2^n-1), (j+1)/(2^n-1)]$, where $j \in \{0, \ldots, 2^n-1\}$, is determined by linearly interpolating between the locations of the curve at $j/(2^n-1)$ and $(j+1)/(2^n-1)$.

Again it is quite intuitive that, unless Conditions (1)–(4) in the definition of the problem fail, the curves defined by any input to CrossingCurves should cross. Indeed, this can be proven via the Jordan curve theorem, and because of Proposition 2 via Brouwer's fixed point theorem as well. We provide a direct proof using Brouwer's fixed point theorem, which implies as a corollary that the problem lies within PPAD.

▶ **Theorem 3.** CrossingCurves *is in PPAD.*

Under monotonicity conditions on at least one of the two curves, it is easy to see that CrossingCurves is in P. For example, suppose that at least one of the two curves f_i satisfies that, for all $0 \le t < t' \le 1$, $f_i(t)_1 < f_i(t')_1$, where $f_i(t)_1$ represents the first coordinate of $f_i(t)$ and similarly for $f_i(t')_1$. Under this condition, CrossingCurves can be easily solved via binary search. Similar conditions can be defined with respect to the second coordinate. When neither curve satisfies such a monotonicity condition with respect to neither coordinate, we do not see how to construct a polynomial time algorithm. At the same time, we do not see how an instance of CrossingCurves can encode the several paths and cycles that may co-exist in an instance of EndOfTheLine, the canonical PPAD-complete problem—see Section 2. (In comparison, the intersections of the surfaces of ZeroSurfaceCrossing with the zero-plane may comprise several paths and cycles, which allow encoding EndOfTheLine instances.) We leave pinning down the precise complexity of CrossingCurves for future work, expecting that the complexity classes defined in [4] may be useful in this classification.

1.2 Jordanian Action Over PPAD

While so far all action has taken place inside TFNP, we also explore how the three theorems, Jordan, Brouwer and Hex, are related higher in the complexity hierarchy. It was recently established that several algorithms for computing Brouwer fixed points and Nash equilibria are in fact capable of solving all of PSPACE [8]. For example, given an instance $\mathcal{I}$ of some problem in PSPACE, one can construct a 2-player game $\mathcal{G}$ such that the Nash equilibrium output by the Lemke-Howson algorithm provides a solution to $\mathcal{I}$ as a byproduct. Similar facts are known for homotopy methods.

We are thus interested in whether computational problems relating to Jordan and Hex also have the power of solving PSPACE. We propose the problem WhoWonHex, asking to determine whether player 1 is the winner of a Hex play. An instance of the problem comprises a circuit that takes as input the binary description of a cell in the Hex board and outputs the name of the player, 1 or 2, who claimed it during the play. We provide a formal

description of Hex in Section 2.3, and define WhoWonHex in Section 5.2, establishing the following:

▶ **Theorem 4.** WhoWonHex *is PSPACE-complete.*

The proof of Theorem 4 can be obtained fairly easily using recent work of Goldberg [7]. There is a canonical method to determine who is the winner in a play of Hex by performing a walk on the Hex board. The walk starts at one of the corners of the Hex board and performs pivoting steps depending on which player has claimed the cells neighboring the current location of the walk, until another corner of the board is reached, which always happens due to topological reasons. What corner is reached determines which player won. This pivoting algorithm is quite reminiscent to the canonical algorithm for solving instances of 2-dimensional Sperner. An instance of this problem provides a succinct description of the coloring of the vertices of a square lattice using 3 colors and asks to identify a trichromatic triangle or a violation of certain boundary conditions by the coloring. The problem is PPAD-complete [11, 3, 2] and Goldberg recently established that identifying the tri-chromatic triangle reachable by the standard pivoting algorithm for this problem is PSPACE-complete. Theorem 4 is proven by making an analogy between the pivoting algorithms that solve Sperner and WhoWonHex. The precise details are a bit more intricate than this intuition, as we have to exploit the structure of the Sperner instances constructed in Goldberg's proof.

It is worth pointing out that the problem WhoWonHex that we study is very different than the typical computational problem studied in combinatorial game theory, namely determining given a configuration of the board whether some player has a winning strategy. Our problem is instead to determine who is the winner, once the play is completed.

Roadmap. We provide basic definitions in Section 2, recalling the Jordan curve theorem and Brouwer's fixed point theorem. We also describe the game of Hex and define the computational problems Brouwer and EndOfTheLine along with the class PPAD. In Section 3, we show that Brouwer and Jordan are equivalent, both mathematically and computationally (through the ZeroSurfaceCrossing problem). In Section 4, we discuss CurveCrossing showing that it is in PPAD. Finally, in Section 5, we show that WhoWonHex is PSPACE-complete. In Section 6 we conclude with some open problems suggested by our work.

2 Preliminaries

In this section, we will formally define all the theorems, problems, and constructs which we will analyze; in particular, we will define:

1. Brouwer's fixed-point theorem;
2. the Jordan curve theorem;
3. the game of Hex and the Hex theorem;
4. the complexity class PPAD, and the canonical computational problem EndOfTheLine that is associated with it.

Additionally, in order to formally describe the computational problems given above, we also need to define the interpolation schemes we use to transform functions over bitstrings to functions over the (continuous) unit square. We use a standard technique of using the bitstring to describe a point on a lattice; then, for inputs not on the lattice, the output is defined by interpolating from the outputs on the lattice. The formal construction is given in our full paper [1].

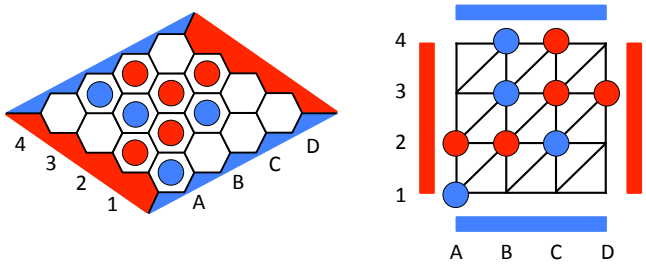


Figure 2 A Hex board and its corresponding dual-graph representation.

2.1 Brouwer's Fixed-Point Theorem and the Jordan Curve Theorem

We define here the theorems of Brouwer and Jordan. For Brouwer, we give a special case in two dimensions, involving functions from the unit square to itself. (This can be extended to the general two-dimensional case on any compact and convex set and higher dimensions; see e.g. [11, 3, 5] and their references.)

▶ **Theorem 5** (Brouwer's fixed-point theorem). *Given any continuous function* $f : [0,1]^2 \to [0,1]^2$*, there is a* fixed point*, i.e. some* $x \in [0,1]^2$ *such that* $f(x) = x$*.*

▶ **Theorem 6** (The Jordan curve theorem). *Any simple closed curve* ϕ *in* $\mathbb{R}^2$ *divides the space into two regions, one finite (the inside) and one infinite (the outside).*

2.2 PPAD and its Related Computational Problems

▶ **Definition 7** (The PPAD graph). Let N and P be circuits, both of which take as input an n-bit string and return an n-bit string as output. We then consider the directed graph whose vertices are n-bit strings such that there is an edge (u, v) if and only if $N(u) = v$ and $P(v) = u$.

By this definition, it is clear that each vertex has in- and out-degree of at most 1 (since any vertex u can only be preceded by $P(u)$ and succeeded by $N(u)$), and thus the graph must consist of a disjoint collection of isolated vertices, directed paths, and directed cycles. We now consider the following computational problem on this graph:

▶ **Definition 8** (EndOfTheLine). Given circuits N and P defining the PPAD graph G:

1. if the vertex 0^n is not a source vertex (with in-degree 0 and out-degree 1), return 0^n;
2. if the vertex 0^n is a source vertex, return any *other* unbalanced vertex (with in-degree and out-degree not equal).

EndOfTheLine is the canonical PPAD-complete problem. Because no directed graph can have exactly one unbalanced vertex, the existence of a solution is guaranteed. Of particular interest to us is the fact that a computational variant of Brouwer's fixed-point theorem (which we will formally describe in the next section) is also PPAD-complete.

2.3 The Game of Hex

The game of Hex is a combinatorial game, played (in its normal, two-player two-dimensional version) on a hexagonally-tiled board (such as the one shown on the left in Figure 2). Each player (player 1 represented in red, and player 2 represented in blue) starts in the possession of two opposing sides of the board; they take turns placing stones of their color on unoccupied tiles. Each has the goal of connecting their two sides with a path (or *bridge*) of stones of

their color; the first to do so wins. For ease of representation, we imagine instead that the players are placing stones on the *vertices* (rather than facets) of the dual graph, which is represented on the right in Figure 2. In order to escape the restriction that the number of red and blue stones is the same (or at most differ by 1), we allow players to 'pass' (i.e. not put a stone down); there is no reason to do so if the player is trying to win, but it makes the following analysis simpler and more general.

Although seemingly just a simple combinatorial game, Hex is known to have very deep mathematical properties. First, a very elegant proof shows that some player is guaranteed to win [6], i.e. if every vertex (in the dual-graph representation) is occupied by a stone, then there must be a pair of opposing sides which are joined by a path of stones of their corresponding color. In fact, exactly one player must win (the fact that it's impossible for both players to have bridges at the same time is intuitively connected to the Jordan curve theorem). Intriguingly, the theorem that Hex must have a winner (which seems at first to be merely an interesting curiosity) can, like Sperner's Lemma, be used to actually prove Brouwer's fixed-point theorem [6].

3 Brouwer vs. Jordan in TFNP

In this section, we consider the relationship between Brouwer's fixed-point theorem and the Jordan curve theorem. In particular, we want to show that Brouwer's fixed-point theorem can be proven directly from the Jordan curve theorem, thus complementing Maehara's result that the Jordan curve theorem can be proved directly from Brouwer's fixed-point theorem [10]. We then consider a computational version of the Jordan curve theorem, which we call ZeroSurfaceCrossing; this problem is a search problem where the existence of a solution is guaranteed by the Jordan curve theorem. We then show that ZeroSurfaceCrossing is PPAD-complete. This makes it equivalent to the computational problem of finding a fixed point of a function (where the existence of a solution is guaranteed by Brouwer's fixed-point theorem), thus demonstrating a computational link between the two theorems in addition to the mathematical link.

3.1 The Zero Surface Crossing Problem

We define a problem where the existence of a solution is intended to be guaranteed by Jordan.

▶ **Definition 9** (Zero Surface Crossing). We are given continuous functions $f_1, f_2 : [0,1]^2 \to [-1,1]$ (which are therefore 2-dimensional surfaces in $\mathbb{R}^3$) satisfying the following conditions:
1. $f_1(0,y) \geq 0$ and $f_1(1,y) \leq 0$ for all y;
2. $f_2(x,0) \geq 0$ and $f_2(x,1) \leq 0$ for all x.

The goal is to find some $(x,y) \in [0,1]^2$ such that $f_1(x,y) = f_2(x,y) = 0$.

We wish to show that the Jordan curve theorem implies that such a point (x,y) exists; in order to give this proof, we consider the computational version defined in the introduction as ZeroSurfaceCrossing.

We will first show that the Jordan curve theorem implies that the computational version is guaranteed to have a valid output; we will then use basic topological principles to show that the mathematical version given in Definition 9 also must have a solution. We need the following notation:
1. let $X_0 = \{(0,y) : 0 \leq y \leq 1\}$ and $X_1 = \{(1,y) : 0 \leq y \leq 1\}$;
2. let $Y_0 = \{(x,0) : 0 \leq x \leq 1\}$ and $Y_1 = \{(x,1) : 0 \leq x \leq 1\}$.

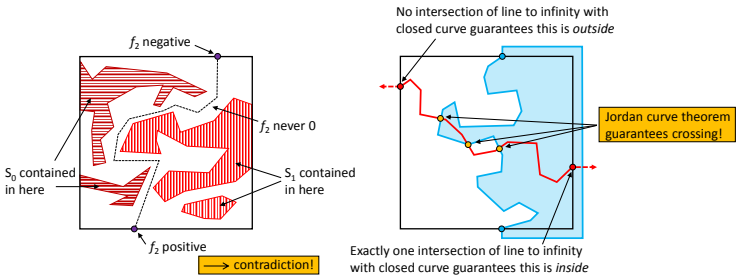


Figure 3 *Left:* illustration of proof of assertion 1; *Right:* illustration of proof of assertion 2.

In short, these sets are the four sides of the square $[0,1]^2$. For these lemmas, we sketch the proofs; the full formal proofs are given in the full version of the paper [1].

▶ **Lemma 10** (Jordan to Computational ZeroSurfaceCrossing). *The Jordan curve theorem implies that the computational version of* ZeroSurfaceCrossing *has a valid output.*

Proof Sketch. We assume that the boundary conditions hold; we thus want to show the existence of a point $(x,y) \in [0,1]^2$ such that $f_1(x,y) = f_2(x,y) = 0$. We will show the following:

1. there exists a path from X_0 to X_1 such that at every point on the path, f_2 has value 0; by symmetry, there then must also be a path from Y_0 to Y_1 such that at every point, f_1 has value 0.
2. the Jordan curve theorem then implies that these two paths must cross, giving a point $(x,y) \in [0,1]^2$ such that $f_1(x,y) = f_2(x,y) = 0$.

The proofs are depicted in Figure 3 (assertion 1 on the left, assertion 2 on the right). We remark that assertion 1 only holds because of the computational setting, where the function is defined by a circuit and the interpolation procedure (see full proof for details). ◀

We can now use this to tackle the problem of showing the existence of a solution to the mathematical version as given in Definition 9.

▶ **Proposition 11** (Jordan to Non-Computational Zero Surface Crossing). *The Jordan curve theorem implies that the mathematical version of Zero Surface Crossing always has a solution.*

Proof Sketch. We use the following general strategy: we note that only the set of points where f_1 is 0 and the set of points where f_2 is 0 matters. Thus, we can use instead two functions f_1^* and f_2^* which are 0 at the same points, but which are Lipschitz. We can then show that for any $\epsilon > 0$, if we approximate f_1^* and f_2^* with circuits F_1, F_2 with sufficiently many bits, any point which is a zero of the circuits must be within ϵ of 0 for f_1^*, f_2^*. We then take a sequence of such approximate zeroes as $\epsilon \to 0$; by compactness of $[0,1]^2$, this sequence must have at least one limit point, which we can then show is a shared zero of f_1^*, f_2^* and hence of f_1 and f_2. ◀

Finally, we can use this result to prove Brouwer as a consequence of Jordan.

▶ **Theorem 12** (Jordan implies Brouwer). *Brouwer's fixed-point theorem can be shown as a direct consequence of the Jordan curve theorem.*

Proof. Given any mapping $g : [0,1]^2 \to [0,1]^2$, we wish to show the existence of a fixed point (x, y) such that $g(x, y) = (x, y)$. Let g_x and g_y be the x- and y-components of g respectively.

We then define the functions $f_1, f_2 : [0,1]^2 \to [-1,1]$ as follows: $f_1(x, y) = g_x(x, y) - x$ and $f_2(x, y) = g_y(x, y) - y$. First, we note that the outputs of f_1, f_2 indeed must fall in $[-1, 1]$; this is because g_x and g_y have outputs in $[0, 1]$ and $-x$ and $-y$ each range through $[-1, 0]$. We also note that f_1, f_2 satisfy the boundary conditions given in Definition 9, since $f_1(0, y) = g_x(0, y) \geq 0$ and $f_1(1, y) = g_x(1, y) - 1 \leq 0$ for all y, and similarly $f_2(x, 0) = g_y(x, 0) \geq 0$ and $f_2(x, 1) = g_y(x, 1) - 1 \leq 0$ for all x. Thus, we can apply Proposition 11 to show that there is some (x, y) such that $f_1(x, y) = f_2(x, y) = 0$.

But this implies that $g_x(x, y) = f_1(x, y) + x = x$ and $g_y(x, y) = f_2(x, y) + y = y$, i.e. that $g(x, y) = (x, y)$, thus proving Brouwer's fixed-point theorem as a consequence of Zero Surface Crossing and hence as a consequence of the Jordan curve theorem. ◀

This, along with Maehara's result on showing Jordan from Brouwer [10] and Gale's result on the equivalence of Brouwer and the Hex theorem [6], thus concludes the proof of Proposition 2 (that Brouwer, Jordan, and Hex are all mathematically equivalent theorems). We also remark that reversing the above reduction yields an alternative proof that Brouwer's fixed-point theorem can be used to prove the Jordan curve theorem; however, as this is a known result, we will not show this in detail.

3.2 Computational Equivalence of Brouwer and Jordan

We now want to show that ZeroSurfaceCrossing is PPAD-complete. To do this, we first define the computational version of Brouwer, which is well-known to be PPAD-complete [11]:

> Brouwer
> `Input:` A circuit $G : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$, defining a mapping $g : [0,1]^2 \to [0,1]^2$ via interpolation (as described in Appendix A).
>
> `Output:` a point (x, y) such that $g(x, y) = (x, y)$ (i.e. a fixed point of g).

▶ **Lemma 13.** ZeroSurfaceCrossing *is PPAD-hard.*

Proof. We prove this by showing that Brouwer can be reduced to it, using the same reduction as in the proof of Theorem 12. In particular, we note that this reduction requires no fudging with outputs, as if (x, y) is a lattice point (where x and y can each be expressed by an n-bit string) then $f_1(x, y) = g_x(x, y) - x$ can be expressed by an $(n+1)$-bit string, since the output $g_x(x, y)$ is an n-bit string as well; the same obviously holds for $f_2(x, y) = g_y(x, y) - y$. We then note that since all lattice points behave well under the transformation, and the interpolation given in Appendix A is linear in both cases, the reduction requires no further steps; a solution to the derived ZeroSurfaceCrossing instance is immediately a fixed point of the original Brouwer instance. ◀

▶ **Lemma 14.** ZeroSurfaceCrossing *is in PPAD.*

Proof. We show this by reducing ZeroSurfaceCrossing to Brouwer. Without loss of generality, we assume that f_1 and f_2 take two n-bit strings as input and output two $(n+1)$-bit strings (we add the extra bit to account for the fact that they can have negative output, so as to keep the interval size constant). We can now define $g : [0,1]^2 \to [0,1]^2$ as follows; this is essentially the above reduction, reversed and with outputs truncated to be within $[0, 1]$:

$$g_x(x, y) = \max\big[\min[x + f_1(x, y), 1], 0\big], \text{ and } g_y(x, y) = \max\big[\min[y + f_2(x, y), 1], 0\big].$$

As before, since the reduction holds exactly at lattice points, it holds exactly everywhere else as well (by how the interpolation works). Furthermore, it trivially has a fixed point at any (x, y) such that $f_1(x, y) = f_2(x, y) = 0$. We now wish to show that any *additional* fixed points can only be a result of f_1 or f_2 violating a boundary condition (which we recally is an acceptable output to ZeroSurfaceCrossing).

The only way g could have another fixed point is if the capping of g_x or g_y to be between 0 and 1 held the displacement to 0 when otherwise it would have been nonzero. This happens only if the input is already on the boundary (otherwise the cap cannot completely remove a nonzero displacement in any direction); hence, it can only happen if one of the following four events happens: (a) $f_1(0, y) < 0$; (b) $f_1(1, y) > 0$; (c) $f_2(x, 0) < 0$; or (d) $f_2(x, 1) > 0$, which all represent violations of the boundary conditions set by ZeroSurfaceCrossing. Hence, any fixed point of g corresponds to a solution to ZeroSurfaceCrossing, which means that ZeroSurfaceCrossing can be reduced to Brouwer, and so it is in PPAD. ◀

Lemmas 13 and 14 thus imply Theorem 1 (ZeroSurfaceCrossing is PPAD-complete).

4 Crossing Curves

We now discuss the CrossingCurves problem; in particular, we show that it is in PPAD.

Proof of Theorem 3. We recall that the CrossingCurves problem involves two curves $f_1, f_2 : [0, 1] \to [0, 1]^2$ such that $f_1(0) = (0, 0)$, $f_1(1) = (1, 1)$, $f_2(0) = (0, 1)$ and $f_2(1) = (1, 0)$; these curves are formally defined by circuits which map n-bit strings to points on a discrete lattice in $[0, 1]^2$ (with the continuous curves defined by interpolation over these circuits). We denote f_1^x, f_1^y, f_2^x, and f_2^y as the functions describing f_1 and f_2's outputs in the dimensions x and y. We note that although $f_1^x(0) = f_2^x(0) = 0$ and $f_1^x(1) = f_2^x(1) = 1$, f_1^x and f_2^x are not necessarily monotonically increasing; the two curves can snake back and forth. Nevertheless, the Jordan curve theorem does guarantee that there will be a crossing point, i.e. a pair of *times* (if we interpret the input of f_1 and f_2 to be a time between 0 and 1) t_1, t_2 such that $f_1(t_1) = f_2(t_2)$. The task is to find the crossing point.

We do this by defining the function $g : [0, 1]^2 \to \mathbb{R}^2$ (where its x and y components are denoted g^x and g^y respectively) such that

$$g^x(t_1, t_2) = t_1 - f_1^x(t_1) + f_2^x(t_2) \text{ and } g^y(t_1, t_2) = t_2 - f_1^y(t_1) + f_2^y(t_2)\,.$$

Clearly this function is continuous, and (t_1, t_2) is a fixed point if and only if $f_1^x(t_1) = f_2^x(t_2)$ and $f_1^y(t_1) = f_2^y(t_2)$, i.e. if and only if $f_1(t_1) = f_2(t_2)$. This is already very close to showing what we need to show; the only trouble is that an application of g might end up at a point outside of $[0, 1]^2$, breaking our use of Brouwer. We thus define $\hat{g}$ to be g, but with upper and caps to its values at 1 and 0 respectively. Formally:

$$\hat{g}^x(t_1, t_2) = \max\left[\min[g^x(t_1, t_2), 1], 0\right] \text{ and } \hat{g}^y(t_1, t_2) = \max\left[\min[g^y(t_1, t_2), 1], 0\right].$$

We now have a function which does not leave $[0, 1]^2$ and has a fixed point at any t_1, t_2 such that $f_1(t_1) = f_2(t_2)$. The only trouble is we need to make sure we did not create any new fixed points by this capping method;

A new fixed point can only happen if one of the following four events happens:

1. $-f_1^x(t_1) + f_2^x(t_2) < 0$ when $t_1 = 0$;
2. $-f_1^x(t_1) + f_2^x(t_2) > 0$ when $t_1 = 1$;
3. $-f_1^y(t_1) + f_2^y(t_2) < 0$ when $t_2 = 0$;
4. $-f_1^y(t_1) + f_2^y(t_2) > 0$ when $t_2 = 1$.

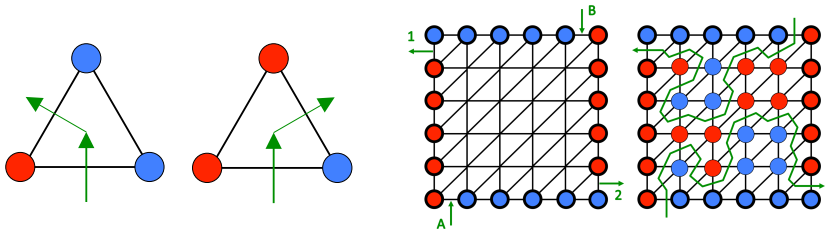


Figure 4 *Left:* proceeding on a Hex walk through a cell; each step is forced. *Right:* an empty board (with player-owned edges) has two entries, A and B, and two exits 1 and 2. If the walk entering at A exits through 2 (as shown at far right), then player 1 is the winner; if it exits through 1, then player 2 is the winner.

But we note that since $f_1^x(0) = 0$ and $f_1^x(1) = 1$ (and both f_1 and f_2 are constrained to be in $[0, 1]$ in both components) that the first two cannot happen; furthermore, since $f_2^y(0) = 1$ and $f_2^y(1) = 0$, the second two cannot happen. Thus, a fixed point still occurs at (t_1, t_2) if *and only if* $f_1(t_1) = f_2(t_2)$, as we wanted. We can thus apply the Brouwer fixed-point algorithm to $\hat{g} : [0, 1]^2 \to [0, 1]^2$ and read off a solution to the crossing-curves problem. Since finding a Brouwer fixed-point is in PPAD [11], CrossingCurves is also in PPAD. ◀

5 Hex, Brouwer, and Jordan in PSPACE

We have already established that the Hex, Brouwer, and Jordan theorems are mathematically equivalent. We have also identified two problems motivated by Jordan's curve theorem that lie in PPAD. One of these is in fact PPAD-complete and thereby computationally equivalent to Brouwer. It is natural to ask whether a computational version of the Hex theorem is also related to PPAD. The link between PPAD and the Hex theorem is more immediate and striking than that between the Jordan curve theorem and PPAD. In particular, Gale's proof that Hex always has a winner [6] is strikingly similar to the proof of Sperner's lemma, another well-known topological fact giving rise to the PPAD-complete problem called Sperner. Gale's proof generates a PPAD type graph, as defined in Section 2.2. In the next sections, we briefly summarize Gale's results and discuss natural questions arising from it in relation to PPAD and, as it turns out, PSPACE.

5.1 Hex and PPAD

As with all PPAD-type problems, in Hex the proof of existence of a 'bridge' (a winning sequence for one of the two players) in a filled board comes with a pivoting algorithm to find it; this algorithm was described by Gale, who (remarkably) used it to show that Brouwer's fixed-point theorem and the Hex theorem are mathematically equivalent. The pivoting algorithm is briefly sketched in Figure 4 on the dual-graph representation of the Hex board (in which the stones are placed on vertices, rather than faces, of the graph). In brief, there are two places on the edge of the board where one can enter with red on their left and blue on their right, as shown in the figure; suppose one starts at A. Then, one walks over the faces of the dual graph (or, equivalently, on the vertices of the original board), keeping red on their left and blue on their right, until they exit the board at either 1 or 2. Exiting at 2 implies a victory for player 1 (red), and exiting at 1 implies a loss. This pivoting algorithm is especially reminiscent of the Sperner's Lemma pivoting algorithm [11], and induces a (directed) PPAD graph with nodes corresponding to the faces of the dual-graph board.

Even though Gale's proof induces a PPAD graph, there is no natural analogue of the EndOfTheLine problem for Hex. This is because the PPAD graph induced by Gale's argument only has four unbalanced vertices, corresponding to the entries A and B and the exits 1 and 2. The natural question is which pairs of unbalanced vertices are connected, which is equivalent to asking *"who won?"* after a game of Hex has filled the board. This corresponds to finding the *specific* unbalanced vertex of the PPAD graph which is connected to the starting one, corresponding to entry A. Of course, this would be polynomial-time solvable if the board were polynomially-sized, so we will assume an *exponentially-large* board where a polynomially-sized circuit tells us who claimed any particular tile.

5.2 WhoWonHex

WhoWonHex

Input: a Hex board, such that it takes n bits to uniquely specify a tile (the canonical way to do this is via the dual graph, by having $n/2$ bits dedicated to specifying the position of the tile in each dimension); a circuit C which takes an n-bit string (i.e. a tile) and outputs either 0 ('player 1 does not occupy this tile') or 1 ('player 1 does occupy this tile').

Output: 0 if there does not exist a bridge of tiles occupied by player 1 from the left facet to the right facet ('player 1 has not won'), and 1 if there does exist such a bridge ('player 1 has won'). Formally, with the canonical tile-specifying scheme described above, a 'bridge' is defined as a sequence of tiles $v_1, v_2, \ldots, v_m$ such that: (a) v_i is adjacent to v_{i+1} for all $i = 1, 2, \ldots, m-1$; (b) $C(v_i) = 1$ for all i; (c) the first $n/2$ bits of v_1 are all 0 (it's adjacent to the left facet), and the first $n/2$ bits of v_m are all 1 (it's adjacent to the right facet).

We want to show that WhoWonHex is PSPACE-complete (Theorem 4). It is tempting to try to prove this hardness via a reduction from the so-called OtherEndOfThisLine problem,[1] which is PSPACE-hard [8], by simulating paths in a given PPAD graph via "Gale paths." However, Gale paths do not cross, and the usual embedding of a PPAD graph into a 2-dimensional plane without crossings results in a drastic change of the graph topology (unbalanced vertices remain unbalanced vertices, but which unbalanced vertices are connected to each other changes) [2, 7]. Instead, we will obtain our result as a direct consequence of Goldberg's proof that it is PSPACE-hard to identify the solution output by a path-following algorithm for Sperner [7].

Goldberg's proof proceeds directly from the well-known PSPACE-complete problem of evaluating a *quantified Boolean formula* (QBF). Since our proof is an application of Goldberg's proof, we only explain how to make the gadgets required to modify his proof for our purposes.

Proof of Theorem 4. As described above, we will be adapting Goldberg's proof that identifying the solution chosen by the standard pivoting algorithm for Sperner is PSPACE-hard. (The inclusion into PSPACE is easy to establish; the details are given in the full version of our paper [1])). Given a QBF instance Goldberg defines horizontal and vertical wires in an exponentially large square lattice in such a way that the existence of a connected path of

[1] This problem is: given a PPAD graph via circuits P, N, identify the unbalanced vertex connected to 0^n, if 0^n is itself unbalanced.

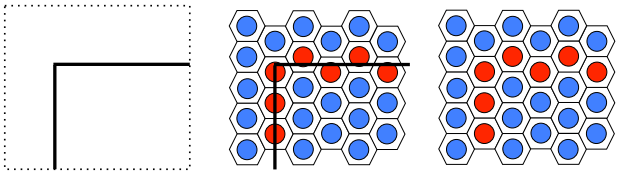


Figure 5 *Left:* A wire occupying part of the space. *Middle:* The wire, superimposed on the Hex board, with the appropriate tiles turned to red to carry the wire. *Right:* The wire, represented only as a sequence of red Hex tiles.

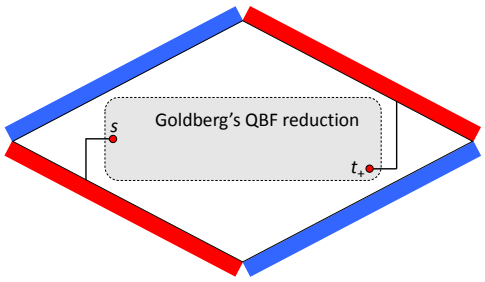


Figure 6 Embedding Goldberg's rectangular reduction into a Hex board, so that player 1 wins if and only if s is connected to t_+.

wires from a special point s to a special point t_+ (which themselves always lie on wires but may not be connected through wires) is equivalent to the QBF instance evaluating to 1. He also provides an efficient algorithm to determine if a wire passes through a specific point, and then simulates wires with paths in the PPAD graph of a Sperner instance.

Our result will be based on a 'wire' gadget that is based on Hex rather than Sperner – which is simply a connected path of red tiles 'insulated' from other red tiles by blue tiles (in fact, all tiles not on a 'wire' gadget are defined to be occupied by blue stones, i.e. player 2). For simplicity, we use only wires whose segments are perfectly vertical or horizontal; how a wire translates into tiles is shown in Figure 5.

Usefully the special points s and t_+ in Goldberg's construction lie on the left and right boundary of the square lattice respectively. This allows us to embed Goldberg's construction into a Hex board as shown in Figure 6. In particular:

- we allocate a part of the board where we replicate Goldberg's construction using our wire gadget, as described above given a QBF instance;
- we add wires connecting s to the bottom-left edge of the board;
- we add wires connecting t_+ to the up-right edge of the board.

Since s and t_+ lie on wires they are red tiles.

Since we do not have any wires not defined by Goldberg's construction (other than those from s and t_+ to the boundary of the Hex board shown in Figure 6), there is a bridge connecting the red edges of the Hex board if and only if s is connected to t_+ by wires inside Goldberg's construction. This is PSPACE-hard to determine, hence it is PSPACE-hard to determine if Player 1 wins. ◀

We remark that this proof had to be derived from Goldberg's *construction* itself, and not his result in a black-box manner. This is because the construction guarantees that the end-points s and t_+ in the proof above lie on the boundaries of the construction (and thus can be wired to the bottom-left and upper-right edges of the Hex board).

In contrast, general SPERNER instances may very well have the solution reachable through the standard path-following algorithm residing in the interior of the construction (thereby making it hard to translate into Hex, since the question in Hex is whether there is a bridge of red stones joining the left and right edges of the board).

6 Conclusion

In this paper, we explored the links between Brouwer's fixed-point theorem, the Jordan curve theorem, and the game of Hex. We showed that Brouwer and Jordan are mathematically and computationally equivalent, complementing Maehara's result that Jordan is a consequence of Brouwer. Combined with Gale's result that Brouwer's fixed-point theorem is mathematically equivalent to the seemingly innocuous Hex theorem (that a completed game of Hex must have a winner), our result implies that all three theorems, Brouwer, Jordan and Hex, are mathematically equivalent. Within PPAD, we defined two computational problems, CURVECROSSING and ZEROSURFACECROSSING, which always have solutions by dint of the Jordan curve theorem. We show that both lie in PPAD, and the second is also PPAD-complete. Finally, we relate the Hex theorem to results in the literature pertaining to the complexity of standard pivoting algorithms for ENDOFTHELINE, BROUWER and SPERNER. It has been shown that identifying the solution computed by standard pivoting algorithms for these problems is PSPACE-complete, and we show that the problem WHOWONHEX, of determining who is the winner in a play of Hex, is also PSPACE-complete. We thereby establish computational relations among Brouwer, Hex and Jordan both within PPAD and at the level of PSPACE. The main problem left open by our work is the complexity of CURVECROSSING. Is it PPAD-complete? We discuss structural properties of instances of CURVECROSSING that make us believe that the problem could lie lower in TFNP. It would be interesting to identify a potential function argument guaranteeing a solution to this problem, thereby placing it in the intersection of PLS and PPAD, and potentially one of the classes defined in [4].

References

1. Aviv Adler, Constantinos Daskalakis, and Erik Demaine. The Complexity of Hex and the Jordan Curve Theorem. *Arxiv*, 2016.
2. Xi Chen and Xiaotie Deng. On the Complexity of 2D Discrete Fixed Point Problem. In *the 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, 2006.
3. Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The Complexity of Computing a Nash Equilibrium. In *the 38th Annual ACM Symposium on Theory of Computing (STOC)*, 2006.
4. Constantinos Daskalakis and Christos H. Papadimitriou. Continuous local search. In *the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011.
5. Kousha Etessami and Mihalis Yannakakis. On the Complexity of Nash Equilibria and Other Fixed Points (Extended Abstract). In *the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2007.
6. David Gale. The game of Hex and the Brouwer fixed-point theorem. *American Mathematical Monthly*, pages 818–827, 1979.
7. Paul Goldberg. The Complexity of the Path-following Solutions of Two-dimensional Sperner/Brouwer Functions. *arXiv*, 2015.
8. Paul W Goldberg, Christos H Papadimitriou, and Rahul Savani. The Complexity of the Homotopy Method, Equilibrium Selection, and Lemke-Howson Solutions. *ACM Transactions on Economics and Computation*, 1(2):9, 2013.

9 Camille Jordan. *Cours d'analyse de l'École polytechnique*, volume 1. Gauthier-Villars et fils, 1893.

10 Ryuji Maehara. The Jordan curve theorem via the Brouwer fixed point theorem. *American Mathematical Monthly*, pages 641–643, 1984.

11 Christos H. Papadimitriou. On the Complexity of the Parity Argument and Other Inefficient Proofs of Existence. *Journal of Computer and System Sciences*, 48(3):498–532, 1994.

A Interpolations of Functions on the Unit Square

In order to develop computational problems relating to our theorems of interest, we need to give a formal definition of how circuits can represent functions on the unit square. We will consider two types of functions we wish to represent: $f : [0,1]^2 \to [-1,1]$, and $g : [0,1]^2 \to [0,1]^2$. They are represented by the following circuits:

1. f is represented by a circuit F which takes two n-bit strings as input (representing a point in $[0,1]^2$) and returns an m-bit string (representing the value of the function);
2. g is represented by a circuit G which takes two n-bit strings as input and returns two new n-bit strings.

These circuits directly define the values of their respective functions at lattice points in $[0,1]^2$, namely points whose coordinates are integer multiples of $1/(2^n-1)$; for the values of f and g at points which are not on this lattice, we use the triangulations depicted in Figure 7. For f, we take the output at lattice points (which is directly given by F) and use them to generate a 'mesh' which defines f at points not on the lattice; for g, we make a similar construction as shown at center and right in the figure.

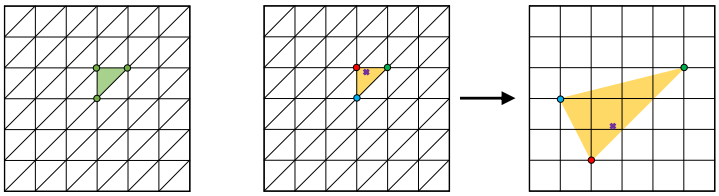


Figure 7 *Left:* the value of f at the three marked lattice points uniquely determines a plane; for input points in the triangle shared by the three, the output of f is consistent with this plane. *Center and Right:* the marked lattice points at right are the outputs of g at the marked lattice points at center; inputs in the triangle shared by them are mapped to the corresponding point in the triangle shared by their outputs (for example, the purple 'x' at center is mapped to the purple 'x' at right).

Fractals for Kernelization Lower Bounds, With an Application to Length-Bounded Cut Problems*

Till Fluschnik[†1], Danny Hermelin[‡2], André Nichterlein[§3], and Rolf Niedermeier[4]

1 **Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany**
till.fluschnik@tu-berlin.de

2 **Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Israel**
hermelin@bgu.ac.il

3 **Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany**
andre.nichterlein@tu-berlin.de

4 **Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Berlin, Germany**
rolf.niedermeier@tu-berlin.de

Abstract

Bodlaender et al.'s [6] cross-composition technique is a popular method for excluding polynomial-size problem kernels for NP-hard parameterized problems. We present a new technique exploiting triangle-based fractal structures for extending the range of applicability of cross-compositions. Our technique makes it possible to prove new no-polynomial-kernel results for a number of problems dealing with length-bounded cuts. Roughly speaking, our new technique combines the advantages of serial and parallel composition. In particular, answering an open question of Golovach and Thilikos [13], we show that, unless NP $\subseteq$ coNP / poly, the NP-hard Length-Bounded Edge-Cut problem (delete at most k edges such that the resulting graph has no s-t path of length shorter than ℓ) parameterized by the combination of k and ℓ has no polynomial-size problem kernel. Our framework applies to planar as well as directed variants of the basic problems and also applies to both edge and vertex deletion problems.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 [Discrete Mathematics] Graph Theory

Keywords and phrases Parameterized complexity, polynomial-time data reduction, cross-compositions, lower bounds, graph modification problems, interdiction problems

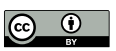
Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.25

* A full version of the paper is available at http://arxiv.org/abs/1512.00333.

† Till Fluschnik acknowledges support by the DFG, project DAMM (NI 369/13-2).

‡ Danny Hermelin was supported by a DFG Mercator fellowship within the project DAMM (NI 369/13-2) while staying at TU Berlin (August 2015). He has also received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11, and by the ISRAEL SCIENCE FOUNDATION (grant No. 551145/14).

§ André Nichterlein is from February 2016 to January 2017 on postdoctoral leave to Durham University (GB), funded by DAAD.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 25; pp. 25:1–25:14

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

1 Introduction

Lower bounds are of central concern all over computational complexity analysis. With respect to fixed-parameter tractable problems, currently there are two main streams in this context: (i) ETH-based lower bounds for the running times of exact algorithms [18] and (ii) lower bounds on problem kernel sizes; more specifically, the exclusion of polynomial-size problem kernels [17]. Both these research directions for lower bounds rely on plausible complexity-theoretic assumptions, namely the Exponential-Time Hypothesis (ETH) and NP $\not\subseteq$ coNP / poly, respectively. In this work, we contribute to the second research direction, developing a new technique that exploits a triangle-based fractal structure in order to exclude polynomial-size problem kernels (polynomial kernels for short) for edge and vertex deletion problems in the context of length-bounded cuts.

Kernelization is a key method for designing fixed-parameter algorithms [14, 17]; among all techniques of parameterized algorithm design, it has the presumably greatest potential for delivering practically relevant algorithms. Hence, it is of key interest to explore its power and its limitations. In a nutshell, the fundamental idea of kernelization is as follows. Given a parameterized problem instance I with parameter k, in polynomial time preprocess I by applying data reduction rules in order to simplify it and reduce it to an "equivalent" instance (so-called (problem) kernel) of the same problem. For NP-hard problems the best one can hope for is a problem kernel of size polynomial in the parameter k. In a way, one may interpret kernelization (requested to run in polynomial time) as an "exact counterpart" of polynomial-time approximation algorithms. Indeed, linear-size problem kernels often imply constant-factor approximation algorithms [20, page 15]. Approximation algorithmics has a highly developed theory (having produced concepts such as MaxSNP-hardness and the famous PCP theory) for proving (relative to some plausible complexity-theoretic assumption) lower bounds on the approximation factors [24].

It is fair to say that in the younger field of kernelization the arsenal for proving lower bounds (particularly excluding polynomial kernels) so far is of smaller scope and needs further development. The most influential result in this context is due to Bodlaender et al. [5] and Fortnow and Santhanam [12]: Based on the assumption NP $\not\subseteq$ coNP / poly, it is shown that e.g. the NP-hard graph problem Longest Path parameterized by solution size has no polynomial kernel. The core tool for showing this are so-called "OR-compositions". To ease the use of this kernel-lower-bound framework, one natural idea is to use "polynomial parameter transformations", that is, a form of "parameter-preserving reductions" [7, 10]. An easier-to-use generalization of the OR-composition technique is given by so-called OR-cross-compositions [6]. Currently, these two approaches constitute the known core tools to exclude polynomial kernels. Building on OR-cross-compositions, we add a further tool (which we baptized "fractalism") in order to extend the range of problems to be addressed by OR-cross-compositions. The usefulness of our new technique is substantiated by resolving an open problem posed by Golovach and Thilikos [13], here specifically applying our technique to the NP-hard Length-Bounded Edge-Cut problem.

Next, we discuss in some more detail OR-(cross)-compositions. Roughly speaking, the idea behind an OR-composition for a parameterized problem is to encode the logical "or" of t instances with parameter value k into a single instance of the same problem with parameter value $k' = k^{O(1)}$. In particular, given t instances, the obtained instance is a yes-instance if and only if at least one of the given instances is a yes-instance. If an OR-composition is possible, then this excludes polynomial kernels. Whereas in OR-compositions one combines instances of an NP-hard parameterized problem into one instance of a parameterized problem,

in OR-cross-compositions one combines instances of classical NP-hard problems into one instance of a parameterized problem (see Section 2 for details and formal definitions).

While for some problems, for example Longest Path with parameter solution size [5], a simple disjoint union yields the desired OR-composition, other problems seem to require involved constructions, for example Set Cover with parameter universe size [10]. Indeed, devising a cross-composition can be quite challenging and the task becomes even harder when considering several, seemingly orthogonal parameterizations at once. To illustrate the problem with such combined parameters, let us consider the problem Length-Bounded Edge-Cut (LBEC). Herein, an undirected graph $G = (V, E)$ with $s, t \in V$, and two integers $k, \ell \in \mathbb{N}$ are given, and the question is whether it is possible to delete at most k edges such that the shortest s-t path is of length at least ℓ. Using a simple branching algorithm, one can show that LBEC(k, ℓ) is fixed-parameter tractable for the combined parameter (k, ℓ) [13, 3]. To exclude the existence of a polynomial kernel for LBEC(k, ℓ), we would like to apply the OR-cross-composition framework to the problem, and as a natural candidate for the input problem we decide for LBEC itself.

A standard approach to applying the OR-cross-composition to a problem like LBEC would be to concatenate the input instances on the source and sink vertices, also referred to as "serial" composition. To this end, one needs some additional gadgets to ensure that only in one instance edges are deleted. This form of composition, however, induces a dependency of the second parameter ℓ on the number of instances, which is not allowed. Another standard approach is introducing a "global" sink and source vertex, and connecting all source vertices with the global source and all sink vertices, also referred to as a "parallel" composition. This form of composition would keep ℓ small enough, but induces a dependency of the first parameter k on the number of instances. Summarizing, the parameter k seems to ask for a serial composition and the parameter ℓ seems to ask for a parallel composition. For some problems using a tree as "instance selector" was helpful, see for example Bevern et al. [4] or Bazgan et al. [2]. The problem with trees is that they introduce small (constant-size) s-t cuts, which is problematic for Length-Bounded Edge-Cut. In this work, we introduce a fractal structure as instance selector which has the nice properties of trees but does not introduce small cuts. So, our fractal structure helps to exclude polynomial kernels for several problems.

Our contributions. Our main technical contribution is to introduce a family of graphs that we call T-fractals and that build on triangles. T-fractals feature a fractal-like structure, in the sense of self-similarity and scale-invariance. Using these T-fractals in OR-cross-compositions, we show that the following parameterized graph modification problems and several of their variants do not admit polynomial kernels (unless NP $\subseteq$ coNP / poly):

- Length-Bounded Edge-Cut(k, ℓ) (LBEC(k, ℓ)), where k is the number of edges to delete, and ℓ is the lower bound on the length of the shortest path.
- Minimum Diameter Edge Deletion(k, ℓ) (MDED(k, ℓ)), that is, given an undirected connected graph $G = (V, E)$ and two integers k, ℓ (the parameters), decide whether there are at most k edge deletions such that the remaining graph remains connected and has diameter at least ℓ.
- Directed Small Cycle Transversal(k, ℓ) (DSCT(k, ℓ)), that is, given a directed graph $G = (V, E)$ and two integers k, ℓ (the parameters), decide whether there are at most k edge deletions such that the remaining graph has no cycle of length smaller than ℓ.

Table 1 surveys our no-polynomial-kernel results and spots an open question.

Table 1 Survey of the concrete results of this paper (under the assumption that NP $\not\subseteq$ coNP / poly). PK stands for polynomial kernel and a "?" indicates that it is open whether a polynomial kernel exists. We remark that the no-polynomial-kernel results for LBEC(k, ℓ) on directed graphs still hold for directed acyclic graphs. Note that we claim without proof that, except for the planar variants, our proofs also transfer to the vertex deletion case, both for directed and undirected graphs.

Problem	edge deletion			
	directed		undirected	
	planar	general	planar	general
LBEC(k, ℓ)	No PK [Thm. 12]	No PK [Thm. 12]	No PK [Thm. 12]	No PK [Thm. 11]
MDED(k, ℓ)	No PK [Thm. 13]	No PK [Thm. 13]	No PK [Thm. 13]	No PK [Thm. 13]
DSCT(k, ℓ)	No PK [Thm. 14]	No PK [Thm. 14]	PK [25]	?

2 Preliminaries

Graph Theory. Let $G = (V, E)$ be a graph. For $C \subseteq V(G)$ ($C \subseteq E(G)$) we write $G - C$ for the graph G where all vertices (edges) in C are deleted. Let $s, t \in V(G)$. An edge set $C \subseteq E(G)$ is an *s-t edge cut* in G if the vertices s and t are disconnected in $G - C$. An s-t edge cut C is called *minimal* if $C \backslash \{e\}$ is not an s-t edge cut in G for all $e \in C$. An s-t edge cut C is called *minimum* if there is no s-t edge cut C' in G such that $|C'| < |C|$.

The length of a path (cycle) is the number of edges in the path (cycle). An s-t path is a path where the vertices s and t are the endpoints of the path. In directed graphs, an s-t path is a path where all arcs are directed toward t, and a cycle is a connected graph where every vertex has outdegree and indegree exactly one. The *diameter* of a graph G is the maximum length of any shortest v-w path over all $v, w \in V(G)$, $v \neq w$.

For $v, w \in V(G)$, we say we *merge* the vertices v and w if we add a new vertex vw to V as well as the edge set $\{\{vw, x\} \mid \{x, v\} \in E\} \cup \{\{vw, x\} \mid \{x, w\} \in E\}$ to E, and we delete the vertices v and w and all edges incident to v and w.

Parameterized Complexity. A *parameterized problem* is a set of instances $(\mathcal{I}, k)$ where $\mathcal{I} \in \Sigma^*$ for a finite alphabet Σ, and $k \in \mathbb{N}$ is the *parameter*. A parameterized problem L is *fixed-parameter tractable (fpt)* if it can be decided in $f(k) \cdot |\mathcal{I}|^{O(1)}$ time whether $(\mathcal{I}, k) \in L$, where f is a computable function only depending on k. We say that two instances $(\mathcal{I}, k)$ and $(\mathcal{I}', k')$ of parameterized problems P and P' are *equivalent* if $(\mathcal{I}, k)$ is YES for P if and only if $(\mathcal{I}', k')$ is YES for P'. A *kernelization* is an algorithm that, given an instance $(\mathcal{I}, k)$ of a parameterized problem P, computes in polynomial time an equivalent instance $(\mathcal{I}', k')$ of P (the *kernel*) such that $|\mathcal{I}'| + k' \leq f(k)$ for some computable function f only depending on k. We say that f measures the *size* of the kernel, and if $f \in k^{O(1)}$, we say that P admits a polynomial kernel. We remark that a decidable parameterized problem is fixed-parameter tractable if and only if it admits a kernel [8].

Given an NP-hard problem L, an equivalence relation $\mathcal{R}$ on the instance of L is a *polynomial equivalence relation* if (i) one can decide for any two instances in time polynomial in their sizes whether they belong to the same equivalence class, and (ii) for any finite set S of instances, $\mathcal{R}$ partitions the set into at most $(\max_{x \in S} |x|)^{O(1)}$ equivalence classes.

▶ **Definition 1.** Given an NP-hard problem L, a parameterized problem P, and a polynomial equivalence relation $\mathcal{R}$ on the instances of L, an *OR-cross-composition* of L into P (with respect to $\mathcal{R}$) is an algorithm that takes ℓ $\mathcal{R}$-equivalent instances $\mathcal{I}_1, \ldots, \mathcal{I}_\ell$ of L and constructs

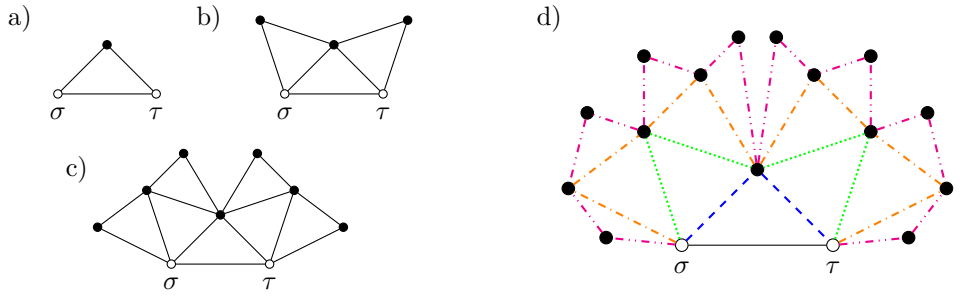


Figure 1 T-fractals a) $\triangle_1$, b) $\triangle_2$, c) $\triangle_3$, and d) $\triangle_4$. The two special vertices σ and τ are highlighted by empty circles. In $\triangle_4$ the different boundaries are highlighted by line-types (solid: boundary B_0; dashed: boundary B_1; dotted: boundary B_2; dash-dotted: boundary B_3; dash-dot-dotted: boundary B_4).

in time polynomial in $\sum_{i=1}^{\ell} |\mathcal{I}_\ell|$ an instance $(\mathcal{I}, k)$ such that
1. k is polynomially upper-bounded in $\max_{1\leq i\leq \ell} |\mathcal{I}_i| + \log(\ell)$ and
2. $(\mathcal{I}, k)$ is YES for P if and only if there is at least one $\ell' \in [\ell]$ such that $\mathcal{I}_{\ell'}$ is YES for L.

If a parameterized problem P admits an OR-cross-composition for some NP-hard problem L, then P does not admit a polynomial kernel with respect to its parameterization, unless NP $\subseteq$ coNP / poly [6]. We remark that we can assume that $\ell = 2^j$ for some $j \in \mathbb{N}$ since we can add trivial NO-instances from the same equivalence class to reach a power of two. We refer to the survey of Kratsch [17] for an overview on kernelization and lower bounds.

3 The "Fractalism" Technique

In this section, we describe our new technique based on *triangle fractals* (*T-fractals* for short). We provide a general construction scheme for cross-compositions using T-fractals. To this end, we first define T-fractals and then discuss several of their properties in Section 3.1. Subsequently, in Section 3.2 we present a "construction manual" for an application of T-fractals in cross-compositions.

Roughly speaking, a T-fractal can be constructed by iteratively putting triangles on top of each other, see Figure 1 for four examples.

▶ **Definition 2.** For $q \geq 1$, the *q-T-fractal* $\triangle_q$ is the graph constructed as follows:
(1) Set $\triangle_0 := \{\sigma, \tau\}$ with $\{\sigma, \tau\}$ being a "marked edge" with endpoints σ and τ, subsequently referred to as special vertices.
(2) Let F be the set of marked edges.
(3) For each edge $e \in F$, add a new vertex and connect it by two new edges with the endpoints of e, and mark the two added edges.
(4) Unmark all edges in F.
(5) Repeat (2)–(4) $q - 1$ times.

The fractal structure of $\triangle_q$ might be easier to see when considering the following equivalent recursive definition of $\triangle_q$: For the base case we define $\triangle_0 := \{\sigma, \tau\}$ as in Definition 2. Then, the q-T-fractal $\triangle_q$ is constructed as follows. Take two $(q-1)$-T-fractals $\triangle'_{q-1}$ and $\triangle''_{q-1}$, where σ', τ' and σ'', τ'' are the special vertices of $\triangle'_{q-1}$ and $\triangle''_{q-1}$, respectively. Then $\triangle_q$ is obtained by merging the vertices τ' and σ'' and adding the edge $\{\sigma', \tau''\}$. Set $\sigma = \sigma'$ and $\tau = \tau''$ as the special vertices of $\triangle_q$. We remark that we make use of the recursive structure in later proofs.

In the ith execution of (2)–(4) in Definition 2, we obtain 2^{i-1} many triangles. We say that these triangles have depth i. The *boundary* $B_i \subseteq E(\triangle_q)$, $i \in [q]$, are those edges of the triangles of depth i which are not edges of the triangles of depth $i-1$. As a convention, the edge $\{\sigma, \tau\}$ connecting the two special vertices σ and τ forms the boundary B_0. Refer to Figure 1 for an illustration of the boundaries in the T-fractal $\triangle_4$. Moreover, by construction, we obtain the following:

▶ **Observation 3.** *In every T-fractal, each boundary forms a σ-τ path, and all boundaries are pairwise edge-disjoint.*

Note that the boundary B_q contains $p = 2^q$ edges. Thus, the number of edges in $\triangle_q$ is $\sum_{i=0}^{q} 2^i = 2^{q+1} - 1 = 2 \cdot p - 1$. Further observe that all vertices of $\triangle_q$ are incident with the edges in B_q, and B_q forms a σ-τ path. Hence, $\triangle_q$ contains $p+1$ vertices.

Reducing the Weighted to the Unweighted Case. In the remainder of the paper, we focus on the unweighted case of T-fractals without multiple edges or loops. This is possible due to the following reduction of the weighted to the unweighted case. Equip the T-fractal with an *edge cost*, that is, the cost for deleting any edge in the T-fractal. If $c \in \mathbb{N}$ is the edge cost of $\triangle_q$, then we write $\triangle_q^c$ (we drop the superscript if $c = 1$). To reduce to the case with an unweighted, simple graph, we add $c-1$ further copies for each edge. Thus, to make two adjacent vertices non-adjacent, it requires c edge-deletions. To make the graph simple, we subdivide each edge. We remark that in this way we double the distances of the vertices in the original T-fractal. Thus, whenever we consider distances in the fractal with edge cost and the graph obtained by the reduction above, we have to take into account a factor of two.

3.1 Properties of T-Fractals

The goal of this subsection is to prove several properties of T-fractals that are used in later constructions. Some key properties of T-fractals appear in the context of σ-τ edge cuts in $\triangle_q$.

The minimum edge cuts in $\triangle_q$ will play a central role when using T-fractals in cross-compositions since the minimum edge cuts serve as instance selectors (see Section 3.2). First, we discuss the size and structure of the minimum edge cuts in $\triangle_q$.

▶ **Lemma 4.** *Every minimum σ-τ edge cut in $\triangle_q$ is of size $q+1$.*

Proof. Let C be a minimum σ-τ edge cut in $\triangle_q$. Note that the degrees of σ and τ are exactly $q+1$, and thus $|C| \leq q+1$. Moreover, the boundaries in $\triangle_q$ are pairwise edge-disjoint and each boundary forms a σ-τ path (Observation 3). Since $\triangle_q$ contains $q+1$ boundaries, it follows that there are at least $q+1$ disjoint σ-τ paths in $\triangle_q$. Menger's theorem [21] states that in a graph with distinct source and sink, the maximum number of disjoint source-sink paths equals the minimum size of any source-sink edge cut. Thus, by Menger's theorem, it follows that $|C| \geq q+1$. Hence $|C| = q+1$. ◀

From the fact that the boundaries are pairwise edge-disjoint and each boundary forms a σ-τ path, we can immediately derive the following from Lemma 4.

▶ **Corollary 5.** *Every minimum σ-τ edge cut in $\triangle_q$ contains exactly one edge of each boundary.*

In the following we describe a (hidden) dual structure in $\triangle_q$, that is, a complete binary tree with p leaves. We refer to Figure 2 for an example of the dual structure in $\triangle_3$. To talk

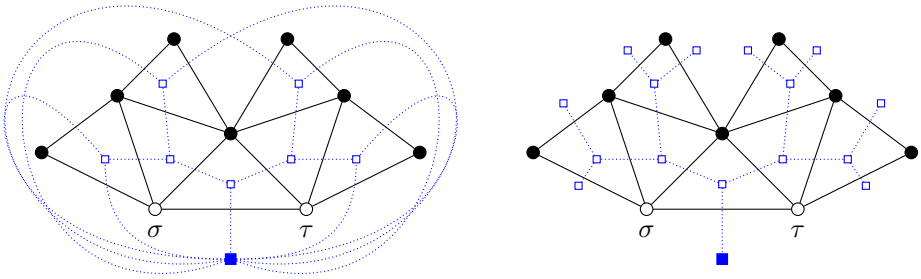


Figure 2 Left: The T-fractal $\triangle_3$ (circles and solid lines) and its dual graph (squares and dotted lines). The filled square is the vertex dual to the outer face in the dual graph. Right: The T-fractal $\triangle_3$ (circles and solid lines) and its dual structure T_3, illustrated by squares and dotted lines, where the filled square corresponds to the root of the dual structure.

about the dual structure by means of duality of plane graphs, we need a plane embedding of $\triangle_q$. Hence we assume that $\triangle_q$ is embedded as in Figure 1 (iteratively extended). By T_q we denote the dual structure in $\triangle_q$, where the vertex dual to the outer face is replaced by $p+1$ vertices (*split vertices*) such that each edge incident with the dual vertex is incident with exactly one split vertex. We consider the split vertex incident with the vertex dual to the triangle containing the edge $\{\sigma, \tau\}$ as the root vertex of the dual structure T_q. Thus, the other split vertices correspond to the leaves of the dual structure T_q. Note that the depth of a triangle one-to-one corresponds to the depth of the dual vertex in T_q.

Observe that there is a one-to-one correspondence between the edges in T_q and the edges in $\triangle_q$. The following lemma states duality of root-leaf paths in T_q and minimum σ-τ edge cuts in $\triangle_q$, demonstrating the utility of the dual structure T_q.

▶ **Lemma 6.** *There is a one-to-one correspondence between root-leaf paths in the dual structure T_q of $\triangle_q$ and minimum σ-τ edge cuts in $\triangle_q$. Moreover, there are exactly $p = 2^q$ pairwise different minimum σ-τ edge cuts in $\triangle_q$.*

Proof. Observe that each path from the root to a leaf in the dual structure T_q corresponds to a cycle in the dual graph. It is well-known that there is a one-to-one correspondence between minimal edge cuts in a plane graph and cycles in its dual graph [9, Proposition 4.6.1]. Herein, every cycle in the dual graph that "cuts" the edge $\{\sigma, \tau\}$ in $\triangle_q$ is a root-leaf path in T_q. Thus, the only minimal σ-τ edge cuts are those corresponding to the root-leaf paths. By the one-to-one correspondence of the depth of the triangles in $\triangle_q$ and the depth of the vertices in T_q, these edge cuts are of cardinality $q+1$. Hence, by Lemma 4, these edge cuts are minimum edge cuts.

Since $|B_q| = p$, there are exactly p leaves in T_q, and thus there are exactly p different root-leaf paths in T_q. It follows that the number of pairwise different minimum σ-τ edge cuts in $\triangle_q$ is exactly $p = 2^q$. ◀

Further, we obtain the following.

▶ **Lemma 7.** *Let C be a minimum σ-τ edge cut in $\triangle_q$. Let $\{x, y\} = C \cap B_q$, where x is in the same connected component as σ in $\triangle_q - C$. Then $\mathrm{dist}(\sigma, x) + \mathrm{dist}(y, \tau) = q$ in $\triangle_q - C$.*

Proof. We prove the lemma by induction on q. For the base case $q = 0$, observe that $C = \{\sigma, \tau\}$ and $\mathrm{dist}_{\triangle_0 - C}(\sigma, x) + \mathrm{dist}_{\triangle_0 - C}(y, \tau) = 0$.

For the induction step, assume that the statement of the lemma is true for $\triangle_{q-1}$. Now, let C be a minimum σ-τ edge cut in $\triangle_q$. Hence, $\{\sigma, \tau\} \in C$. Denote by u the (unique)

vertex that is adjacent to the two special vertices σ and τ. Let $\triangle'_{q-1}$ and $\triangle''_{q-1}$ be the two $(q-1)$-T-subfractals of $\triangle_q$, so that $\triangle'_{q-1}$ ($\triangle''_{q-1}$) has the special vertices σ and u (u and τ). By Lemma 6, the minimum σ-τ edge cut C corresponds to a root-leaf path in T_q. Hence, $C' := C \setminus \{\sigma, \tau\}$ is either a subset of $E(\triangle'_{q-1})$ or of $E(\triangle''_{q-1})$. Assume w.l.o.g. that $C' \subseteq E(\triangle'_{q-1})$. It follows from the induction hypothesis that $\mathrm{dist}_{\triangle'_{q-1}-C'}(\sigma, x) + \mathrm{dist}_{\triangle'_{q-1}-C'}(y, u) = q - 1$. Since $\mathrm{dist}_{\triangle_q - C}(y, \tau) = \mathrm{dist}_{\triangle'_{q-1}-C'}(y, u) + 1$, it follows that $\mathrm{dist}_{\triangle_q - C}(\sigma, x) + \mathrm{dist}_{\triangle_q - C}(y, \tau) = q$. ◀

▶ Remark. By an inductive proof like the one of Lemma 7, one can easily show that the maximum degree Δ of $\triangle_q$ is exactly $2 \cdot q$ for $q > 0$. Moreover, due to Lemma 7, the diameter of $\triangle_q$ is bounded in $O(q)$.

Another observation on $\triangle_q$ is that any deletion of d edges increases the length of any shortest σ-τ path to at most $d + 1$, unless the edge deletion forms a σ-τ edge cut.

▶ **Lemma 8.** *Let $D \subseteq E(\triangle_q)$ be a subset of edges of $\triangle_q$. If D is not a σ-τ edge cut, then there is a σ-τ path of length at most $|D| + 1$ in $\triangle_q - D$.*

Proof. We prove the statement of the lemma by induction on q. For the induction base with $q = 0$, observe that since D is not a σ-τ edge cut, it follows that $D = \emptyset$ and, hence, σ and τ have distance one.

For the induction step, assume that the statement of the lemma is true for $\triangle_{q-1}$. Now, let $D \subseteq E(\triangle_q)$ be a subset of edges of $\triangle_q$ such that D is not a σ-τ edge cut. If $\{\sigma, \tau\} \notin D$, then there is a σ-τ path of length one and the statement of the lemma holds. Now consider the case $\{\sigma, \tau\} \in D$. Denote by u the (unique) vertex that is adjacent to the two special vertices σ and τ. If $\{\sigma, \tau\} \in D$, then every σ-τ path in $\triangle_q - D$ contains u and hence $\mathrm{dist}_{\triangle_q - D}(\sigma, \tau) = \mathrm{dist}_{\triangle_q - D}(\sigma, u) + \mathrm{dist}_{\triangle_q - D}(u, \tau)$. (If there is no σ-u-path or no u-τ-path in $\triangle_q - D$, then D is a σ-τ edge cut; a contradiction to the assumption of the lemma.) Now let $\triangle'_{q-1}$ and $\triangle''_{q-1}$ be the two $(q-1)$-T-subfractals of $\triangle_q$, so that $\triangle'_{q-1}$ ($\triangle''_{q-1}$) has the special vertices σ and u (u and τ). It follows that D can be partitioned into $D = D' \cup D'' \cup \{\sigma, \tau\}$ with $D' \subseteq E(\triangle'_{q-1})$ and $D'' \subseteq E(\triangle''_{q-1})$. By induction hypothesis, it follows that there is a σ-u path of length at most $|D'| + 1$ in $\triangle'_{q-1} - D'$ and a u-τ path of length at most $|D''| + 1$ in $\triangle''_{q-1} - D''$. Hence, there is a σ-τ path of length at most $|D'| + |D''| + 2 = |D| + 1$ in $\triangle_q - D$. ◀

We remark that there is a directed version of T-fractals. Herein, the edges of a T-fractal are directed in such a way that each boundary forms a directed σ-τ path. Note that the obtained graph is acyclic, and σ has no incoming arcs, and τ has no outgoing arcs. We further remark that all stated lemmas also hold for the directed T-fractal. The dual structure of a directed T-fractal is defined as the dual structure of the underlying undirected T-fractal. For more details we refer to the full version.

3.2 Application Manual for T-Fractals

The aim of this subsection is to provide a general guideline on how to use T-fractals in cross-compositions to obtain kernel lower bounds.

▶ **Construction 9.** *Given $p = 2^q$ instances $\mathcal{I}_1, \ldots, \mathcal{I}_p$ of an NP-hard graph problem, where each instance $\mathcal{I}_i$ has a unique source vertex s_i and a unique sink vertex t_i.*

(i) *Equip $\triangle_q^c$ with some "appropriate" edge cost $c \in \mathbb{N}$.*

(ii) *Let $v_0, \ldots, v_p$ be the vertices of the boundary B_q, labeled by their distances to σ in the σ-τ path corresponding to B_q (observe that $v_0 = \sigma$ and $v_p = \tau$).*

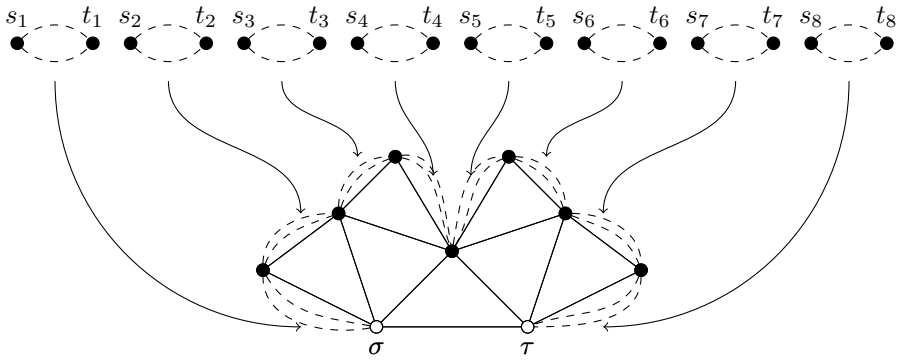


Figure 3 Illustration of Construction 9 with $p = 2^3 = 8$. The vertices $s_1, \ldots, s_8$ indicate the source vertices in the eight input instances, and $t_1, \ldots, t_8$ indicate the sink vertices in the eight input instances. We use dashed lines to sketch the input graphs.

(iii) *Incorporate each of the p graphs of the input instances into $\triangle_q^c$ as follows: for each $i \in [p]$, merge s_i with vertex v_{i-1} in $\triangle_q^c$ and merge t_i with v_i in $\triangle_q^c$.*

Refer to Figure 3 for an illustrative example of Construction 9.

In Construction 9, the T-fractal works as an instance selector by deleting edges corresponding to a minimum edge cut, which, by Lemma 4, is of size $q+1$. Hence, each minimum edge cut costs $c \cdot (q+1)$. The idea is that if we choose an appropriate value for c (larger than the budget in the instances $\mathcal{I}_1, \ldots, \mathcal{I}_p$) and an appropriate budget in the composed instance (e. g. $c \cdot (q+1)$ plus the budget in the instances $\mathcal{I}_1, \ldots, \mathcal{I}_p$), then we can only afford to delete at most $q+1$ edges in $\triangle_q^c$. Furthermore, if the at most $q+1$ edges chosen to be deleted do not form a minimum σ-τ edge cut in $\triangle_q^c$, then, by Lemma 8, the shortest σ-τ path has length at most $q+2$. Thus, by requiring in the composed instance that σ and τ have distance more than $q+2$, we enforce that any solution for the composed instance contains a minimum σ-τ edge cut in $\triangle_q^c$. By Lemma 6, each such minimum edge cut corresponds to one root-leaf path in the dual structure T_q of $\triangle_q^c$. Observe that each leaf in the dual structure of $\triangle_q^c$ one-to-one corresponds to an attached source instance. Hence, with an appropriate choice of c, the budget in the composed instance, and the required distance between σ and τ, the T-fractal ensures that one instance is "selected". We say that a minimum σ-τ edge cut in $\triangle_q^c$ *selects* an instance $\mathcal{I}$ if the edge cut corresponds to the root-leaf path with the leaf corresponding to instance $\mathcal{I}$.

▶ **Observation 10.** *Every minimum edge cut C in $\triangle_q^c$ selects exactly one instance $\mathcal{I}$. Conversely, every instance $\mathcal{I}$ can be selected by exactly one minimum edge cut.*

We use Construction 9 in OR-cross-compositions to rule out the existence of polynomial kernels. We call this approach *fractalism*. In particular, we provide the source and the target problem, appropriate values for the edge cost c and the budget in the composed instance, and the required distance between the special vertices σ and τ.

We remark that there is a similar construction for directed graph problems using the directed T-fractal. Moreover, the construction with directed acyclic input graphs yields a directed acyclic graph. For more details, we refer to the full version.

4 Applications to Length-Bounded Cut Problems

In this section, we rule out the existence of polynomial kernels for several problems (and their variants) under the assumption that NP $\not\subseteq$ coNP / poly. To this end, we combine the framework of OR-cross-compositions with our fractalism technique as described in Section 3.2.

4.1 Length-Bounded Edge-Cut

Our first fractalism application is the Length-Bounded Edge-Cut problem [1], also known as the problem of finding bounded edge undirected cuts [13], or the Shortest Path Most Vital Edges problem [19, 3].

Length-Bounded Edge-Cut (LBEC)
Input: An undirected graph $G = (V, E)$, with $s, t \in V$, and two integers k, ℓ.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that $\text{dist}_{G-F}(s, t) \geq \ell$?

The problem is NP-complete [16] and fixed-parameter tractable with respect to (k, ℓ) [13]. If k is at least the size of any s-t edge cut, then the problem becomes polynomial-time solvable by simply computing a minimum s-t edge cut. Thus, throughout this section, we assume that k is smaller than the size of any minimum s-t edge cut. The generalized problem where each edge is equipped with positive length remains NP-hard even on series-parallel and outerplanar graphs [1]. The directed variant with positive edge lengths remains NP-hard on planar graphs where the source and the sink vertex are incident to the same face [22]. Recently, Dvořák and Knop [11] showed that the problem can be solved in polynomial time on graphs of bounded treewidth. Here, we answer an open question [13] concerning the existence of a polynomial kernel with respect to the combined parameter (k, ℓ).

▶ **Theorem 11.** *Unless* NP $\subseteq$ coNP / poly*,* Length-Bounded Edge-Cut *parameterized by* (k, ℓ) *does not admit a polynomial kernel.*

Proof. We OR-cross-compose $p = 2^q$ instances of LBEC into one instance of LBEC(k', ℓ'). An instance $(G_i, s_i, t_i, k_i, \ell_i)$ of LBEC is called *bad* if $\max\{k_i, \ell_i\} > |E(G_i)|$ or $\min\{k_i, \ell_i\} < 0$. We define the polynomial equivalence relation $\mathcal{R}$ on the instances of LBEC as follows: two instances $(G_i, s_i, t_i, k_i, \ell_i)$ and $(G_j, s_j, t_j, k_j, \ell_j)$ of LBEC are $\mathcal{R}$-equivalent if and only if $k_j = k_i$ and $\ell_j = \ell_i$, or both are bad instances. Clearly, the relation $\mathcal{R}$ fulfills condition (i) of an equivalence relation (see Section 2). Observe that the number of equivalence classes of a finite set of instances is upper-bounded by the maximal size of a graph over the instances, hence condition (ii) holds. Thus, we consider p $\mathcal{R}$-equivalent instances $\mathcal{I}_i := (G_i, s_i, t_i, k, \ell)$, $i = 1, \ldots, p$. We remark that we can assume that $\ell \geq 3$, since otherwise LBEC is solvable in polynomial time by counting all edges connecting the source with the sink vertex. We OR-cross-compose into one instance $\mathcal{I} := (G, s, t, k', \ell')$ of LBEC(k', ℓ') with $k' = k^2 \cdot (\log(p) + 1) + k$ and $\ell' = \ell + \log(p)$ as follows.

Construction: Apply Construction 9 with edge cost $c = k^2$. In addition, set $s := \sigma$ and $t := \tau$. Let G denote the obtained graph.

Correctness: We show that $\mathcal{I}$ is a YES-instance if and only if there exists an $i \in [p]$ such that $\mathcal{I}_i$ is a YES-instance.

"⇐": Let $i \in [p]$ be such that $\mathcal{I}_i$ is YES. Following Observation 10 in Section 3.2, let C be the minimum s-t cut in $\triangle_q^c$ that selects instance $\mathcal{I}_i$. Recall that C is of size $q+1$ and that the edge cost equals k^2. Thus, the minimum s-t cut C has cost $(q+1) \cdot k^2 = (\log(p)+1) \cdot k^2$.

Note that after deleting the edges in C, the vertices s and t are only connected via paths through the incorporated graph G_i. Since $\mathcal{I}_i$ is YES, we can delete k edges (equal to the remaining budget) such that the distance of s_i and t_i in G_i is at least ℓ. Together with Lemma 7 in Section 3.1, such an additional edge deletion increases the length of any shortest s-t path in G to at least $\ell + \log(p) = \ell'$. Hence, $\mathcal{I}$ is a YES-instance.

"⇒": Suppose that one can delete at most k' edges in G such that each s-t path is of length at least ℓ'. Since the budget allows $\log(p)+1$ edge-deletions in $\triangle_q^c$, by Lemma 8 in Section 3.1, if we do not cut s and t in $\triangle_q^c$, then there is an s-t path of length $\log(p)+2$. Since $\ell \geq 3$, such an edge deletion does not yield a solution. Thus, in every solution of $\mathcal{I}$, a subset of the deleted edges forms a minimum s-t edge cut in $\triangle_q^c$ and thus, by Observation 10, selects an input instance.

Consider an arbitrary solution to $\mathcal{I}$, that is, an edge subset of $E(G)$ of cardinality at most k' whose deletion increases the shortest s-t path to at least ℓ'. Let $\mathcal{I}_i$, $i \in [p]$, be the selected instance. Note that any shortest s-t path contains edges in the selected instance $\mathcal{I}_i$. By Lemma 7, we know that the length of the shortest s-s_i path and the length of the shortest t_i-t path sum up to exactly $\log(p)$. It follows that the remaining budget of k edge deletions is spent in G_i in such a way that there is no path from s_i to t_i of length smaller than ℓ in G_i. Hence, $\mathcal{I}_i$ is a YES-instance. ◀

Using LBEC on planar graphs and on planar directed acyclic graphs as the input problem, fractalism yields the following (the proof is deferred to the full version).

▶ **Theorem 12.** *Unless* NP $\subseteq$ coNP / poly, LENGTH-BOUNDED EDGE-CUT *on planar undirected graphs as well as on planar directed acyclic graphs parameterized by* (k, ℓ) *does not admit a polynomial kernel.*

4.2 Further Applications

We present two further problems (and their variants) to which the fractalism technique applies. First, we consider the following NP-hard [23] problem.

MINIMUM DIAMETER EDGE DELETION (MDED)
Input: A connected, undirected graph $G = (V, E)$, two integers k, ℓ.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that $G - F$ remains connected and $\operatorname{diam}(G - F) \geq \ell$?

Note that MDED on directed strongly connected graphs asks for a subset of arcs such that the graph after deleting the arcs remains strongly connected and has diameter at least ℓ. Our second NP-hard [15] problem is the following.

DIRECTED SMALL CYCLE TRANSVERSAL (DSCT)
Input: A directed graph $G = (V, E)$, two integers k, ℓ.
Question: Is there a subset $F \subseteq E$ of cardinality at most k such that there is no induced directed cycle of length at most ℓ in $G - F$?

Both problems are fixed-parameter tractable with respect to (k, ℓ) (see full version). The fractalism technique yields the following (the proofs are deferred to the full version).

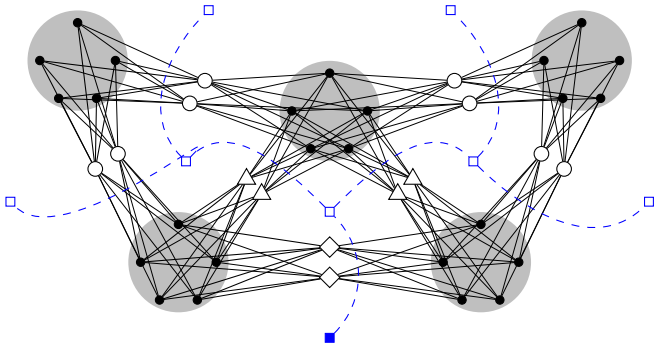


Figure 4 The vertex deletion variant $\triangle_2^{2;5}$ of T-fractals. Vertex types: empty diamonds belong to the boundary B_0, empty triangles belong to the boundary B_1, empty circles belong to the boundary B_2. The squares and dashed lines indicate the dual structure, where the filled square corresponds to the root. We highlighted vertices in gray-filled circles that correspond to the vertices in the edge-deletion variant $\triangle_2$.

▶ **Theorem 13.** *Unless* NP $\subseteq$ coNP / poly, Minimum Diameter Edge Deletion *on undirected, connected, planar graphs and on directed, strongly connected, planar graphs parameterized by* (k, ℓ) *does not admit a polynomial kernel.*

▶ **Theorem 14.** *Unless* NP $\subseteq$ coNP / poly, Directed Small Cycle Transversal *on planar directed graphs parameterized by* (k, ℓ) *does not admit a polynomial kernel.*

Like in the proof of Theorem 11, in the proofs of Theorems 13 and 14 we use LBEC as input problem and compose instances of LBEC using a T-fractal. The main difference is that we slightly modify the T-fractal. Roughly speaking, for MDED(k, ℓ) we append "long enough" paths to σ and τ, with endpoints σ' and τ' different to σ and τ. Those paths ensure that the only two vertices that can yield the required diameter are σ' and τ'. For DSCT(k, ℓ) we add the arc (τ, σ) to the directed T-fractal. Recall that the directed T-fractal is acyclic, and thus, every directed cycle in the composed graph contains the arc (τ, σ). For more details, we refer to the full version.

5 Conclusion

We start with briefly sketching how our technique can be adapted such that it also applies to the vertex deletion (instead of edge deletion) versions of the considered problems. Afterwards, we discuss future challenges and open problems.

Extension to Vertex-Deletion Variants. Most of our results can be transferred to the vertex deletion variants of the considered edge deletion problems as follows.

To this end, we modify the T-fractal as displayed in Figure 4:

First, subdivide each edge. Then, replace each vertex v in the original T-fractal by many pairwise non-adjacent vertices with the same neighborhood as v. The number of these introduced "false twins" is larger than the given budget such that the only way to disconnect vertices in the new fractal will be to delete vertices introduced from the subdivision of the edges. In this way, deleting a vertex in the new T-fractal corresponds to deleting an edge in the original fractal. This new fractal might not be planar anymore, but, as in the edge deletion variant, one can direct the edges in such a way that the obtained directed graph is acyclic.

We claim that the new T-fractal can be used in the same way as the original T-fractal in order to exclude polynomial kernels for vertex deletion variants of the problems discussed in this work – both in undirected and directed, but not for planar variants.

Outlook. We provided several case studies where our fractalism technique applies. It remains open to further explore the limitations and possibilities of our technique in more contexts. Table 1 in Section 1 presents an open question which should be clarified. Moreover, we could not settle the cases for vertex deletion problems when the underlying graphs are planar.

Acknowledgement. We thank Manuel Sorge (TU Berlin) for fruitful discussions.

References

1 Georg Baier, Thomas Erlebach, Alexander Hall, Ekkehard Köhler, Petr Kolman, Ondrej Pangrác, Heiko Schilling, and Martin Skutella. Length-bounded cuts and flows. *ACM Transactions on Algorithms*, 7(1):4, 2010. doi:10.1145/1868237.1868241.

2 Cristina Bazgan, Morgan Chopin, Marek Cygan, Michael R. Fellows, Fedor V. Fomin, and Erik Jan van Leeuwen. Parameterized complexity of firefighting. *Journal of Computer and System Sciences*, 80(7):1285–1297, 2014.

3 Cristina Bazgan, André Nichterlein, and Rolf Niedermeier. A refined complexity analysis of finding the most vital edges for undirected shortest paths. In *Proc. of the 9th International Conference on Algorithms and Complexity (CIAC 2015)*, volume 9079 of *LNCS*, pages 47–60. Springer, 2015. doi:10.1007/978-3-319-18173-8_3.

4 René van Bevern, Robert Bredereck, Morgan Chopin, Sepp Hartung, Falk Hüffner, André Nichterlein, and Ondřej Suchý. Parameterized complexity of dag partitioning. In *Proc. of the 8th International Conference on Algorithms and Complexity (CIAC'13)*, volume 7878 of *LNCS*, pages 49–60. Springer, 2013.

5 Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computer and System Sciences*, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001.

6 Hans L. Bodlaender, Bart M. P. Jansen, and Stefan Kratsch. Kernelization lower bounds by cross-composition. *SIAM Journal on Discrete Mathematics*, 28(1):277–305, 2014. doi:10.1137/120880240.

7 Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011.

8 Liming Cai, Jianer Chen, Rodney G. Downey, and Michael R. Fellows. Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, 84(1):119–138, 1997.

9 Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 4th edition, 2010.

10 Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Kernelization lower bounds through colors and ids. *ACM Transactions on Algorithms*, 11(2):13, 2014. doi:10.1145/2650261.

11 Pavel Dvořák and Dušan Knop. Parametrized complexity of length-bounded cuts and multi-cuts. In *Proc. of the 12th Annual Conference on Theory and Applications of Models of Computation (TAMC 2015)*, volume 9076 of *LNCS*, pages 441–452. Springer, 2015. doi:10.1007/978-3-319-17142-5_37.

12 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer and System Sciences*, 77(1):91–106, 2011. doi:10.1016/j.jcss.2010.06.007.

13 Petr A. Golovach and Dimitrios M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization*, 8(1):72–86, 2011. doi:10.1016/j.disopt.2010.09.009.

14 Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007.

15 Venkatesan Guruswami and Euiwoong Lee. Inapproximability of feedback vertex set for bounded length cycles. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:6, 2014.

16 Alon Itai, Yehoshua Perl, and Yossi Shiloach. The complexity of finding maximum disjoint paths with length constraints. *Networks*, 12(3):277–286, 1982. doi:10.1002/net.3230120306.

17 Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113:58–97, 2014.

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

19 Kavindra Malik, Ashok K. Mittal, and Santosh K. Gupta. The k most vital arcs in the shortest path problem. *Operations Research Letters*, 8(4):223–227, 1989.

20 Dániel Marx. Parameterized complexity and approximation algorithms. *The Computer Journal*, 51(1):60–78, 2008.

21 Karl Menger. Über reguläre Baumkurven. *Mathematische Annalen*, 96(1):572–582, 1927.

22 Feng Pan and Aaron Schild. Interdiction problems on planar graphs. *Discrete Applied Mathematics*, 198:215–231, 2016.

23 Anneke A. Schoone, Hans L. Bodlaender, and Jan van Leeuwen. Diameter increase caused by edge deletion. *Journal of Graph Theory*, 11(3):409–427, 1987. doi:10.1002/jgt.3190110315.

24 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

25 Ge Xia and Yong Zhang. On the small cycle transversal of planar graphs. *Theoretical Computer Science*, 412(29):3501–3509, 2011.

Kernelization of Cycle Packing with Relaxed Disjointness Constraints

Akanksha Agrawal[1], Daniel Lokshtanov[2], Diptapriyo Majumdar[3], Amer E. Mouawad[4], and Saket Saurabh[5]

1 **University of Bergen, Bergen, Norway**
akanksha.agrawal@uib.no
2 **University of Bergen, Bergen, Norway**
daniel.lokshtanov@uib.no
3 **Institute of Mathematical Sciences, Chennai, India**
diptapriyom@imsc.res.in
4 **University of Bergen, Bergen, Norway**
a.mouawad@uib.no
5 **University of Bergen, Bergen, Norway, and**
Institute of Mathematical Sciences, Chennai, India
saket@imsc.res.in

Abstract

A key result in the field of kernelization, a subfield of parameterized complexity, states that the classic Disjoint Cycle Packing problem, i.e. finding k vertex disjoint cycles in a given graph G, admits no polynomial kernel unless NP $\subseteq$ coNP/poly. However, very little is known about this problem beyond the aforementioned kernelization lower bound (within the parameterized complexity framework). In the hope of clarifying the picture and better understanding the types of "constraints" that separate "kernelizable" from "non-kernelizable" variants of Disjoint Cycle Packing, we investigate two relaxations of the problem. The first variant, which we call Almost Disjoint Cycle Packing, introduces a "global" relaxation parameter t. That is, given a graph G and integers k and t, the goal is to find at least k distinct cycles such that every vertex of G appears in at most t of the cycles. The second variant, Pairwise Disjoint Cycle Packing, introduces a "local" relaxation parameter and we seek at least k distinct cycles such that every two cycles intersect in at most t vertices. While the Pairwise Disjoint Cycle Packing problem admits a polynomial kernel for all $t \geq 1$, the kernelization complexity of Almost Disjoint Cycle Packing reveals an interesting spectrum of upper and lower bounds. In particular, for $t = \frac{k}{c}$, where c could be a function of k, we obtain a kernel of size $\mathcal{O}(2^{c^2} k^{7+c} \log^3 k)$ whenever $c \in o(\sqrt{k})$. Thus the kernel size varies from being sub-exponential when $c \in o(\sqrt{k})$, to quasi-polynomial when $c \in o(\log^{\ell} k)$, $\ell \in \mathbb{R}_+$, and polynomial when $c \in \mathcal{O}(1)$. We complement these results for Almost Disjoint Cycle Packing by showing that the problem does not admit a polynomial kernel whenever $t \in \mathcal{O}(k^{\epsilon})$, for any $0 \leq \epsilon < 1$.

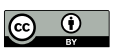
1998 ACM Subject Classification G.2.2 Graph Algorithms, I.1.2 Analysis of Algorithms

Keywords and phrases parameterized complexity, cycle packing, kernelization, relaxation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.26

1 Introduction

Polynomial-time preprocessing is one of the widely used methods to tackle NP-hard problems in practice, as it plays well with exact algorithms, heuristics, and approximation algorithms. Until recently, there was no robust mathematical framework to analyze the performance of



preprocessing routines. Progress in parameterized complexity [11] made such an analysis possible. In parameterized complexity, each problem instance is coupled with a parameter k and the parameterized problem is said to admit a *kernel* if there is a polynomial-time algorithm, called a *kernelization* algorithm, that reduces the input instance down to an instance whose size is bounded by a function $f(k)$ in k, while preserving the answer. Such an algorithm is called an $f(k)$-*kernel* for the problem. If $f(k)$ is a polynomial, quasi-polynomial, subexponential, or exponential function of k, we say that this is a polynomial, quasi-polynomial, subexponential, or exponential kernel, respectively. Over the last decade or so, kernelization has become a very active field of study, especially with the development of complexity-theoretic tools to show that a problem does not admit a polynomial kernel [3, 12, 16, 18], or a kernel of a specific size [8, 9, 19]. We refer the reader to the survey articles by Kratsch [20] and Lokshtanov et al. [21] for recent developments.

One of the first and important problems to which the lower-bounds machinery was applied is the NP-complete Disjoint Cycle Packing problem. In the Disjoint Cycle Packing problem, we are given as input an n-vertex graph G and an integer k, and the task is to find a collection $\mathcal{C}$ of at least k pairwise disjoint vertex sets of G, such that every set $C \in \mathcal{C}$ induces a cycle in G. The Disjoint Cycle Packing problem can be solved in $\mathcal{O}(k^{k \log k} n^{\mathcal{O}(1)})$ using dynamic programming over graphs of bounded treewidth [2, 4]. Bodlaender et al. [5] showed that, when parameterized by k, Disjoint Cycle Packing does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly (and the polynomial hierarchy collapses to its third level, which is considered very unlikely). Beyond the aforementioned negative result for polynomial kernels and the folklore $\mathcal{O}(k^{k \log k} n^{\mathcal{O}(1)})$-time algorithm, the Disjoint Cycle Packing problem has remained mostly unexplored from the viewpoint of parameterized complexity.

Our problems and results. In this paper we study two variants of Disjoint Cycle Packing, obtained by relaxing the disjointness constraint. In particular, we focus on the kernelization complexity of the Disjoint Cycle Packing problem by considering two relaxed versions of the problem, one with a "local" relaxation parameter and the other with a "global" relaxation parameter. In the locally relaxed variant, which we call Pairwise Disjoint Cycle Packing, the goal is to find at least k distinct cycles in a graph G such that they pairwise intersect in at most t vertices.

Pairwise Disjoint Cycle Packing	**Parameter:** k
Input: An undirected (multi) graph G and integers k and t	
Question: Does G have at least k distinct cycles $C_1, \ldots, C_k$ such that $\lvert V(C_i) \cap V(C_j) \rvert \leq t$ for all $i \neq j$?	

We consider two cycles to be distinct whenever their edge sets differ by at least one element. Note that when $t = 0$, Pairwise Disjoint Cycle Packing corresponds to the original Disjoint Cycle Packing problem. However, when $t = |V(G)|$ the Pairwise Disjoint Cycle Packing problem is solvable in time polynomial in $|V(G)|$ and k since we can enumerate distinct cycles in a graph with polynomial delay [24]. In other words, any k distinct cycles in a graph will trivially pairwise intersect in at most $|V(G)|$ vertices. We show that Pairwise Disjoint Cycle Packing remains NP-complete when $t = 1$. Then, we complement this result by showing that the problem admits a polynomial kernel for $t = 1$ and a polynomial compression for $t \geq 2$. An interesting problem which remains unclear is to determine what value of t separates NP-hard instances from polynomial-time solvable ones.

The second relaxation we consider is Almost Disjoint Cycle Packing. The goal in Almost Disjoint Cycle Packing is to determine whether G contains at least k distinct

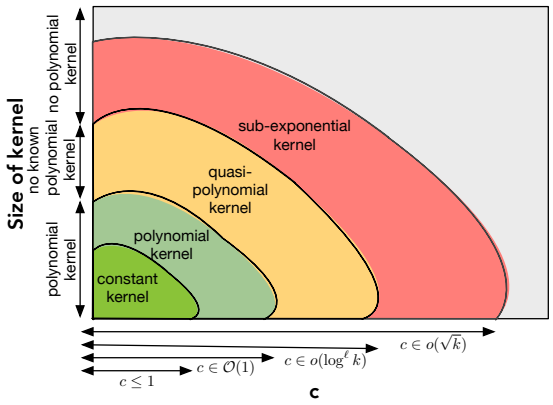


Figure 1 Spectrum of kernelization algorithms for Almost Disjoint Cycle Packing as c grows in the denominator of $t = \frac{k}{c}$.

cycles such that every vertex in $V(G)$ appears in at most t of them. As we shall see, the kernelization complexity landscape for Almost Disjoint Cycle Packing is much more diverse than that of Pairwise Disjoint Cycle Packing. In some sense, this suggests that the global relaxation parameter does a "better job" of capturing the "hardness" of the original problem.

Almost Disjoint Cycle Packing	**Parameter:** k
Input: An undirected (multi) graph G and integers k and t	
Question: Does G have at least k distinct cycles $C_1, \ldots, C_k$ such that every vertex in $V(G)$ appears in at most t of them?	

Again, for $t = 1$, Almost Disjoint Cycle Packing corresponds to Disjoint Cycle Packing and when $t = k$ the problem is solvable in time polynomial in $|V(G)|$ and k by simply enumerating distinct cycles. However, and rather surprisingly, we show that t has to be "very close" to k for this relaxation to become "easier" than the original problem, at least in terms of kernelization. In fact, we show that as long as $t = \mathcal{O}(k^{1-\epsilon})$, where $0 < \epsilon \leq 1$, Almost Disjoint Cycle Packing remains NP-complete and admits no polynomial kernel unless NP $\subseteq$ coNP/poly. We complement our hardness result by a spectrum of kernel upper bounds. To that end, we consider the case $t = \frac{k}{c}$, where c is a constant or a function of k. We show that we can (in polynomial time) compress an instance of Almost Disjoint Cycle Packing into an equivalent instance with $\mathcal{O}(2^{c^2} k^{7+c} \log^3 k)$ vertices. This implies polynomial, quasi-polynomial, or subexponential size kernels for Almost Disjoint Cycle Packing, depending on whether c is a constant, $c \in o(\log k)$, or $c \in o(\sqrt{k})$, respectively. It remains open whether the problem is in P or NP-hard for $t = \frac{k}{c}$, when c is a constant. A high level summary of our results for Almost Disjoint Cycle Packing is given in Figure 1. Most of the technical details and proofs have been omitted from this extended abstract.

Related Results. Our results also fit into the relatively new direction of research that is concerned with the parameterized complexity of problems with relaxed packing/covering constraints. For several important problems (that we need to solve), there are settings in which we need not be very strict about constraints. This is particularly interesting for "strict" problems where, e.g., (a) it is known that no polynomial kernels are possible unless NP $\subseteq$ coNP/poly, or where (b) the algorithm with the best running time matches

the known lower bound, or where (c) no considerable improvements have been made either algorithmically or in terms of kernel upper/lower bounds. The Disjoint Cycle Packing problem falls into categories (a) and (c) and is the main subject of this work. Before we state our results, let us look at some examples where the introduction of relaxation parameters has been successful. Abasi et al. [1], followed by Gabizon et al. [17], studied a generalization of the k-Path problem, namely r-Simple k-Path, where the task is to find a walk of length k that never visits any vertex more than r times. Here r is the relaxation parameter. By definition, the generalized problem is computationally harder than the original. However, observe that for $r = 1$ the problem is exactly the problem of finding a simple path of length k in G. On the other hand, for $r = k$ the problem is easily solvable in polynomial time, as any walk in G of length k will suffice. In some sense, the "further away" an instance of the generalized problem is from being an instance of the original, the easier the instance is. Put differently, gradually increasing r from 1 to k should make the problem computationally easier. This intuition was confirmed by the authors by providing, amongst other results, algorithms for the generalized problem whose worst-case running time matches the running time of the best algorithm for the original problem up to constants in the exponent, and improves significantly as the relaxation parameter increases. Also closely related is the work of Romero et. al. [26, 27] and Fernau et al. [14] who studied relaxations of graph packing problems allowing certain overlaps.

2 Preliminaries

We let $\mathbb{N}$ denote the set of natural numbers, $\mathbb{R}$ denote the set of real numbers, $\mathbb{R}_+$ denote the set of non-zero positive real numbers, and $\mathbb{R}_{\geq 1}$ denote the set of real numbers greater than or equal to one. For $r \in \mathbb{N}$, by $[r]$ we denote the set $\{1, 2, \ldots, r\}$.

Graphs. We use standard terminology from the book of Diestel [10] for those graph-related terms which are not explicitly defined here. We only consider finite graphs possibly having loops and multi-edges. For a graph G, $V(G)$ and $E(G)$ denote the vertex and edge sets of the graph G, respectively. For a vertex $v \in V(G)$, we use $d_G(v)$ to denote the degree of v, i.e the number of edges incident on v, in the (multi) graph G. We also use the convention that a loop at a vertex v contributes two to its degree. For a vertex subset $S \subseteq V(G)$, $G[S]$ and $G - S$ are the graphs induced on S and $V(G) \setminus S$, respectively. For a vertex subset $S \subseteq V(G)$, we let $N_G(S)$ and $N_G[S]$ denote the open and closed neighborhood of S in G. That is, $N_G(S) = \{v \mid (u, v) \in E(G), u \in S\} \setminus S$ and $N_G[S] = N_G(S) \cup S$. For a graph G and an edge $e \in E(G)$, G/e denotes the graph obtained by contracting e in G.

A *path* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that (v_i, v_{i+1}) is an edge for all $0 \leq i < \ell$. A *cycle* in a graph is a sequence of distinct vertices $v_0, v_1, \ldots, v_\ell$ such that $(v_i, v_{(i+1) \mod \ell})$ is an edge for all $0 \leq i \leq \ell$. We note that both a double edge and a loop are cycles. If P is a path from a vertex u to a vertex v in graph G then we say that u and v are the end vertices of the path P and P is a (u, v)-path. For a path P, we use $V(P)$ to denote the set of vertices in the path P and the length of P is denoted by $|P|$ (i.e, $|P| = |V(P)|$). For a cycle C, we use $V(C)$ to denote the set of vertices in the cycle C and length of C, denoted by $|C|$, is $|V(C)|$. For a path or a cycle Q we use $N_G(Q)$ and $N_G[Q]$ to denote the set $N_G(V(Q))$ and $N_G[V(Q)]$, respectively. For a collection of paths/cycles $\mathcal{Q}$, we use $|\mathcal{Q}|$ to denote the number of paths/cycles in $\mathcal{Q}$ and $V(\mathcal{Q})$ to denote the set $\bigcup_{Q \in \mathcal{Q}} V(Q)$. We sometimes refer to a path or a cycle Q as a $|Q|$-path or $|Q|$-cycle. Given a vertex $v \in V(G)$, a *v-flower* of order k is a set of k cycles in G whose pairwise

intersection is exactly $\{v\}$. We say a set of distinct vertices $P = \{v_1, \ldots, v_\ell\}$ in G forms a *degree-two path* if P is a path and all vertices $\{v_1, \ldots, v_\ell\}$ have degree exactly two in G. We say P is a *maximal degree-two path* if no proper superset of P also forms a degree-two path. Finally, a *feedback vertex set* is a subset S of vertices such that $G - S$ is a forest.

▶ **Theorem 1** ([13]). *There exists a constant c such that every (multi) graph either contains k vertex disjoint cycles or it has a feedback vertex set of size at most $ck \log k$. Moreover, there is a polynomial-time algorithm that takes a graph G and an integer k as input, and outputs either k vertex disjoint cycles or a feedback vertex set of size at most $ck \log k$.*

Parameterized Complexity. We only state the basic definitions and general results needed for our purposes. For more details on parameterized complexity in general, and kernelization in particular, we refer the reader to the books of Downey and Fellows [11], Flum and Grohe [15], Niedermeier [23], and the more recent book by Cygan et al. [7].

▶ **Definition 2.** A *polynomial compression* of a parameterized language $L \subseteq \Sigma \times \mathbb{N}$ into a language $R \subseteq \Sigma^*$ is an algorithm that takes as input an instance $(I, k) \in \Sigma \times \mathbb{N}$, works in time polynomial in $|I| + k$, and returns a string I' such that:

- $|I'| \leq p(k)$ for some polynomial $p(.)$, and
- $|I'| \in R$ if and only if $(I, k) \in L$.

In case $|\Sigma| = 2$, the polynomial $p(.)$ is called the *bitsize* of the compression.

Note that polynomial compressions are a generalization of kernels and being able to rule out a compression algorithm automatically rules out a kernelization algorithm.

▶ **Definition 3.** Let $L, R \subseteq \Sigma \times \mathbb{N}$ be two parameterized problems. An algorithm $\mathcal{A}$ is called a *polynomial parameter transformation* from L to R if, given an instance (I, k) of problem :, $\mathcal{A}$ works in polynomial time and outputs an equivalent instance (I', k') of problem R, i.e., $(I, k) \in L$ if and only if $(I', k') \in R$, such that $k' \leq p(k)$ for some polynomial $p(.)$.

▶ **Theorem 4** ([7]). *Let $L, R \subseteq \Sigma \times \mathbb{N}$ be two parameterized problems and assume there exists a polynomial parameter transformation from L to R. Then, if R does not admit a polynomial compression, neither does L. In particular, if R does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly then the same holds for L.*

3 Almost Disjoint Cycle Packing

As previously noted, Bodlaender et al. [5] showed that Disjoint Cycle Packing admits no polynomial kernel unless NP $\subseteq$ coNP/poly. On the other hand, finding k distinct cycles in a graph is solvable in time polynomial in n and k [24]. The intuition is that the more cycles we allow a vertex to belong to, the easier the problem of finding k distinct cycles should become. In this section, we study the spectrum of kernelization algorithms for Almost Disjoint Cycle Packing based on the "distance" between k and t. Recall that given an instance (G, k, t) of Almost Disjoint Cycle Packing, our goal is to find at least k distinct cycles such that each vertex appears in at most t of them. To formalize the notion of distance between k and t, we define the following class of problems.

Let $L = \{(G, k, t) \mid$ G has k cycles such that every vertex appears in at most t of them$\}$. Basically, L is the language Almost Disjoint Cycle Packing. For a monotonically increasing computable function $f : \mathbb{N} \to \mathbb{R}_+$, we define the following sub-language of L.

$$L_f = \{(G, k, t) \mid (G, k, t) \in L \text{ and } t = \lceil k/f(k) \rceil\}.$$

When f is the identity function, i.e. when $f(k) = k$, L_f is exactly the Disjoint Cycle Packing problem which is known not to admit a polynomial kernel [5]. In Section 3.1, we show that even when $f(k) = k^\epsilon$, for any fixed $0 < \epsilon \leq 1$, L_f (or equivalently Almost Disjoint Cycle Packing with $t = k^{1-\epsilon}$) is NP-complete and does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly. If $f = a$ (a constant function), where $a \leq 1$ and $a \in \mathbb{R}_+$, then L_f can be decided in polynomial time (as finding any k distinct cycles is enough). This implies that for $f = a$ we have a constant kernel. In Section 3.2, we obtain a polynomial kernel for $f = c$ (another constant function), where $c > 1$ and $c \in \mathbb{R}$. In fact, our result implies that for $f \in \mathcal{O}(1)$, $f \in o(\log^\ell k)$ ($\ell \in \mathbb{N}$), or $f \in o(\sqrt{k})$, we can (in polynomial time) compress an instance of Almost Disjoint Cycle Packing into an equivalent instance of polynomial, quasi-polynomial, or subexponential size, respectively (see Figure 1).

Before we consider the kernelization complexity of the Almost Disjoint Cycle Packing problem, we first show, using standard arguments, that the problem is fixed-parameter tractable when parameterized by k. Armed with Theorem 1, we can assume that, for an instance (G, k, t) of Almost Disjoint Cycle Packing, the treewidth of G is at most $\mathcal{O}(k \log k)$; as G has a feedback vertex set of size at most $\mathcal{O}(k \log k)$. Courcelle's Theorem [6] gives a powerful way of quickly showing that a problem is fixed-parameter tractable on bounded treewidth graphs. That is, it suffices to show that our problem can be expressed in monadic second-order logic (MSO_2).

▶ **Theorem 5** ([6]). *If a graph property can be described as a formula ϕ in the monadic second-order logic of graphs, then it can be recognized in time $f(||\phi||, tw(G))(|E(G)| + |V(G)|)$ if a given graph G has this property, where f is a computable function, $||\phi||$ is the length of the encoding of ϕ as a string, and $tw(G)$ is the treewidth of G.*

▶ **Lemma 6.** Almost Disjoint Cycle Packing *can be solved in $f(k)n^{\mathcal{O}(1)}$ time, for some computable function f. In other words, the problem is fixed-parameter tractable when parameterized by k.*

3.1 Refuting polynomial kernels for $t = \mathcal{O}(k^{1-\epsilon})$

We now show that Almost Disjoint Cycle Packing restricted to L_f, where $f(k) = k^\epsilon$, does not admit a polynomial kernel, for any $0 < \epsilon \leq 1$, unless NP $\subseteq$ coNP/poly. Here k is the number of required cycles and $t = \frac{k}{f(k)} = k^{1-\epsilon}$ is the maximum number of cycles a vertex can belong to. Below we define the Disjoint Factors problem [5] which is known to admit no polynomial compression unless NP $\subseteq$ coNP/poly.

Let Σ_q be an alphabet set of q elements. By $\Sigma_q^\star$ we denote the set of all strings over Σ_q. A factor of a string $\bar{y} = y_1 y_2 \dots y_n \in \Sigma_q^\star$ is a pair (s, e), where $s, e \in [n]$ and $s < e$, such that $y_s y_{s+1} \dots y_e$ is a substring of $\bar{y}$ and $y_s = y_e$. Two factors (s, e) and (s', e') of $\bar{y}$ are said to be disjoint if $\{s, s+1, \dots, e\} \cap \{s', s'+1, \dots, e'\} = \emptyset$. The string $\bar{y}$ is said to have a disjoint factor over Σ_q if for all $x \in \Sigma_q$ there is a factor (s_x, e_x) such that $y_{s_x} = y_{e_x} = x$, and for all $x, \hat{x} \in \Sigma_q$, (s_x, e_x) and $(s_{\hat{x}}, e_{\hat{x}})$ are disjoint factors.

Disjoint Factors	**Parameter:** q
Input: Alphabet set Σ_q, string $\bar{y} \in \Sigma_q^\star$	
Question: Does $\bar{y}$ have a disjoint factor?	

Construction. We give a polynomial parameter transformation from an instance $(\Sigma_q, \bar{y})$ of Disjoint Factors to an instance (G, k, t) of Almost Disjoint Cycle Packing. For

technical reasons, we will assume that $t-1 = 2^l$, for some $l \in \mathbb{N}$. Note that this can be achieved by at most doubling the value of t while keeping t in $\mathcal{O}(k^{1-\epsilon})$. We let $l = \log_2(t-1)$. The end goal will be to construct a graph in which we have to find k cycles such that every vertex appears in at most $t = \mathcal{O}(k^{1-\epsilon})$ of them.

The reduction is as follows. Let $\Sigma_q = \{x_1, x_2, \dots x_q\}$. We create a vertex $\hat{x}_i \in V(G)$ corresponding to each element x_i, where $i \in [q]$. For $\bar{y} = y_1 y_2 \dots y_n \in \Sigma_q^\star$ we create a path $P_y = (u, \hat{y}_1, \hat{y}_2, \dots \hat{y}_n, u')$. We add an edge between $\hat{x}_i$ and $\hat{y}_j$, for $i \in [q]$ and $j \in [n]$, if and only if $x_i = y_j$. We also add four more vertices u_1, u_2, u_1', and u_2' to $V(G)$ and add edges (u_1, u_2), (u_2, u), (u, u_1), (u_1', u_2'), (u_2', u'), and (u', u_1') to $E(G)$. For each $x_i \in \Sigma$, we attach $t-1$ triangles to $\hat{x}_i$, i.e. we add edges $\{(z_i^1, \tilde{z}_i^1), (z_i^2, \tilde{z}_i^2), \dots, (z_i^{t-1}, \tilde{z}_i^{t-1})\}$ and $(z_i^j, \hat{x}_i), (\hat{x}_i, \tilde{z}_i^j)$, for $j \in [t-1]$. Next, we create a path $P_w = (w_1, w_1', w_2, w_2', \dots, w_l, w_l')$ in G. We add a set $R = \{r_i \mid i \in [l]\}$ of l of independent vertices and for $i \in [l]$, we add the edges (w_i, r_i) and (w_i', r_i) to $E(G)$. Finally, we add edges (u, w_1) and (w_l', u'). We set $k = tq + t + l + 1$, which completes the construction.

▶ **Proposition 7.** *Let $P = (s, a_1, a_1', a_2, a_2', \dots, a_n, a_n', s')$ be a path and $B = \{b_i \mid i \in [n]\}$ be a set of independent vertices. Let H be the graph consisting of path P, the set B, and, for $i \in [n]$, the edges (a_i, b_i) and (a_i', b_i). Then, for each $B' \subseteq B$, there is a path $P_{B'}$ such that $V(P_{B'}) \cap B = B'$. Moreover, the set $\mathcal{B} = \{P_{B'} \mid B' \subseteq B\}$ is the set of all possible paths between s, s' in H.*

Applying Proposition 7 to G, for each $R' \subseteq R$, we have a (unique) cycle $C_{R'}$ which contains all the vertices in $V(P_y)$, all the vertices in P_w, and exactly R' vertices from R. We define a family of cycles $\mathcal{R} = \{C_{R'} \mid R' \subseteq R\} \cup \{(w_i, w_i', r_i) \mid i \in [l]\}$. Note that $|\mathcal{R}| = 2^l + l = t + l - 1$ and each $C \in \mathcal{R}$ is a cycle in G. The intuition of having the set of cycles $\{C_{R'} \mid R' \subseteq R\}$ in G is that each vertex in path P_y must be used $t-1$ times and can therefore participate in one additional cycle (which contains vertices in $V(P_y)$). We associate each such extra cycle with a factor.

▶ **Theorem 8.** *Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable monotonically increasing function such that $f(k) \in \mathcal{O}(k^\epsilon)$, where $0 < \epsilon \leq 1$. Then,* Almost Disjoint Cycle Packing *admits no polynomial kernel over L_f unless NP $\subseteq$ coNP/poly.*

3.2 A kernel for Almost Disjoint Cycle Packing

Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable monotonically increasing function such that $f(k) \in o(\sqrt{k})$. In this section, we consider the Almost Disjoint Cycle Packing problem restricted to L_f. The kernelization algorithm presented below is inspired by the *lossy kernel* for the Cycle Packing problem given in [22]. To simplify notation, we let $c = f(k)$ and use c instead of $f(k)$ throughout the section, which implies that $t = \lceil \frac{k}{c} \rceil$. As we shall see, the assumption $c \in o(\sqrt{k})$ is required to guarantee that our kernelization algorithm does in fact run in time polynomial in the input size. We show that, as long as $c \in o(\sqrt{k})$, we can in polynomial time reduce an instance to at most $\mathcal{O}(2^{\lceil c \rceil^2} k^{7+\lceil c \rceil} \log^3 k)$ vertices. Our kernelization algorithm can be more or less divided into three stages. We start by computing (using Theorem 1) a feedback vertex set of size at most $\mathcal{O}(k \log k)$ and denote this set by F (assuming no k vertex disjoint cycles were found). We let $T = G - F$ and let $T_{\leq 1}$, T_2, and $T_{\geq 3}$, denote the sets of vertices in T having degree at most one in T, degree exactly two in T, and degree greater than two in T, respectively. Moreover, we let $\mathcal{P}$ denote the set of all maximal degree-two paths in $G[T_2]$. Next, we bound the size of $T_{\leq 1}$, which implies a bound on the size of $T_{\geq 3}$ and $\mathcal{P}$. In the second stage, we show that (roughly speaking) the graph can have at most

$\lceil c \rceil - 1$ vertices of high degree. Using this fact, the last stage consists of bounding the size of T_2.

Bounding the size of $T_{\leq 1}$. First, we get rid of vertices of degree one and two using Reduction Rules A1 and A2.

▶ **Reduction Rule A1.** *Delete vertices of degree zero or one in G.*

▶ **Reduction Rule A2.** *If there is a vertex v of degree exactly two in G then delete v and connect its two neighbors by a new edge.*

▶ **Reduction Rule A3.** *If there exists an edge $(u, v) \in E(G)$ of multiplicity more than $2t$ then reduce its multiplicity to $2t \leq 2k$.*

The fact that we can assume $2t \leq 2k$ follows from the observation that when $t = k$ the problem becomes solvable in time polynomial in n and k. Once Reduction Rules A1, A2, and A3 are no longer applicable, the minimum degree of the graph is three and the multiplicity of every edge is at most $2t$. Note that every vertex in $T_{\leq 1}$ is either a leaf or an isolated vertex in T. Therefore, every vertex of $T_{\leq 1}$ has at least two neighbours in F. For $(u, v) \in F \times F$, let $L(u, v)$ be the set of vertices of degree at most one in $T = G - F$ such that each $x \in L(u, v)$ is adjacent to both u and v (if $u = v$, then $L(u, u)$ is the set of vertices which have degree at most one in $T = G - F$ and at least two edges to u). For each pair $(u, v) \in F \times F$, we mark $|F|\lceil \frac{k}{c} \rceil + 2k + 1$ vertices from $L(u, v)$ if $L(u, v) > |F|\lceil \frac{k}{c} \rceil + 2k + 1$ and mark all vertices in $L(u, v)$ if $L(u, v) \leq |F|\lceil \frac{k}{c} \rceil + 2k + 1$.

▶ **Reduction Rule A4 [22].** *If $|T_{\leq 1}| \geq |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k + 1) + 1$ then there exists an unmarked vertex $v \in T_{\leq 1}$.*

- *If $d_{G-F}(v) = 0$ then delete v.*
- *If $d_{G-F}(v) = 1$ contract the unique edge in $G - F$ which is incident to v. We let e denote this unique edge and we let w denote the other endpoint onto which we contract e.*

Bounding the number of high-degree vertices. When none of the aforementioned reduction rules are applicable, the size of $T_{\leq 1}$, $T_{\geq 3}$, and $\mathcal{P}$, is at most $|F|^2(|F|\lceil \frac{k}{c} \rceil + 2k + 1) = \mathcal{O}(k^4 \log^3 k)$. Consider $\mathcal{P}$, i.e. the collection of maximal degree-two paths in T_2, and assume that there exists a set $F_{\lceil c \rceil} = \{x_1, \ldots, x_{\lceil c \rceil}\} \subseteq F$ (of size $\lceil c \rceil$) such that for every vertex $x \in F_{\lceil c \rceil}$ there exists a path $P \in \mathcal{P}$ such that x has at least $4k\lceil c \rceil$ neighbours in P. We show that if $F_{\lceil c \rceil}$ exists then we have a yes-instance.

▶ **Reduction Rule A5.** *If there exists a set of $\lceil c \rceil$ vertices $F_{\lceil c \rceil} = \{x_1, \ldots, x_{\lceil c \rceil}\} \subseteq F$ such that for all x_i, $1 \leq i \leq \lceil c \rceil$, $|N_G(x_i) \cap V(\mathcal{P})| > |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k + 1)4k\lceil c \rceil$, then return a trivial yes-instance.*

After applying Reduction Rule A5, there can be at most $\lceil c \rceil - 1$ vertices in F having more than $|F|^2(|F|\lceil \frac{k}{c} \rceil + 2k + 1)4k\lceil c \rceil = \mathcal{O}(k^5 \log^3 k)$ neighbors in T_2. We let $F_{\lceil c \rceil - 1} \subseteq F$ denote the maximum sized such subset and we let $F^\star = F \setminus F_{\lceil c \rceil - 1}$. For any vertex $x \in F^\star$, $|N_G(x) \cap V(\mathcal{P})| \leq |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k + 1)4k\lceil c \rceil$ and, consequently, $|N_G(F^\star) \cap V(\mathcal{P})| \leq |F|^2(|F|\lceil \frac{k}{c} \rceil + 2k + 1)4k\lceil c \rceil |F^\star| \leq |F|^3(|F|\lceil \frac{k}{c} \rceil + 2k + 1)4k\lceil c \rceil = \mathcal{O}(k^6 \log^3 k)$.

Bounding the size of T_2. We start by marking all vertices in F, $T_{\leq 1}$, $T_{\geq 3}$, and $N_G(F^\star) \cap V(\mathcal{P})$. The total number of marked vertices is therefore in $\mathcal{O}(k^6 \log^3 k)$. Moreover, all the unmarked vertices must be in T_2 and form degree-two paths. Each unmarked vertex must

have at least one neighbor in $F_{\lceil c\rceil-1}$ and cannot have neighbors in $F^\star$. We call a set of unmarked vertices a *region* if they form a maximal path in $G[T_2]$. At this point, the total number of regions is in $\mathcal{O}(k^6 \log^3 k)$, as the number of marked vertices is in $\mathcal{O}(k^6 \log^3 k)$. Therefore, our last step is to bound the size of each region. To do so, we first recursively further subdivide each region as follows. Fix a region R and check for each vertex $x_i \in F_{\lceil c\rceil-1}$, the value of $|N_G(x_i) \cap R|$. If $|N_G(x_i) \cap R| < 4k\lceil c\rceil 2^{\lceil c\rceil}$, then we again mark the vertices in $N_G(x_i) \cap R$, increasing the number of regions by a multiplicative factor of at most $4k\lceil c\rceil 2^{\lceil c\rceil}$. We repeat this process as long as there exists a region R and a vertex $x_i \in F_{\lceil c\rceil-1}$ satisfying $|N_G(x_i) \cap R| < 4k\lceil c\rceil 2^{\lceil c\rceil}$. Since $|F_{\lceil c\rceil-1}| < \lceil c\rceil$, repeating this procedure for every region and every vertex in $F_{\lceil c\rceil-1}$ increases the number of regions to at most $\mathcal{O}(2^{\lceil c\rceil^2} k^{6+\lceil c\rceil} \log^3 k)$; each of the initial $\mathcal{O}(k^6 \log^3 k)$ regions can be subdivided into at most $(4k\lceil c\rceil 2^{\lceil c\rceil})^{\lceil c\rceil}$ subregions.

▶ **Lemma 9.** *Let H be a graph consisting of a path P and an independent set $X = \{x_1, \ldots, x_{\lceil c\rceil}\}$ of size $\lceil c\rceil \geq 1$. Let $k \geq \lceil c\rceil^2$ be an integer. If $\forall x \in X$ we have $|N_H(x)| \geq 4k\lceil c\rceil 2^{\lceil c\rceil}$ and $\forall p \in V(P)$ we have $|N_H(p) \cap X| > 0$, then we can construct a set of distinct cycles $\mathcal{C} = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_{\lceil c\rceil}$ such that (a) $|\mathcal{C}_i| = \lceil \frac{k}{c} \rceil$, (b) all cycles in $\mathcal{C}_i$ pairwise intersect in x_i, and (c) every vertex in P appears in at most one cycle in $\mathcal{C}$.*

Using Lemma 9, we can get an upper bound on the size of a region R by applying the following reduction rule. Recall that by construction (and after subdividing regions), vertices of a region have neighbours only in $F_{\lceil c\rceil-1}$, where $F_{\lceil c\rceil-1}$ is a set of at most $\lceil c\rceil - 1$ vertices. In fact, for each region R, there exists a set $F_R \subseteq F_{\lceil c\rceil-1}$ such that each vertex in R has at least one neighbor in F_R and each vertex in F_R has at least $4k\lceil c\rceil 2^{\lceil c\rceil}$ neighbors in R.

▶ **Reduction Rule A6.** *Let R be a region such that $|R| > 4k\lceil c\rceil 4^{\lceil c\rceil}$. Let $\mathcal{Q} = \{Q_1, Q_2, \ldots\}$ be a family of sets which partitions R such that for any two vertices $u, v \in R$, we have $u, v \in Q_i$ if and only if $N_G(u) \cap F_R = N_G(v) \cap F_R$. In other words, two vertices belong to the same set in $\mathcal{Q}$ if and only if they share the same neighborhood in F_R. Since $|R| > 4k\lceil c\rceil 4^{\lceil c\rceil}$ and $|\mathcal{Q}| \leq 2^{\lceil c\rceil}$, there exists a set $Q \in \mathcal{Q}$ such that $|Q| > 4k\lceil c\rceil 2^{\lceil c\rceil}$. Let v be a vertex in Q and let w be a neighbor of v in R (v can have at most two neighbors in R). Contract the edge (v, w) onto w. Note that since $|Q| > 4k\lceil c\rceil 2^{\lceil c\rceil}$, each vertex in F_R has at least $4k\lceil c\rceil 2^{\lceil c\rceil}$ neighbors in R even after the contraction.*

Since the number of regions is in $\mathcal{O}(2^{\lceil c\rceil^2} k^{6+\lceil c\rceil} \log^3 k)$ and the size of a region is at most $4kc4^c$, the theorem follows.

▶ **Theorem 10.** *Let $f : \mathbb{N} \to \mathbb{R}_{\geq 1}$ be a computable monotonically increasing function such that $f(k) \in o(\sqrt{k})$. For $c = f(k)$,* Almost Disjoint Cycle Packing *admits a kernel consisting of at most $\mathcal{O}(2^{c^2} k^{7+c} \log^3 k)$ vertices over L_f.*

Theorem 10 implies that when $c \in o(\sqrt{k})$ the Almost Disjoint Cycle Packing problem admits a subexponential kernel. When $c \in o(\log^\ell k)$, $\ell \in \mathbb{N}$, the problem admits a quasi-polynomial kernel. Finally, when $c \in \mathcal{O}(1)$ the problem admits a polynomial kernel.

4 Pairwise Disjoint Cycle Packing

Recall that in the Pairwise Disjoint Cycle Packing problem, given a graph G and integers k and t, the goal is to find at least k cycles such that every pair of cycles intersects in at most t vertices. To show NP-completeness of Pairwise Disjoint Cycle Packing, for $t = 1$, we give a reduction from a variant of SAT called 2/2/4-SAT defined as follows: Each clause contains four literals, each variable appears four times in the formula, twice

negated and twice not negated, and the question is whether there is a truth assignment of the variables such that in each clause there are exactly two true literals. This variant was shown NP-complete by Ratner and Warrnuth [25].

▶ **Theorem 11.** Pairwise Disjoint Cycle Packing *is NP-complete for* $t = 1$.

4.1 A polynomial kernel for $t = 1$

There are many similarities but also some subtle differences when dealing with the cases $t = 1$ and $t \geq 2$. For instance, for any value of $t \geq 1$, finding a flower of order k in the graph is sufficient to solve the problem. On the other hand, not all vertices of degree two can be bypassed when $t \geq 2$. More importantly, finding two vertices in G with more than $2k$ common neighbors is enough to solve the problem for $t \geq 2$ but not for $t = 1$. As we shall see, this seemingly small difference requires major changes when dealing with the case $t = 1$. We start with some classical results and reduction rules which will be used throughout. Whenever some reduction rule applies, we apply the lowest-numbered applicable rule.

The first step in our kernelization algorithm is to run the algorithm of Theorem 1 and either output a trivial yes-instance (if k vertex disjoint cycles are found) or mark the vertices of the feedback vertex set and denote this set by F. We proceed with the following simple reduction rules to handle low-degree vertices and self-loops in the graph.

▶ **Reduction Rule B1.** *Delete vertices of degree zero or one in* G.

▶ **Reduction Rule B2.** *If there is a vertex* v *of degree exactly two in* G *then delete* v *and connect its two neighbors by a new edge.*

▶ **Reduction Rule B3.** *If there exists a vertex* $v \in V(G)$ *with a self-loop then delete the loop (not the vertex) and decrease the parameter* k *by one.*

▶ **Reduction Rule B4.** *If there is a pair of vertices* u *and* v *in* $V(G)$ *such that there are more than two parallel edges between them then reduce the multiplicity of the edge to two.*

Once none of the above reduction rules are applicable, our next goal is to bound the maximum degree in the graph. To do so, we make use of the following.

▶ **Lemma 12** ([7]). *Given a (multi) graph* G*, an integer* k*, and a vertex* $v \in V(G)$*, there is a polynomial-time algorithm that either finds a* v*-flower of order* k *or finds a set* Z_v *such that* $Z_v \subseteq V(G) \setminus \{v\}$ *intersects all cycles passing through* v*,* $|Z_v| \leq 2k$*, and there are at most* $2k$ *edges incident to* v *and with second endpoint in* Z_v.

A *q-star*, $q \geq 1$, is a graph with $q + 1$ vertices, one vertex of degree q and all other vertices of degree 1. Let G be a bipartite graph with vertex bipartition (A, B). A set of edges $M \subseteq E(G)$ is called a *q-expansion* of A into B if (i) every vertex of A is incident with exactly q edges of M and (ii) M saturates exactly $q|A|$ vertices in B, i.e. there is a set of $q|A|$ vertices in B which are incident to edges in M.

▶ **Lemma 13** (See [7, 28]). *Let* q *be a positive integer and* G *be a bipartite graph with vertex bipartition* (A, B) *such that* $|B| \geq q|A|$ *and there are no isolated vertices in* B*. Then, there exist nonempty vertex sets* $X \subseteq A$ *and* $Y \subseteq B$ *such that:*

- X *has a* q*-expansion into* Y *and*
- *no vertex in* Y *has a neighbour outside* X*, i.e.* $N(Y) \subseteq X$.

Furthermore, the sets X *and* Y *can be found in time polynomial in the size of* G.

For every vertex $v \in V(G)$ of high degree (which will be specified later), we apply the algorithm of Lemma 12. If the algorithm finds a v-flower of order k, the following reduction rule allows us to deal with it.

▶ **Reduction Rule B5.** *If G has a vertex v such that there is a v-flower of order at least k then return a trivial yes-instance.*

Hence, in what follows we assume that no such flower was found but instead we have a set Z_v of size at most $2k$ such that $Z_v \subseteq V(G)$ intersects all cycles passing through v. Consider the connected components of the graph $G[V(G) \setminus (Z_v \cup \{v\})]$. At most $k-1$ of those components can contain a cycle, as otherwise we again have a trivial yes-instance consisting of k vertex disjoint cycles.

▶ **Reduction Rule B6.** *If there are k or more components in $G \setminus (\{v\} \cup Z_v)$ containing a cycle then return a trivial yes-instance.*

Moreover, for every component D of $G[V(G) \setminus (Z_v \cup \{v\})]$, we have $|N_G(v) \cap V(D)| \leq 1$. In other words, v has at most one neighbor in any component and out of those components at most $k-1$ are not trees. Let $\mathcal{D} = \{D_1, D_2, \ldots, D_q\}$ denote those trees in which v has a neighbor. Since the minimum degree of the graph is three, every leaf of a tree in $\mathcal{D}$ must have at least one neighbor in Z_v.

▶ **Lemma 14.** *Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a solution in G and let C be a cycle in $\mathcal{C}$ such that $V(C) \cap (Z_v \cup \{v\}) \neq \emptyset$. Then, C can intersect with at most $2k+1$ components in $\mathcal{D}$ and therefore the solution $\mathcal{C}$ can intersect with at most $2k^2 + k$ components in $\mathcal{D}$.*

We now construct a bipartite graph $\mathcal{H}$ with bipartition $(A = Z_v, B = \mathcal{D})$. We slightly abuse notation and assume that every component in $\mathcal{D}$ corresponds to a vertex in B and every vertex in Z_v corresponds to a vertex in A. For every $D_i \in \mathcal{D}$ and for every $z \in Z_v$, $(D_i, z) \in E(\mathcal{H})$ if and only if there exists $u \in V(D_i)$ such that $(u, z) \in E(G)$. After exhaustive application of Reduction Rule B4, every pair of vertices in G can have at most two edges between them. In particular, there can be at most two edges between any $z \in Z_v$ and v. Therefore, if the degree of v in G is more than $(2k^2 + k + 2)2k + 3k - 1$ then the number of components $|\mathcal{D}|$ is at least $(2k^2 + k + 2)2k$ (taking into account the at most $k-1$ neighbors of v in components containing a cycle as well as the at most $2k$ edges incident to v and some vertex in Z_v). Consequently, $|\mathcal{D}| \geq (2k^2 + k + 2)|Z_v|$. We are now ready to state our main reduction rule.

▶ **Reduction Rule B7.** *If there exists a vertex $v \in V(G)$ such that $d_G(v) > (2k^2 + k + 2)2k + 3k - 1$ then apply Lemma 13 with $q = 2k^2 + k + 2$ in the bipartite graph $\mathcal{H}$.*

- *Let $\mathcal{D}' \subseteq \mathcal{D}$ and $Z'_v \subseteq Z_v$ be the sets obtained after applying Lemma 13 with $q = 2k^2 + k + 2$, $A = Z_v$, and $B = \mathcal{D}$, such that Z'_v has a $(2k^2 + k + 2)$-expansion into $\mathcal{D}'$ in $\mathcal{H}$.*
- *Delete all the edges of the form $(u, v) \in E(G)$ such that $u \in D_i$ and $D_i \in \mathcal{D}'$.*
- *Add two parallel edges between v and every vertex in Z'_v.*

We now have all the required ingredients to bound the size of our kernel. From Theorem 1, we know that the graph has a feedback vertex set F of size at most $\mathcal{O}(k \log k)$. The degree of any vertex in the graph is at least three (Reduction Rule B2) and at most in $\mathcal{O}(k^3)$ (Reduction Rule B7). Theorem 16 follows from combining these facts with Lemma 15.

▶ **Lemma 15** ([7]). *Let $G = (V, E)$ be an undirected (multi) graph having minimum degree at least three, maximum degree at most d, and a feedback vertex set of size at most r. Then, $|V(G)| < (d+1)r$ and $|E(G)| < 2dr$.*

▶ **Theorem 16.** *For $t = 1$,* Pairwise Disjoint Cycle Packing *admits a kernel with $\mathcal{O}(k^4 \log k)$ vertices and $\mathcal{O}(k^4 \log k)$ edges.*

4.2 A polynomial compression for $t \geq 2$ (independent of t)

When $t \geq 2$, finding two vertices in G with $2k$ internally vertex-disjoint paths connecting them is enough to pack k cycles pairwise intersecting in at most 2 vertices. Hence, bounding the maximum degree is relatively easy. We first mark the feedback vertex set F and exhaustively apply Reduction Rule B1 and the following modified variant of Reduction Rule B2.

▶ **Reduction Rule B8.** *If there exists a set of vertices $P = \{v_1, \ldots, v_{t+2}\} \subseteq V(G)$ such that $G[P]$ is a path, $d_G(v_i) = 2$, $2 \leq i \leq t+1$, and $|P| \geq t+2$, then contract the edge v_1v_2.*

As before, for every vertex $v \in V(G)$, we apply the algorithm of Lemma 12. If the algorithm finds a v-flower of order k, we apply Reduction Rule B5. Otherwise, consider the connected components of the graph $G[V(G) \setminus (Z_v \cup \{v\})]$. We ignore the at most $k-1$ components that can contain a cycle and focus on the set $\mathcal{D} = \{D_1, D_2, \ldots, D_q\}$ of trees in which v has a neighbor (recall that $|N_G(v) \cap V(D)| \leq 1$ for all $D \in \mathcal{D}$ and each component D must have a neighbor in Z_v).

▶ **Reduction Rule B9.** *If $|\mathcal{D}| > 4k-2$ (or equivalently if $d_G(v) > 7k-3$) return a trivial yes-instance.*

Having bounded the maximum degree of any vertex by $\mathcal{O}(k)$, we immediately obtain a bound of $\mathcal{O}(k^2 \log k)$ on $|T_{\leq 1}|$, $|T_{\geq 3}|$, and the number of maximal degree-two paths in T_2. Recall that $T_{\leq 1}$, T_2, and $T_{\geq 3}$, are the sets of vertices in $T = G[V(G) \setminus F]$ having degree at most one in T, degree exactly two in T, and degree greater than two in T, respectively. To bound the size of T_2, note that if we mark all vertices in $F \cup N_G(F)$ we would have marked a total of $\mathcal{O}(k^2 \log k)$ vertices and the only unmarked vertices form (not necessarily maximal) degree-two paths in T_2 (and G), which we call segments. However, we know from Reduction Rule B8 that the size of any segment is at most $t+1$. Moreover, the total number of such segments is at most $\mathcal{O}(k^2 \log k)$. Putting it all together, we now have a kernel with $\mathcal{O}(tk^2 \log k)$ vertices.

▶ **Lemma 17.** *For any $t \geq 2$,* Pairwise Disjoint Cycle Packing *admits a kernel with $\mathcal{O}(tk^2 \log k)$ vertices.*

More work is needed to get rid of the dependence on t. The first step is to show that we can solve Pairwise Disjoint Cycle Packing in $c^{p(k)}n^{\mathcal{O}(1)}$ time, where c is a fixed constant and $p(.)$ is a polynomial function in k. In the second step, we introduce a "succinct" version of Pairwise Disjoint Cycle Packing, namely Succinct Pairwise Disjoint Cycle Packing, and show that we can reduce Pairwise Disjoint Cycle Packing to an instance of Succinct Pairwise Disjoint Cycle Packing where all the information can be encoded using a number of bits polynomially bounded in k alone.

Succinct Pairwise Disjoint Cycle Packing	**Parameter:** k
Input: An undirected (multi) graph G, integers k and t, a weight function $\alpha : V(G) \to \mathbb{N}$, and a weight function $\beta : E(G) \to \mathbb{N}$	
Question: Does G have at least k distinct cycles $C_1, \ldots, C_k$ such that $\alpha(V(C_i) \cap V(C_j)) \leq t$ and $\beta(E(C_i) \cap E(C_j)) \leq t$ for all $i \neq j$?	

▶ **Lemma 18.** *For any $t \geq 2$,* Pairwise Disjoint Cycle Packing *can be solved in $2^{k^3 \log k} n^{\mathcal{O}(1)}$ time.*

▶ **Theorem 19.** *For any $t \geq 2$, we can compress an instance of* Pairwise Disjoint Cycle Packing *to an equivalent instance of* Succinct Pairwise Disjoint Cycle Packing *using at most $\mathcal{O}(k^5 \log^2 k)$ bits. In other words,* Pairwise Disjoint Cycle Packing *admits a polynomial compression.*

References

1. Hasan Abasi, Nader H. Bshouty, Ariel Gabizon, and Elad Haramaty. On r-simple k-path. In *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 1–12, 2014. doi:10.1007/978-3-662-44465-8_1.
2. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, December 1996.
3. Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009. doi:10.1016/j.jcss.2009.04.001.
4. Hans L. Bodlaender and Arie M. C. A. Koster. Combinatorial optimization on graphs of bounded treewidth. *Comput. J.*, 51(3):255–269, May 2008.
5. Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theor. Comput. Sci.*, 412(35):4570–4578, 2011. doi:10.1016/j.tcs.2011.04.039.
6. Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
7. Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.
8. Holger Dell and Dániel Marx. Kernelization of packing problems. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 68–81, 2012. URL: http://portal.acm.org/citation.cfm?id=2095122&CFID=63838676&CFTOKEN=79617016.
9. Holger Dell and Dieter van Melkebeek. Satisfiability allows no nontrivial sparsification unless the polynomial-time hierarchy collapses. *J. ACM*, 61(4):23:1–23:27, 2014. doi:10.1145/2629620.
10. Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
11. Rod G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer-Verlag, 1997.
12. Andrew Drucker. New limits to classical and quantum instance compression. *SIAM J. Comput.*, 44(5):1443–1479, 2015. doi:10.1137/130927115.
13. Paul Erdős and Lajos Pósa. On independent circuits contained in a graph. *Canad. Journ. Math*, 17(0):347–352, 1965.
14. Henning Fernau, Alejandro López-Ortiz, and Jazmín Romero. Kernelization algorithms for packing problems allowing overlaps. In *Theory and Applications of Models of Computation – 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings*, pages 415–427, 2015. doi:10.1007/978-3-319-17142-5_35.
15. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

16 Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011. doi:10.1016/j.jcss.2010.06.007.

17 Ariel Gabizon, Daniel Lokshtanov, and Michal Pilipczuk. Fast algorithms for parameterized problems with relaxed disjointness constraints. In *Algorithms – ESA 2015 – 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 545–556, 2015. doi:10.1007/978-3-662-48350-3_46.

18 Danny Hermelin, Stefan Kratsch, Karolina Soltys, Magnus Wahlström, and Xi Wu. A completeness theory for polynomial (turing) kernelization. *Algorithmica*, 71(3):702–730, 2015. doi:10.1007/s00453-014-9910-8.

19 Danny Hermelin and Xi Wu. Weak compositions and their applications to polynomial lower bounds for kernelization. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 104–113, 2012. URL: http://portal.acm.org/citation.cfm?id=2095125&CFID=63838676&CFTOKEN=79617016.

20 Stefan Kratsch. Recent developments in kernelization: A survey. *Bulletin of the EATCS*, 113, 2014. URL: http://eatcs.org/beatcs/index.php/beatcs/article/view/285.

21 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Kernelization – preprocessing with a guarantee. In *The Multivariate Algorithmic Revolution and Beyond – Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, pages 129–161, 2012. doi:10.1007/978-3-642-30891-8_10.

22 Daniel Lokshtanov, Fahad Panolan, M.S. Ramanujan, and Saket Saurabh. Lossy kernelization. *arXiv:1604.04111*, 2016.

23 Rolf Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford University Press, Oxford, 2006.

24 Jan Ramon and Siegfried Nijssen. Polynomial-delay enumeration of monotonic graph classes. *The Journal of Machine Learning Research*, 10:907–929, 2009.

25 Daniel Ratner and Manfred K. Warmuth. NxN puzzle and related relocation problem. *J. Symb. Comput.*, 10(2):111–138, 1990. doi:10.1016/S0747-7171(08)80001-6.

26 Jazmín Romero and Alejandro López-Ortiz. The $\mathcal{G}$-packing with t-overlap problem. In *Algorithms and Computation – 8th International Workshop, WALCOM 2014, Chennai, India, February 13-15, 2014, Proceedings*, pages 114–124, 2014. doi:10.1007/978-3-319-04657-0_13.

27 Jazmín Romero and Alejandro López-Ortiz. A parameterized algorithm for packing overlapping subgraphs. In *Computer Science – Theory and Applications – 9th International Computer Science Symposium in Russia, CSR 2014, Moscow, Russia, June 7-11, 2014. Proceedings*, pages 325–336, 2014. doi:10.1007/978-3-319-06686-8_25.

28 Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Trans. Algorithms*, 6(2):32:1–32:8, April 2010. doi:10.1145/1721837.1721848.

The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems*

Andreas Emil Feldmann[1] and Dániel Marx[2]

1 **Department of Applied Mathematics, Charles University, Prague, Czech Republic; and**
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
feldmann.a.e@gmail.com

2 **Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary**
dmarx@cs.bme.hu

Abstract

Given a directed graph G and a list $(s_1, t_1), \ldots, (s_k, t_k)$ of terminal pairs, the Directed Steiner Network problem asks for a minimum-cost subgraph of G that contains a directed $s_i \to t_i$ path for every $1 \leq i \leq k$. The special case Directed Steiner Tree (when we ask for paths from a root r to terminals $t_1, \ldots, t_k$) is known to be fixed-parameter tractable parameterized by the number of terminals, while the special case Strongly Connected Steiner Subgraph (when we ask for a path from every t_i to every other t_j) is known to be W[1]-hard parameterized by the number of terminals. We systematically explore the complexity landscape of directed Steiner problems to fully understand which other special cases are FPT or W[1]-hard. Formally, if $\mathcal{H}$ is a class of directed graphs, then we look at the special case of Directed Steiner Network where the list $(s_1, t_1), \ldots, (s_k, t_k)$ of requests form a directed graph that is a member of $\mathcal{H}$. Our main result is a complete characterization of the classes $\mathcal{H}$ resulting in fixed-parameter tractable special cases: we show that if every pattern in $\mathcal{H}$ has the combinatorial property of being "transitively equivalent to a bounded-length caterpillar with a bounded number of extra edges," then the problem is FPT, and it is W[1]-hard for *every* recursively enumerable $\mathcal{H}$ not having this property. This complete dichotomy unifies and generalizes the known results showing that Directed Steiner Tree is FPT [Dreyfus and Wagner, *Networks* 1971], Strongly Connected Steiner Subgraph is W[1]-hard [Guo et al., *SIAM J. Discrete Math.* 2011], and Directed Steiner Network is solvable in polynomial-time for constant number of terminals [Feldman and Ruhl, *SIAM J. Comput.* 2006], and moreover reveals a large continent of tractable cases that were not known before.

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

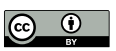
Keywords and phrases Directed Steiner Tree, Directed Steiner Network, fixed-parameter tractability, dichotomy

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.27

1 Introduction

Steiner Tree is a basic and well-studied problem of combinatorial optimization: given an edge-weighted undirected graph G and a set $R \subseteq V(G)$ of terminals, it asks for a minimum-

* Supported by ERC Starting Grant PARAMTIGHT (No. 280152) and OTKA grant NK105645.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 27; pp. 27:1–27:14

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

cost tree connecting the terminals. The problem is well known to be NP-hard, in fact, it was one of the 21 NP-hard problems identified by Karp's seminal paper [22]. There is a large literature on approximation algorithms for Steiner Tree and its variants, resulting for example in constant-factor approximation algorithms for general graphs and approximation schemes for planar graphs (see [8, 15, 9, 4, 3, 2, 7, 26, 24, 23, 1, 17]). From the viewpoint of parameterized algorithms, the first result is the classic dynamic-programming algorithm of Dreyfus and Wagner [17] from 1971, which solves the problem with $k = |R|$ terminals in time $3^k \cdot n^{O(1)}$, showing that the problem is fixed-parameter tractable (FPT) parameterized by the number of terminals. More recently, the running time was improved to $2^k \cdot n^{O(1)}$ by Björklund et al. [5] using the technique of fast subset convolution. Steiner Forest is the generalization where the input contains an edge-weighted graph G and a list $(s_1, t_1), \ldots, (s_k, t_k)$ of pairs of terminals and the task is to find a minimum-cost subgraph containing an s_i–t_i path for every $1 \leq i \leq k$. The fixed-parameter tractability of Steiner Forest follows from the observation that the connected components of the solution induces a partition on the set $\{s_1, \ldots, s_k, t_1, \ldots, t_k\}$ of terminals, and hence we can solve the problem by for example trying every partition and invoking a Steiner Tree algorithm for each class of the partition.

On directed graphs, Steiner problems can become significantly harder, and while there is a richer landscape of variants, very few results are known [21, 11, 18, 10, 27, 14, 13]. A natural and well-studied generalization of Steiner Tree to directed graphs is Directed Steiner Tree (DST), where an arc-weighted directed graph G and terminals $r, t_1, \ldots, t_k$ are given and the tasks is to find a minimum-cost subgraph containing an $r \to t_i$ path for every $1 \leq i \leq k$. Using essentially the same techniques as in the undirected case [5, 17], one can show that this problem is also FPT parameterized by the number of terminals. An equally natural generalization of Steiner Tree to directed graphs is the Strongly Connected Steiner Subgraph (SCSS) problem, where an arc-weighted directed graph G with terminals $t_1, \ldots, t_k$ is given, and the task is to find a minimum-cost subgraph containing a $t_i \to t_j$ path for any $1 \leq i, j \leq k$ with $i \neq j$. Guo et al. [21] showed that, unlike DST, the SCSS problem is W[1]-hard parameterized by the number k of terminals (see also [14]). A common generalization of DST and SCSS is the Directed Steiner Network (DSN) problem (also called Directed Steiner Forest or Point-to-Point Connection), where an arc-weighted directed graph G and a list $(s_1, t_1), \ldots, (s_k, t_k)$ of terminal pairs are given and the task is to find a minimum-cost subgraph containing an $s_i \to t_i$ path for every $1 \leq i \leq k$. Being a generalization of SCSS, the Directed Steiner Network problem is also W[1]-hard, but Feldman[1] and Ruhl [18] showed that the problem is solvable in time $n^{O(k)}$, that is, in polynomial time for every constant k.

Besides Directed Steiner Tree, what other special cases of Directed Steiner Network are fixed-parameter tractable? Our main result gives a complete map of the complexity landscape of directed Steiner problems, precisely describing all the FPT/W[1]-hard variants and revealing highly non-trivial generalizations of Directed Steiner Tree that are still tractable. Our results are expressed in the following formal framework. The pairs $(s_1, t_1), \ldots, (s_k, t_k)$ in the input of DSN can be interpreted as a directed (unweighted) *pattern graph* on a set R of terminals. If this pattern graph is an out-star, then the problem is precisely DST; if it is a bidirected clique, then the problem is precisely SCSS. More generally, if $\mathcal{H}$ is any class of graphs, then we define the Directed Steiner $\mathcal{H}$-Network ($\mathcal{H}$-DSN) problem as the restriction of DSN where the pattern graph is a member of $\mathcal{H}$. That is, the

[1] We note that Jon Feldman (co-author of [18]) is not the same person as Andreas Emil Feldmann (co-author of this paper).

Figure 1 Two 4-caterpillars: an out- (left) and an in-caterpillar (right).

input of $\mathcal{H}$-DSN is an arc-weighted directed graph G, a set $R \subseteq V(G)$ of terminals, and an unweighted directed graph $H \in \mathcal{H}$ on R; the task is to find a minimum-cost network $N \subseteq G$ such that N contains an $s \to t$ path for every $st \in E(H)$.

We give a complete characterization of the classes $\mathcal{H}$ for which $\mathcal{H}$-DSN is FPT or W[1]-hard. We need the following definition of "almost-caterpillar graphs" to describe the borderline between the easy and hard cases (see Figure 1).

▶ **Definition 1.** A λ_0-*caterpillar* graph is constructed as follows. Take a directed path $(v_1, \ldots, v_{\lambda_0})$ from v_1 to v_{λ_0}, and let $W_1, \ldots, W_{\lambda_0}$ be pairwise disjoint vertex sets such that $v_i \in W_i$ for each $i \in \{1, \ldots, \lambda_0\}$. Now add edges such that either every W_i forms an out-star with root v_i, or every W_i forms an in-star with root v_i. In the former case we also refer to the resulting λ_0-caterpillar as an *out-caterpillar*, and in the latter as an *in-caterpillar*. A 0-caterpillar is the empty graph. The class $\mathcal{C}_{\lambda,\delta}$ contains all directed graphs H such that there is a set of edges $F \subseteq E(H)$ of size at most δ for which the remaining edges $E(H) \setminus F$ span a λ_0-caterpillar for some $\lambda_0 \leq \lambda$.

If there is an $s \to t$ path in the pattern graph H for two terminals $s, t \in R$, then adding the edge st to H does not change the problem: connectivity from s to t is already implied by H, hence adding this edge does not change the feasible solutions. That is, adding a transitive edge does not change the solution space and hence it is really only the transitive closure of the pattern H that matters. We say that two pattern graphs are *transitively equivalent* if their transitive closures are isomorphic. We denote the class of patterns that are transitively equivalent to some pattern of $\mathcal{C}_{\lambda,\delta}$ by $\mathcal{C}^*_{\lambda,\delta}$. Our main result is a sharp dichotomy saying that $\mathcal{H}$-DSN is FPT if every pattern of $\mathcal{H}$ is transitively equivalent to an almost-caterpillar graph and it is W[1]-hard otherwise. We measure the running time in λ, δ, and the *vertex cover number* τ of the input pattern H, i.e. τ is the size of the smallest vertex subset W of H such that every edge of H is incident to a vertex of W.

▶ **Theorem 2.** *Let $\mathcal{H}$ be a recursively enumerable class of patterns.*

1. *If there are constants λ and δ such that $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$, then $\mathcal{H}$-DSN with parameter $k = |R|$ is FPT and can be solved in $2^{O(k+\max\{\omega^2,\, \tau\omega \log \omega)\})} n^{O(\omega)}$ time, where $\omega = (1+\lambda)(\lambda+\delta)$ and τ is the vertex cover number of the given input pattern $H \in \mathcal{H}$.*
2. *Otherwise, if there are no such constants λ and δ, then the problem is* W[1]*-hard for parameter k.*

Invoking Theorem 2 with specific classes $\mathcal{H}$, we can obtain algorithmic or hardness results for specific problems. For example, we may easily recover the following facts:

- If $\mathcal{H}_{\text{DST}}$ is the class of all out-stars, then $\mathcal{H}_{\text{DST}}$-DSN is precisely the DST problem. As $\mathcal{H}_{\text{DST}} \subseteq \mathcal{C}^*_{1,0}$ holds, Theorem 2(1) recovers the fact that DST can be solved in time $2^{O(k)} n^{O(1)}$ and is hence FPT parameterized by the number $k = |R|$ of terminals [17, 5].
- If $\mathcal{H}_{\text{SCSS}}$ is the class of all bidirected cliques, then $\mathcal{H}_{\text{SCSS}}$-DSN is precisely the SCSS problem. One can observe that $\mathcal{H}_{\text{SCSS}}$ is not contained in $\mathcal{C}^*_{\lambda,\delta}$ for any constants λ, δ (for example, because every graph in $\mathcal{C}_{\lambda,\delta}$ has at most $\lambda + 2\delta$ vertices with both positive

in-degree and positive out-degree, and this remains true also for the graphs in $\mathcal{C}^*_{\lambda,\delta}$). Hence Theorem 2(2) recovers the fact that SCSS is W[1]-hard [21].

- Let $\mathcal{H}_d$ be the class of directed graphs with at most d edges. As $\mathcal{H}_d \subseteq \mathcal{C}^*_{0,d}$ holds, Theorem 2(1) recovers the fact that Directed Steiner Network with at most d requests is polynomial-time solvable for every constant d [18]. Note that any pattern of $\mathcal{H}_{\text{SCSS}}$ is transitively equivalent to a bidirected star, which has vertex cover number $\tau = 1$. Hence for the important spacial case of SCSS, our algorithm recovers the running time of $2^{O(d \log d)} n^{O(d)} = n^{O(d)}$ given in [18].
- Very recently, Suchý [25] studied the following generalization of DST and SCSS: in the q-Root Steiner Tree (q-RST) problem, a set of q roots and a set of k leaves are given, and the task is to find a minimum-cost network where the roots are in the same strongly connected component and every leaf can be reached from every root. Building on the work of [18], Suchý [25] presented an algorithm with running time $2^{O(k)} \cdot n^{O(q)}$ for this problem, which shows that it is FPT for every constant q. Let $\mathcal{H}_{q\text{-RST}}$ be the class of directed graphs that are obtained from an out-star by making $q-1$ of the edges bidirected. Observe that $\mathcal{H}_{q\text{-RST}}$ is a subset of $\mathcal{C}_{1,q-1}$, that q-RST can be expressed by an instance of $\mathcal{H}_{q\text{-RST}}$-DSN, and that any pattern of $\mathcal{H}_{q\text{-RST}}$ has vertex cover number $\tau = 1$. Thus Theorem 2(1) implies that q-RST can be solved in time $2^{O(k+q \log q)} \cdot n^{O(q)} = 2^{O(k)} \cdot n^{O(q)}$, recovering the fact that it is FPT for every constant q.

Thus the algorithmic side of Theorem 2 unifies and generalizes three algorithmic results: the fixed-parameter tractability of DST (which is based on dynamic programming on the tree structure of the solution) and q-RST (which is based on simulating a "pebble game"), and also the polynomial-time solvability of DSN with constant number of requests (which also is based on simulating a "pebble game"). Let us point out that our algorithmic results are significantly more general than just the unification of these three results: the generalization from stars to bounded-length caterpillars is already a significant extension and very different from earlier results. We consider it a major success of the systematic investigation that, besides finding the unifying algorithmic ideas generalizing all previous results, we were able to find tractable special cases in an unexpected new direction.

There is a surprising non-monotonicity in the classification result of Theorem 2. As DST is FPT and SCSS is W[1]-hard, one could perhaps expect that $\mathcal{H}$-DSN becomes harder as the pattern become denser. However, it is possible that the addition of further requests makes the problem easier. For example, if $\mathcal{H}$ contains every graph that is the vertex-disjoint union of two out-stars, then $\mathcal{H}$-DSN is classified to be W[1]-hard by Theorem 2(2). However, if we consider those graphs where there is also a directed edge from the center of one star to the other star, then these graphs are 2-caterpillars (i.e., contained in $\mathcal{C}_{2,0}$) and hence $\mathcal{H}$-DSN becomes FPT by Theorem 2(1). This unexpected non-monotonicity further underlines the importance of completely mapping the complexity landscape of the problem area: without complete classification, it would be very hard to predict what other tractable/intractable special cases exist.

We mention that one can also study the vertex-weighted version of the problem, where the input graph has weights on the vertices and the goal is to minimize the total vertex-weight of the solution. In general, vertex-weighted problems can be more challenging than edge-weighted variants [15, 4, 23, 12]. However, for general directed graphs, there are easy transformations between the two variants. Thus the results of this paper can be interpreted for the vertex-weighted version as well.

1.1 Our techniques

We prove Theorem 2 the following way. In Section 2, we first establish the combinatorial bound that there is a solution whose cutwidth, and hence also (undirected) treewidth, is bounded by the number of requests.

▶ **Theorem 3.** *A minimal solution M to a pattern H has cutwidth at most $7m$ if $m = |E(H)|$.*

Then in Section 3 we go on to generalize this to almost-caterpillars, showing that if the pattern is in $\mathcal{C}^*_{\lambda,\delta}$, then the (undirected) treewidth can be bounded in λ and δ.

▶ **Theorem 4.** *The treewidth of a minimal solution to any pattern graph in $\mathcal{C}^*_{\lambda,\delta}$ is at most $7(1+\lambda)(\lambda+\delta)$.*

This combinatorial bound can be exploited in an algorithm that restricts the search for a bounded-treewidth solution.

▶ **Theorem 5.** *Let an instance of $\mathcal{H}$-DSN be given by a graph G with n vertices, and a pattern H on k terminals with vertex cover number τ. If the optimum solution to H in G has treewidth ω then the optimum can be computed in time $2^{O(k+\max\{\omega^2,\, \tau\omega\log\omega\})}n^{O(\omega)}$.*

Combining Theorem 4 and Theorem 5 proves the algorithmic side of Theorem 2. We remark that the proof is completely self-contained (with the exception of some basic facts on treewidth) and in particular we do not build on the algorithms of Feldman and Ruhl [18]. As combining Theorem 3 and Theorem 5 already proves that DSN with a constant number of requests can be solved in polynomial time, as a by-product this gives an independent proof for the result of Feldman and Ruhl [18]. One can argue which algorithm is simpler, but perhaps our proof (with a clean split of a combinatorial and an algorithmic statement) is more methodological and better reveals the underlying reason why the problem is tractable.

Finally, in Section 4 we show that whenever the patterns in $\mathcal{H}$ are not transitively equivalent to almost-caterpillars, the problem is W[1]-hard. We first show that there is only a small number of obstacles for not being transitively equivalent to almost-caterpillars: the graph class contains (possibly after identification of vertices) arbitrarily large strongly connected graphs, pure diamonds, or flawed diamonds (see Lemma 22 for the precise statement). We provide a separate W[1]-hardness proof for each of these cases, completing the proof of the hardness side of Theorem 2.

Due to space limitations we defer all missing proofs to the full version of this extended abstract, including the algorithm that implies Theorem 5.

2 The cutwidth of minimal solutions for bounded-size patterns

Consider a *minimal* solution M to an instance of $\mathcal{H}$-DSN, in which no edge can be removed without making the solution infeasible. The goal of this section is to prove Theorem 3: we bound the *cutwidth* of a minimal solution M to a pattern H in terms of $m = |E(H)|$. A *layout* of a graph G is an injective function $\psi : V(G) \to \mathbb{N}$ inducing a total order on the vertices of G. Given a layout, we define the set $V_i = \{v \in V(G) \mid \psi(v) \leq i\}$ and say that an edge *crosses the cut* $(V_i, \overline{V}_i)$ if it has one endpoint in V_i and one endpoint in $\overline{V}_i := V(G) \setminus V_i$. The *cutwidth of the layout* is the maximum number of edges crossing any cut $(V_i, \overline{V}_i)$ for any $i \in \mathbb{N}$. The cutwidth of a graph is the minimum cutwidth over all its layouts.

Like Feldman and Ruhl [18], we consider the two extreme cases of *directed acyclic graphs (DAGs)* and *strongly connected components (SCCs)* in our proof. Contracting all SCCs of M without removing parallel edges sharing the same head and tail, but removing

the resulting self-loops, results in a directed acyclic multi-graph D, the so-called *condensation graph* of M. We bound the cutwidth of D and the SCCs of M separately, and then put together these two bounds to obtain a bound for the cutwidth of M. As we will see, bounding the cutwidth of the acyclic multi-graph D and putting together the bounds are fairly simple. The main technical part is bounding the cutwidth of the SCCs.

We will need two simple facts about cutwidth. First, the cutwidth of an acyclic multi-graph can be bounded using the existence of a *topological ordering* of the vertices. That is, for any acyclic graph G there is an injective function $\varphi : V(G) \to \mathbb{N}$ such that $\varphi(u) < \varphi(v)$ if $uv \in E(G)$. Note that such a function in particular is a layout.

▶ **Lemma 6.** *The layout given by a topological ordering φ_D of an acyclic directed multi-graph D that is the union of m paths, has cutwidth at most m.*

▶ **Lemma 7.** *Let G be a directed graph and D be its condensation multi-graph. If the cutwidth of D is x and the cutwidth of every SCC of G is at most y, then the cutwidth of G is at most $x + y$.*

▶ **Lemma 8.** *Any SCC U of a minimal solution M to a pattern H with at most m edges has cutwidth at most $6m$.*

Proof. First we establish that U is a minimal solution to a certain pattern.

▶ **Claim 9.** *U is a minimal solution to a pattern H_U with at most m edges.*

Let R_U be the terminals in the pattern H_U given by Claim 9 and let us select an arbitrary root $t \in R_U$. Note that H_U has at most m edges, hence $|R_U| \leq 2m$. Let S_{in} (resp., S_{out}) be an in-star (resp., out-star) connecting t with every other vertex of R_U. As U is a strongly connected graph containing every vertex of R_U, it is also a solution to the pattern S_{in} on R_U. Let us select an $A_{in} \subseteq U$ that is a minimal solution to S_{in}; it is not hard to see that A_{in} is an in-arborescence with at most $2m$ leaves. Similarly, let $A_{out} \subseteq U$ be an out-arborescence that is a minimal solution to S_{out}. Observe that U has to be exactly $A_{in} \cup A_{out}$: if there is an edge $e \in E(U)$ that is not in $A_{in} \cup A_{out}$, then $U \setminus e$ still contains a path from every vertex of R_U to every other vertex of R_U though t, contradicting the fact that U is a minimal solution to pattern H_U.

Let Z be the set of edges obtained by reversing the edges in $E(A_{in}) \setminus E(A_{out})$. As reversing edges does not change the cutwidth, bounding the cutwidth of $A_{out} \cup Z$ will also imply a bound on the cutwidth of $U = A_{in} \cup A_{out}$.

▶ **Claim 10.** *The union $A_{out} \cup Z$ is a directed acyclic graph.*

Claim 10 implies a topological ordering on the vertices of $A_{out} \cup Z$. This order can be used as a layout for U. Using some more structural insights, the number of edges crossing a given cut can be bounded in the number of edges of the pattern graph, as the following claim shows.

▶ **Claim 11.** *Any topological ordering φ of the graph $A_{out} \cup Z$ has cutwidth at most $6m$.*

As the underlying undirected graph of U and $A_{out} \cup Z$ are the same, Claim 11 implies that the cutwidth of U is at most $6m$. This completes the proof of Lemma 8. ◀

The proof of Theorem 3 follows easily from putting together the ingredients. We remark that the bound on the cutwidth in Claim 11 is asymptotically tight: Take a constant degree expander on m vertices. It has treewidth $\Omega(m)$ [20], and so its cutwidth is at least as large. Now bi-direct each (undirected) edge $\{u, v\}$ by replacing it with the directed edges uv and vu.

Next subdivide every edge uv to obtain edges ut and tv for a new vertex t, and make t a terminal of R. This yields a strongly connected instance G. The pattern graph H for this instance is a cycle on R, which has $O(m)$ edges, since the terminals are subdivision points of bi-directed edges of a constant degree graph with m vertices. As H is strongly connected, every minimal solution to H contains the edges ut and tv incident to each terminal t. Thus a minimal solution contains all of G and has cutwidth $\Omega(m)$. Since G is strongly connected, it also contains the required arborescences A_{in} and A_{out}.

3 The treewidth of minimal solutions to almost-caterpillar patterns

In this section, we prove that any minimal solution M to a pattern $H \in \mathcal{C}^*_{\lambda,\delta}$ has the following structure.

▶ **Theorem 12.** *A minimal solution M to a pattern $H \in \mathcal{C}^*_{\lambda,\delta}$ consists of a subgraph M^c that is a minimal solution to a sub-pattern H^c of H with at most $(1+\lambda)(\lambda+\delta)$ edges, and a forest $M \setminus M^c$ of out-arborescences, each of which intersects M^c only at the root.*

According to Theorem 3, the cutwidth of the core M^c is therefore at most $7(1+\lambda)(\lambda+\delta)$. It is well known [6] that the cutwidth is an upper bound on the treewidth of a graph, and so also the treewidth of M^c is at most $7(1+\lambda)(\lambda+\delta)$. It is easy to see that attaching any number of arborescences to M^c does not increase the treewidth. Thus we obtain Theorem 4, which is the basis for our algorithm to solve $\mathcal{H}$-DSN in case every pattern of $\mathcal{H}$ is transitively equivalent to an almost-caterpillar.

In particular, when adding δ edges to the pattern of the DST problem, which is a single out-star, i.e., a 1-caterpillar, then the pattern becomes a member of $\mathcal{C}_{1,\delta}$ and hence our result implies a linear treewidth bound of $O(\delta)$. The example given at the end of Section 2 also shows that there are patterns $H \in \mathcal{C}_{\lambda,\delta}$ for which every minimal solution has treewidth $\Omega(\lambda+\delta)$: just consider the case when H is a cycle of length $\lambda+\delta$ (i.e., it contains a trivial caterpillar graph). One interesting question is whether the treewidth bound of $7(1+\lambda)(\lambda+\delta)$ in Theorem 4 is tight. We conjecture that the treewidth of any minimal solution to a pattern graph $H \in \mathcal{C}^*_{\lambda,\delta}$ actually is $O(\lambda+\delta)$.

Proof (of Theorem 12). Let M be a minimal solution to a pattern $H \in \mathcal{C}^*_{\lambda,\delta}$. Since every pattern in $\mathcal{C}^*_{\lambda,\delta}$ has a transitively equivalent pattern in $\mathcal{C}_{\lambda,\delta}$ and replacing a pattern with a transitively equivalent pattern does not change the space of feasible solutions, we may assume that H is actually in $\mathcal{C}_{\lambda,\delta}$, i.e., H consists of a caterpillar of length at most λ and δ additional edges.

The statement is trivial if $|E(H)| \leq \delta$. Otherwise, according to Definition 1, H contains a λ_0-caterpillar for some $1 \leq \lambda_0 \leq \lambda$ and at most δ additional edges. Hence let us fix a set F of at most δ edges of H, such that the remaining edges of H form a λ_0-caterpillar C, for some $1 \leq \lambda_0 \leq \lambda$, with a path $(v_1, \ldots, v_{\lambda_0})$ on the roots of the stars S_i. We only consider the case when C is an out-caterpillar as the other case is symmetric, i.e., every S_i is an out-star. Define $I = H \setminus \bigcup_{i=1}^{\lambda_0} S_i$ to be all of H except the stars. Note that $|E(I)| \leq \lambda+\delta$. We fix a subgraph M_I of M that is a minimal solution to the sub-pattern I, and for every $st \in E(I)$ we fix a path P_{st} in M_I. Note that M_I is the union of these at most $\lambda+\delta$ paths, since M_I is a minimal solution. For each star S_i, let us consider a minimal solution $M_{S_i} \subseteq M$ to S_i; note that M_{S_i} has to be an out-arborescence.

For $i \in \{1, \ldots, \lambda_0\}$, let ℓ be a leaf of S_i, and let e be an edge of M. If $M \setminus e$ has no path from v_i to ℓ, then we say that e is *ℓ-necessary*. More generally, we say that e is *i-necessary* if e is ℓ-necessary for some leaf ℓ of S_i.

▶ **Claim 13.** *Let P be a path in M, and for some $i \in \{1, \ldots, \lambda_0\}$, let $W_i \subseteq E(M)$ contain all i-necessary edges f for which $f \notin E(P)$, but the head of f is a vertex of P. Then there exists one leaf ℓ of S_i such that every $f \in W_i$ is ℓ-necessary.*

Using this observation, we identify the core M^c of M using the at most $\lambda + \delta$ paths P_{st} that make up M_I, and then selecting an additional at most λ_0 paths for each P_{st}. To construct M^c together with its pattern graph H^c, we initially let $M^c = M_I$ and $H^c = I$ and repeat the following step for every $st \in E(I)$ and $1 \leq i \leq \lambda_0$. For a given st and i, let us check if there are i-necessary edges $f \notin E(P_{st})$ that have their heads on the path $P_{st} \subseteq M_I$. If so, then by Claim 13 all these edges are ℓ-necessary for some leaf ℓ of S_i. We add an arbitrary path of M from v_i to ℓ (which contains all these edges) to M^c and add the edge $v_i\ell$ to H^c. After repeating this step for every $st \in E(H)$ and i, we remove superfluous edges from M^c: as long as there is an edge $e \in E(M^c)$, which can be removed while maintaining feasibility for the pattern H^c, i.e., for every $vw \in E(H^c)$ there is a $v \to w$ path in M^c not containing e, we remove e. Finally, we remove any isolated vertices from M^c.

Note that the resulting network M^c is a minimal solution to H^c by construction. Also note that H^c contains at most $\lambda + \delta$ edges from I and at most $\lambda_0 \leq \lambda$ additional edges for each edge of I, so that $|E(H^c)| \leq (1 + \lambda)(\lambda + \delta)$. We prove that the remaining graph $M^c \setminus E(M)$ consists of arborescences, each of which intersects M^c only at the root. For this, we rely on the following key observation.

▶ **Claim 14.** *If a vertex u has at least two incoming edges in M, then every such edge is in the core M^c.*

Proof. First we show that there is an $st \in E(I)$ such that every $s \to t$ path in M goes through u. Suppose for contradiction that for every $st \in E(I)$ there is a path from s to t in M avoiding u. Since M is a minimal solution, the edges entering u must then be needed for some stars S_i of the pattern H instead. Let e and f be two edges entering u. As e and f have the same head, they cannot be part of the same out-arborescence M_{S_i}. Therefore, there are indices $i < j$ such that (w.l.o.g.) e is i-necessary and f is j-necessary.

There is a path in M from the root v_i of S_i to the root v_j of S_j, due to the path $(v_1, \ldots, v_{\lambda_0})$ in the caterpillar $C \subseteq H$. Since path $(v_1, \ldots, v_{\lambda_0})$ is part of I, our assumption on e and f implies that there is a path P in M from v_i to v_j that avoids both e and f. As $f \in E(M_{S_j})$, there is a path Q in M starting in v_j and passing through f. This path cannot contain e, as e and f have the same head u. The existence of P and Q implies that u can be reached from v_i by a path through v_j and f, avoiding the edge e. Thus for any edge $v_i\ell \in E(S_i)$, if there is a $v_i \to \ell$ path going through e (and hence vertex u), then it can be rerouted to avoid e and use edge f instead. This however contradicts the fact that e is i-necessary.

We have proved that there is an $st \in E(I)$ such that every $s \to t$ path in M goes through u. Suppose that there is an edge $e \notin E(M^c)$ entering u. If e is needed for some $s't' \in E(I)$ in M, then e is also present in M^c, and we are done. Otherwise, as M is a minimal solution, edge e is i-necessary for some $i \in \{1, \ldots, \lambda_0\}$. Consider now the step in the construction of M^c when we considered $st \in E(I)$ and integer i. As we have shown, the $s \to t$ path P_{st} goes through u. Thus e is an i-necessary edge not in $E(P_{st})$ such that its head is on P_{st}. This means that we identified a leaf ℓ of S_i such that e is ℓ-necessary, introduced $v_i\ell$ into H^c, and added a $v_i \to \ell$ path to H^c, which had to contain e. Moreover, since all paths from v_i to ℓ in M pass through e, edge e then remains in M^c when removing superfluous edges. ◀

We are now ready to show that every component of the remaining part is an out-arborescence and intersects the core only in a single vertex.

▶ **Claim 15.** *The remaining graph $M^+ := M \setminus E(M^c)$ is a forest of out-arborescences, each of which intersects M^c only at the root.*

Since we have already established that M^c is a minimal solution to H^c with $|E(H^c)| \leq (1+\lambda)(\lambda+\delta)$, Claim 15 completes the proof of Theorem 12. ◀

4 Characterizing the hard cases

We now turn to proving the second part of Theorem 2, i.e., that $\mathcal{H}$-DSN is W[1]-hard for every class $\mathcal{H}$ where the patterns are not transitively equivalent to almost-caterpillars.

▶ **Theorem 16.** *Let $\mathcal{H}$ be a recursively enumerable class of patterns for which there are no constants λ and δ such that $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$. Then the problem $\mathcal{H}$-DSN is* W[1]*-hard for parameter k.*

A major technical simplification is to assume that the class $\mathcal{H}$ is closed under identifying terminals and transitive equivalence. As we show in Section 4.1, this assumption is not really restrictive: it is sufficient to prove hardness for the closure of $\mathcal{H}$ under identification and transitive equivalence, since any W[1]-hardness result for the closure can be transferred to $\mathcal{H}$. For classes closed under these operations, it is possible to give an elegant characterization of the classes that are not almost-caterpillars. There are only a few very specific reasons why a class $\mathcal{H}$ is not in $\mathcal{C}^*_{\lambda,\delta}$ for any λ and δ: either $\mathcal{H}$ contains every directed cycle, or $\mathcal{H}$ contains every "pure diamond," or $\mathcal{H}$ contains every "flawed diamond" (see Section 4.2 for the precise definitions). Then in Section 4.3, we provide a W[1]-hardness proof for each of these cases, completing the hardness part of Theorem 2.

4.1 Closed classes

We define the operation of *identifying terminals* in the following way: given a partition $\mathcal{V}$ of the vertex set $V(H)$ of a pattern graph H, each set $W \in \mathcal{V}$ is identified with a single vertex of W, after which any resulting isolated vertices and self-loops are removed, while parallel edges having the same head and tail are replaced by only one copy of that edge. A class of patterns is *closed* under this operation if for any pattern H in the class, all patterns that can be obtained by identifying terminals are also in the class. Similarly, we say that a class $\mathcal{H}$ is closed under transitive equivalence if whenever H and H' are two transitively equivalent patterns such that $H \in \mathcal{H}$, then H' is also in $\mathcal{H}$. The closure of the class $\mathcal{H}$ under identifying terminals and transitive equivalence is the smallest closed class $\mathcal{H}' \supseteq \mathcal{H}$. It is not difficult to see that any member of the closure can be obtained by a replacement with a transitively equivalent pattern and a single application of identifying terminals.

The following lemma shows that if we want to prove W[1]-hardness for a class, then it is sufficient to prove hardness for its closure. More precisely, due to an slight technicality, the actual statement we prove is that it is sufficient to prove W[1]-hardness for a decidable subclass of the closure.

▶ **Lemma 17.** *Let $\mathcal{H}$ be a recursively enumerable class of patterns, let $\mathcal{H}'$ be the closure of $\mathcal{H}$ under identifying terminals and transitive equivalence, and let $\mathcal{H}''$ be a decidable subclass of $\mathcal{H}'$. There is a parameterized reduction from $\mathcal{H}''$-DSN to $\mathcal{H}$-DSN with parameter k.*

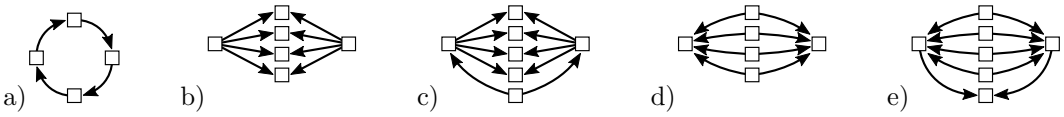


Figure 2 The obstruction appearing in Lemma 19: a) a directed cycle of length 4, b) a pure 4-out-diamond, c) a flawed 4-out-diamond, d) a pure 4-in-diamond, e) a flawed 4-in-diamond.

4.2 Obstructions: SCCs and diamonds

To show the hardness for a closed class that is not the subset of $\mathcal{C}^*_{\lambda,\delta}$ for any λ and δ, we will characterize such a class in terms of the occurrence of arbitrarily large cycles, and another class of patterns called "diamonds" (cf. Figure 2).

▶ **Definition 18.** A *pure α-diamond* graph is constructed as follows. Take a vertex set L of size $\alpha \geq 1$, and two additional vertices r_1 and r_2. Now add edges such that L is the leaf set of either two in-stars or two out-stars S_1 and S_2 with roots r_1 and r_2, respectively. If we add an additional vertex x with edges r_1x and r_2x if S_1 and S_2 are in-stars, and edges xr_1 and xr_2 otherwise, the resulting graph is a *flawed α-diamond*. We refer to both pure α-diamonds and flawed α-diamonds as *α-diamonds*. If S_1 and S_2 are in-stars we also refer to the resulting α-diamonds as *in-diamonds*, and otherwise as *out-diamonds*.

The goal of this section is to prove the following useful characterization precisely describing classes that are not almost-caterpillars.

▶ **Lemma 19.** *Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive closure. Exactly one of the following statements is true:*

- *$\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$ for some constants λ and δ.*
- *$\mathcal{H}$ contains every directed cycle, or every pure in-diamond, or every pure out-diamond, or every flawed in-diamond, or every flawed out-diamond.*

For the proof of Theorem 16, we only need the fact that at least one of these two statements hold: if the class $\mathcal{H}$ is not in $\mathcal{C}^*_{\lambda,\delta}$, then we can prove hardness by observing that $\mathcal{H}$ contains one of the hard classes. For the sake of completeness, we give a simple proof that the two statements cannot hold simultaneously in the full version of this extended abstract.

Showing that at least one of the two statements of Lemma 19 hold is not as easy to prove. First, the following two lemmas show how a large cycle or a large diamond can be identified if certain structures appear in a pattern. The main part of the proof is to show that if $\mathcal{H}$ contains patterns that are arbitrarily far from being a caterpillar, then one of these two lemmas can be invoked (see Lemma 22).

▶ **Lemma 20.** *Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive closure. If some $H \in \mathcal{H}$ contains a matching of size α, then $\mathcal{H}$ contains a directed cycle of length α.*

Proof. A *matching* of a graph is a subset M of its edges such that no two edges of M share a vertex. A matching $e_1, \ldots, e_\alpha$ of α edges can be transformed into a cycle of length α by identifying the head of e_i and tail of e_{i+1} (and the head of e_α with the tail of e_1). All remaining vertices that do not belong to the cycle can then be identified with any vertex of the cycle, so that the resulting graph consists of the cycle and some additional edges. Since $\mathcal{H}$ is closed under identifying terminals, this graph would be contained in $\mathcal{H}$. As this graph is strongly connected and $\mathcal{H}$ is closed also under transitive equivalence, we can conclude that $\mathcal{H}$ contains a cycle of length α. ◀

Next we give a sufficient condition for the existence of large diamonds. We say that an edge uv of a graph H is *transitively non-redundant* if there is no $u \to v$ path in $H \setminus uv$.

▶ **Lemma 21.** *Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive equivalence. Let $H \in \mathcal{H}$ be a pattern graph that contains two out-stars (or two in-stars) S_1 and S_2 as induced subgraphs, with at least α edges each and roots r_1 and r_2, respectively. The class $\mathcal{H}$ contains an α-diamond if*

1. *H contains neither a path from r_1 to r_2, nor from r_2 to r_1,*
2. *the leaves of S_1 and S_2 have out-degree 0 (if S_1 and S_2 are out-stars) or in-degree 0 (if S_1 and S_2 are in-stars), and*
3. *the edges of the stars are transitively non-redundant.*

To show that at least one of the two statements of Lemma 19 hold, we prove that if the second statement is false, then the first statement is true. That is, if $\mathcal{H}$ does not contain all cycles (i.e., there is an α_1 such that $\mathcal{H}$ contains no cycle larger than α_1), $\mathcal{H}$ does not contain all pure out-diamonds (i.e., there is an α_2 such that $\mathcal{H}$ contains no pure out-diamond larger than α_2), etc., then $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$ for some constants λ and δ. In other words, if we let α to be the maximum of α_1, α_2, etc., then we may assume that $\mathcal{H}$ contains no pure of flawed α-diamond or cycle of length α, and we need to prove $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$ under this assumption. Thus the following lemma completes the proof of Lemma 19.

▶ **Lemma 22.** *Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive equivalence. If for some integer α the class $\mathcal{H}$ contains neither a pure α-diamond, flawed α-diamond, nor a cycle of length α, then there exist constants λ and δ (depending on α) such that $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda,\delta}$.*

Proof. Suppose that there is such an integer α. Let $\lambda := 2\alpha$ and $\delta := 4\alpha^3 + 6\alpha^2$. Given any $H' \in \mathcal{H}$, we show how a transitively equivalent pattern $H \in \mathcal{C}_{\lambda,\delta}$ can be constructed, implying that H' belongs to $\mathcal{C}^*_{\lambda,\delta}$. A *vertex cover* of a graph is a subset X of its vertices such that every edge is incident to a vertex of X. By Lemma 20, H' cannot contain a matching of size α. It is well-known that if a graph has no matching of size α, then it has a vertex cover of size at most 2α (take the endpoints of any maximal matching). Let us fix a vertex cover X of H' having size at most 2α.

To obtain H from H', we start with a graph H on $V(H')$ having no edges and perform the following three steps.

1. Let us take the transitive closure on the vertex set X in H', i.e., let us introduce into H every edge uv with $u, v \in X$ such that there is a $u \to v$ path in H'.
2. Let us add all edges uv of H' to H for which $u \notin X$ or $v \notin X$.
3. Fixing an ordering of the edges introduced in step 2, we remove transitively redundant edges: following this order, we subsequently remove those edges uv for which there is a path from u to v in the remaining graph H that is not the edge uv itself.

It is clear that H is transitively equivalent to H'. Note that X is a vertex cover of H as well, and hence its complement $I = V(H) \setminus X$ is an *independent set*, i.e. no two vertices of I are adjacent. Let $E_I \subseteq E(H)$ be the set of edges between X and I. In the rest of the proof, we argue that the resulting pattern H belongs to $\mathcal{C}_{\lambda,\delta}$. We show that H can be decomposed into a path $P = (v_1, \ldots, v_{\lambda_0})$ in X, a star S_{v_i} centered at each v_i using the edges in E_I, and a small set of additional edges. This small set of additional edges is constructed in three steps, by considering a sequence of larger and larger sets $F_1 \subseteq F_2 \subseteq F_3$.

As E_I consists of edges between X and I, it can be partitioned into a set of stars with roots in X. The following claim shows that almost all of these edges are directed towards X or almost all of them are directed away from X.

▶ **Claim 23.** *Either there are less than $2\alpha^2$ edges uv in E_I with head in X, or less than $2\alpha^2$ edges uv in E_I with tail in X.*

Assume that the former case of Claim 23 is true, so that the number of edges in E_I with heads in X is bounded by $2\alpha^2$; the other case can be handled symmetrically. We will use the out-stars spanned by E_I for the caterpillar, which means that we obtain an out-caterpillar. We use the set F_1 to account for the edges in E_I with heads in X. Additionally, we will also introduce into F_1 those edges in E_I with tails in X that are adjacent to an edge of the former type. Formally, for any edge $uv \in E_i$ with $v \in X$, we introduce into F_1 every edge of E_I incident to u. After this step, F_1 contains less than $4\alpha^3$ edges, since there are less than $2\alpha^2$ edges $uv \in E_I$ with $v \in X$ and u can only be adjacent to vertices in X, which has size less than 2α.

For any vertex $v \in X$, let S_v denote the out-star formed by the edges of $E_I \setminus F$ incident to v. Let $X' \subseteq X$ contain those vertices $v \in X$ for which S_v has at least α leaves.

▶ **Claim 24.** *For any two distinct $u, v \in X'$, at least one of uv and vu is in H, and the stars S_u and S_v are vertex disjoint.*

We extend F_1 to F_2 by adding all edges of stars S_v with $v \in X \setminus X'$ to F_2. Since X contains less than 2α vertices and we extend F_1 only by stars with less than α edges, this step adds less than $2\alpha^2$ edges, i.e., $|F_2| \leq |F_1| + 2\alpha^2 = 4\alpha^3 + 2\alpha^2$.

By Claim 24, X' induces a *semi-complete* directed graph in H, i.e., at least one of the edges uv and vu exists for every pair $u, v \in X'$. It is well-known that every semi-complete directed graph contains a Hamiltonian path (e.g., [16, Chapter 10, Exercise 1]), and so there is a path $P = (v_1, \ldots, v_{\lambda_0})$ with $\lambda_0 = |X'| \leq 2\alpha = \lambda$ in H on the vertices of X'. We extend F_2 to F_3 by including any edge induced by vertices of X' that is not part of P. There are less than $4\alpha^2$ such edges, and hence we have $|F_3| \leq |F_2| + 4\alpha^2 \leq 4\alpha^3 + 6\alpha^2 = \delta$. The edges of H not in F_3 span the path P and disjoint out-stars S_{v_i} with $i \in \{1, \ldots, \lambda_0\}$, i.e., they form a λ_0-caterpillar. This proves that $H \in \mathcal{C}_{\lambda,\delta}$ and hence $H' \in \mathcal{C}^*_{\lambda,\delta}$, what we had to show. ◀

4.3 Reductions

Lemma 19 implies that in order to prove Theorem 16, we need W[1]-hardness proofs for the class of all directed cycles, the class of all pure in-diamonds, the class of all pure out-diamonds, etc. We provide these hardness proofs and then formally show that they imply Theorem 16.

Let us first consider the case when $\mathcal{H}$ is the class of all directed cycles. Recall that, given an arc-weighted directed graph G and a set $R \subseteq V(G)$ of terminals, the Strongly Connected Steiner Subgraph (SCSS) problem asks for a minimum-cost subgraph that is strongly connected and contains every terminal in R. This problem is known to be W[1]-hard parameterized by the number $k := |R|$ of terminals [21]. We can reduce SCSS to an instance of DSN where the pattern H is a directed cycle on R, which expresses the requirement that all the terminals are in the same strongly connected component of the solution. Thus the W[1]-hardness of SCSS immediately implies the W[1]-hardness of $\mathcal{H}$-DSN if $\mathcal{H}$ contains all directed cycles.

▶ **Lemma 25** (follows from [21]). *If $\mathcal{H}$ is the class of directed cycles, then $\mathcal{H}$-DSN is* W[1]*-hard parameterized by the number of terminals.*

Next we turn our attention to classes containing all diamonds. The following reductions are from the W[1]-hard Multicoloured Clique problem [19], in which an undirected graph together with a partition $\{V_1, \ldots, V_k\}$ of its vertices into k sets is given, such that for

any $i \in \{1, \ldots, k\}$ no two vertices of V_i are adjacent. The aim is to find a clique of size k, i.e. a set of pairwise adjacent vertices $\{w_1, \ldots, w_k\}$ with $w_i \in V_i$ for each $i \in \{1, \ldots, k\}$.

▶ **Lemma 26.** *If $\mathcal{H}$ is the class of all pure out-diamonds, then $\mathcal{H}$-DSN is* W[1]*-hard parameterized by the number of terminals. The same holds if $\mathcal{H}$ is the class of all pure in-diamonds.*

The reduction for the case when the pattern is a flawed α-diamond is essentially the same as the one for pure α-diamonds, as we show next.

▶ **Lemma 27.** *If $\mathcal{H}$ is the class of all flawed out-diamonds, then $\mathcal{H}$-DSN is* W[1]*-hard parameterized by the number of terminals. The same holds if $\mathcal{H}$ is the class of all flawed in-diamonds.*

Given the three reductions above, we can now prove Theorem 16, based on the additional reduction given in Lemma 17. We defer the final proof to the full version of this extended abstract.

References

1. Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. doi:10.1137/S0097539792236237.
2. MohammadHossein Bateni, Mohammad Taghi Hajiaghayi, and Dániel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5):21, 2011. doi:10.1145/2027216.2027219.
3. MohammadHossein Bateni and MohammadTaghi Hajiaghayi. Euclidean prize-collecting Steiner forest. *Algorithmica*, 62(3-4):906–929, 2012. doi:10.1007/s00453-011-9491-8.
4. MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Improved approximation algorithms for (budgeted) node-weighted Steiner problems. In *40th International Colloquium on Automata, Languages, and Programming*, pages 81–92, 2013. doi:10.1007/978-3-642-39206-1_8.
5. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 67–74, 2007.
6. Hans L. Bodlaender. Some classes of graphs with bounded treewidth. *Bulletin of the EATCS*, 36:116–125, 1988.
7. Glencora Borradaile, Philip N. Klein, and Claire Mathieu. An $O(n \log n)$ approximation scheme for Steiner tree in planar graphs. *ACM Transactions on Algorithms*, 5(3), 2009. doi:10.1145/1541885.1541892.
8. Glencora Borradaile, Philip N. Klein, and Claire Mathieu. A polynomial-time approximation scheme for Euclidean Steiner forest. *ACM Transactions on Algorithms*, 11(3):19:1–19:20, 2015. doi:10.1145/2629654.
9. Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6, 2013. doi:10.1145/2432622.2432628.
10. Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed Steiner problems. *J. Algorithms*, 33(1):73–91, 1999. doi:10.1006/jagm.1999.1042.
11. Chandra Chekuri, Guy Even, Anupam Gupta, and Danny Segev. Set connectivity problems in undirected graphs and the directed steiner network problem. *ACM Transactions on Algorithms*, 7(2):18, 2011. doi:10.1145/1921659.1921664.

12 Chandra Chekuri, Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R. Salavatipour. Approximation algorithms for node-weighted buy-at-bulk network design. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1265–1274, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283519.

13 Rajesh Hemant Chitnis, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Rohit Khandekar, Guy Kortsarz, and Saeed Seddighin. A tight algorithm for strongly connected Steiner subgraph on two terminals with demands (extended abstract). In *9th International Symposium on Parameterized and Exact Computation*, pages 159–171, 2014. doi:10.1007/978-3-319-13524-3_14.

14 Rajesh Hemant Chitnis, MohammadTaghi Hajiaghayi, and Dániel Marx. Tight bounds for planar strongly connected Steiner subgraph with fixed number of terminals (and extensions). In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1782–1801, 2014. doi:10.1137/1.9781611973402.129.

15 Erik D. Demaine, Mohammad Taghi Hajiaghayi, and Philip N. Klein. Node-weighted Steiner tree and group Steiner tree in planar graphs. *ACM Transactions on Algorithms*, 10(3):13:1–13:20, 2014. doi:10.1145/2601070.

16 Reinhard Diestel. *Graph theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, third edition, 2005.

17 S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/net.3230010302.

18 Jon Feldman and Matthias Ruhl. The directed Steiner network problem is tractable for a constant number of terminals. *SIAM J. Comput.*, 36(2):543–561, 2006. doi:10.1137/S0097539704441241.

19 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009. doi:10.1016/j.tcs.2008.09.065.

20 Martin Grohe and Dániel Marx. On tree width, bramble size, and expansion. *J. Comb. Theory, Ser. B*, 99(1):218–228, 2009. doi:10.1016/j.jctb.2008.06.004.

21 Jiong Guo, Rolf Niedermeier, and Ondrej Suchý. Parameterized complexity of arc-weighted directed Steiner problems. *SIAM J. Discrete Math.*, 25(2):583–599, 2011. doi:10.1137/100794560.

22 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Plenum, 1972.

23 Philip N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *J. Algorithms*, 19(1):104–115, 1995. doi:10.1006/jagm.1995.1029.

24 Sridhar Rajagopalan and Vijay V. Vazirani. On the bidirected cut relaxation for the metric Steiner tree problem. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 742–751, 1999. URL: http://dl.acm.org/citation.cfm?id=314500.314909.

25 Ondřej Suchý. On directed steiner trees with multiple roots. To appear in WG 2016. arXiv:1604.05103.

26 Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discrete Math.*, 19(1):122–134, 2005. doi:10.1137/S0895480101393155.

27 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed Steiner tree problem. *Algorithmica*, 18(1):99–110, 1997. doi:10.1007/BF02523690.

Double-Exponential and Triple-Exponential Bounds for Choosability Problems Parameterized by Treewidth*

Dániel Marx[1] and Valia Mitsou[2]

1 **Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary**
dmarx@cs.bme.hu

2 **Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary**
vmitsou@sztaki.hu

Abstract

Choosability, introduced by Erdős, Rubin, and Taylor [*Congr. Number.* 1979], is a well-studied concept in graph theory: we say that a graph is c-choosable if for any assignment of a list of c colors to each vertex, there is a proper coloring where each vertex uses a color from its list. We study the complexity of deciding choosability on graphs of bounded treewidth. It follows from earlier work that 3-choosability can be decided in time $2^{2^{O(w)}} \cdot n^{O(1)}$ on graphs of treewidth w. We complement this result by a matching lower bound giving evidence that double-exponential dependence on treewidth may be necessary for the problem: we show that an algorithm with running time $2^{2^{o(w)}} \cdot n^{O(1)}$ would violate the Exponential-Time Hypothesis (ETH). We consider also the optimization problem where the task is to delete the minimum number of vertices to make the graph 4-choosable, and demonstrate that dependence on treewidth becomes triple-exponential for this problem: it can be solved in time $2^{2^{2^{O(w)}}} \cdot n^{O(1)}$ on graphs of treewidth w, but an algorithm with running time $2^{2^{2^{o(w)}}} \cdot n^{O(1)}$ would violate ETH.

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

Keywords and phrases Parameterized Complexity, List coloring, Treewidth, Lower bounds under ETH

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.28

1 Introduction

Most NP-hard algorithmic problems become significantly easier when restricted to bounded-treewidth graphs. There are notable exceptions that remain NP-hard on graphs of constant treewidth (e.g., Steiner Forest [1, 20], List Edge Coloring [37], Edge-disjoint Paths [41]), but most of the natural combinatorial problems can be solved in polynomial time (or even in linear time) if treewidth is bounded. Courcelle's Theorem [11] is a meta-result showing that if a problem can be expressed in the language of monadic second order logic (MSOL), then it can be solved in linear-time on bounded-treewidth graphs: there is an algorithm with running time $f(w) \cdot n$, where w is the treewidth of the graph. While this result

* This work was supported by ERC Starting Grant PARAMTIGHT (No. 280152) and OTKA grant NK105645.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 28; pp. 28:1–28:15

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

immediately gives algorithms for a large number of combinatorial problems, focus has shifted in recent years towards trying to obtain a tighter understanding of how the running time has to depend on treewidth, that is, understanding what the best possible $f(w)$ in the running time can be. On the one hand, recent algorithm advances, such as *Cut & Count* [14] and fast subset convolution [3, 47], resulted in improved dependence on treewidth for various problems. On the other hand, conditional results based on the Exponential-Time Hypothesis (ETH) [25, 26, 34] give lower bounds on the best possible dependence that can be achieved, and in many cases these lower bounds tightly match the known algorithms [13, 14, 33, 35, 43]. (ETH can be informally stated as n-variable 3SAT cannot be solved in time $2^{o(n)}$.) Most of these tight bounds are of the form $2^{O(w)}$ (e.g., 3-Coloring, Hamiltonian Cycle, Triangle Packing, etc.), but there are surprising exceptions where the best possible dependence on treewidth is $2^{O(w \log w)}$ [14, 35] or even $2^{O(w^c)}$ for some constant $c > 1$ [13]. A result of Frick and Grohe [19] shows that, assuming $\mathsf{P} \neq \mathsf{NP}$, the dependence on treewidth can be really bad for some problems: Courcelle's Theorem does not remain true if we impose any *elementary* bound on the function $f(w)$. That is, for every $h \geq 1$, there are MSOL sentences and corresponding model-checking problems for which the dependence on treewidth is an exponential tower of height h. Pan and Vardi [42] gave strong evidence that this bad performance is expected for problems high up in the polynomial hierarchy: they showed that, under ETH, there is a strict hierarchy of lower bounds of the form $2^{2^{\cdots^{2^{o(w)}}}}$ inside PSPACE. In this paper, we present two fairly standard graph-theoretic problems that require double-exponential and triple-exponential dependence on treewidth, respectively.

1.1 k-Choosability

A proper k-coloring of a graph G is a mapping $f : V(G) \to [k]$ such that $f(u) \neq f(v)$ for any two adjacent $u, v \in V(G)$. List coloring is the generalization where instead of having the same set $[k]$ of colors available at each vertex, each vertex v has its own list $L(v)$ of available colors and the question is whether there is a proper coloring f that assigns a color $f(v) \in L(v)$ to each vertex v. The algorithmic aspects of list coloring have received significant interest [2, 10, 23, 24, 29, 32, 37, 38, 46].

Erdős, Rubin, and Taylor [16] defined a combinatorial property related to list colorings: we say that a graph G is k-choosable if it has a coloring for any list assignment L that has size k at each vertex. Clearly, k-choosability implies k-colorability. One may feel that the converse implication could also be true: after all, it may seem safe to guess that the "worst case" of list coloring is when every vertex has the same list $[k]$. But this intuition is wrong: for example, for any $k \geq 1$, there are 2-colorable graphs that are not k-choosable.

From the computational complexity point of view, deciding k-choosability is a much harder algorithmic problem than deciding k-colorability. Deciding k-choosability does not seem to belong to the class NP (a witness for k-choosability would need to prove colorability for *every* list assignment L) or to the class coNP (an uncolorable list assignment would be a good witness for non-k-choosability, but it cannot be verified in polynomial time). In fact, the k-choosability problem is known to lie higher in the polynomial hierarchy.

▶ **Theorem 1** (from [21]). *k-Choosability for $k \geq 3$ is Π_2^p-complete.*

We observe a similar gap in complexity between the two problems when looking at algorithms parameterized by treewidth. 3-colorability can be decided in time $3^w \cdot n^{O(1)}$ using standard techniques [12]. Fellows et al. [17] showed that deciding 3-choosability is fixed-parameter tractable parameterized by treewidth. One can make this result quantitative

by observing that the running time is actually $2^{2^{O(w)}} \cdot n^{O(1)}$ for graphs of treewidth w. Our first result shows that this double exponential dependence on treewidth is best possible, assuming ETH.

▶ **Theorem 2.** *For every fixed $k \geq 3$:*

1. *There is an algorithm for k-Choosability with running time $2^{2^{O(w)}} \cdot n^{O(1)}$ on graphs of treewidth w.*
2. *Assuming ETH, there is no algorithm for k-Choosability with running time $2^{2^{o(w)}} \cdot n^{O(1)}$ on graphs of treewidth w.*

1.2 k-Choosability Deletion

Given any class $\mathcal{C}$ of graphs, one can define various graph modification problems where the task is to transform a given graph G into a member of $\mathcal{C}$ with the minimum number of vertex deletions/edge deletions/edge additions. The parameterized algorithms literature is especially rich in this type of problems, where the goal is, for example, to make the graph acyclic [9, 15], bipartite [27, 36, 44, 45], planar [28, 40], chordal [5, 7, 18, 30, 39], or interval [4, 6, 8]. Investigating these problems on graphs of bounded treewidth is an interesting question on its own right, but additional motivation comes from the fact that some of the algorithms on *general graphs* first reduce the treewidth and then invoke an algorithm exploiting bounded treewidth [28, 39, 40].

If we look at the vertex deletion versions of coloring and choosability problems, then we can observe an even larger gap than in the decision version. For technical reasons, we give a proof only for $k \geq 4$ colors. It is not difficult to show (we leave it as an exercise to the reader) that there is an algorithm with running time $2^{O(w)} \cdot n^{O(1)}$ for 4-Coloring Deletion that, given a graph G of treewidth w, computes the minimum number of vertices that needs to be deleted to make the graph 4-colorable. On the other hand, if we consider the 4-Choosability Deletion problem, which asks for the minimum number of vertices that need to be deleted to make a given graph G 4-choosable, then we need *triple-exponential* dependence on treewidth.

▶ **Theorem 3.** *For every fixed $k \geq 4$:*

1. *There is an algorithm for k-Choosability Deletion with running time $2^{2^{2^{O(w)}}} \cdot n^{O(1)}$ on graphs of treewidth w.*
2. *Assuming ETH, there is no algorithm for k-Choosability Deletion with running time $2^{2^{2^{o(w)}}} \cdot n^{O(1)}$ on graphs of treewidth w.*

As a side result, we show that k-Choosability Deletion lies one level higher than k-Choosability in the polynomial hierarchy (compare this with the fact that k-Coloring and k-Coloring Deletion are both in NP).

▶ **Theorem 4.** *For every $k \geq 3$, k-Choosability Deletion is Σ_3^p-complete.*

The reader may feel that the Π_2^p- and Σ_3^p-completeness of these problems already give sufficient explanation why double- or triple-exponential dependence on treewidth is needed. This is true in some sense: the quantifier alternations in the problem definitions are the common underlying reasons for being in the higher levels of the polynomial hierarchy and for requiring unusually large dependence on treewidth. But let us point out that these two types of complexity results require very different proof structures. In Π_2^p- or Σ_3^p-completeness proofs, we start with a canonical Π_2^p- or Σ_3^p-complete quantified satisfiability problem and we use the alternations inherent in the definitions of k-Choosability or k-Choosability

Deletion to express the alternations in quantified satisfiability. On the other hand, in the proofs of Theorems 2(2) and 3(2), we start with problems in NP and use the alternations inherent in k-Choosability or k-Choosability Deletion for *compression*: we want to express the original instance by a graph having treewidth only $O(\log n)$ or $O(\log \log n)$. Thus the main theme of our proofs is trading alternation for compression: we want to use alternation to allow the succinct encoding and verification of information. Note also that both k-Choosability and k-Choosability Deletion can be solved in time $2^{n^{O(1)}}$, hence the exponential explosion appears only in the context of bounded-treewidth graphs.

2 Preliminaries

▶ **Definition 5** (List coloring)**.** Given graph G together with sets $\mathcal{L}(v) \subset \mathbb{N}$, one for every vertex v (we shall call $\mathcal{L}(v)$ the *list* of v), then G is *$\mathcal{L}$-colorable* if there exists a coloring $c : V(G) \to \mathbb{N}$, which is proper and for which $\forall v \in V(G), c(u) \in \mathcal{L}(v)$. In that case, c is called a *proper $\mathcal{L}$-coloring*.

▶ **Definition 6** (Choice number)**.** Given G and some $k \in \mathbb{N}$, G is called *k-choosable* if for any list assignment $\mathcal{L} : V(G) \to 2^{\mathbb{N}}$ with $|\mathcal{L}(v)| = k$ for all $v \in V(G)$, G is $\mathcal{L}$-colorable. The *choice number* (or *list-chromatic number*) of G, denoted by $\chi_\ell(G)$, is the minimum number k such that G is k-choosable.

The notion of f-choosability defined below generalizes k-choosability, but for arbitrary list sizes for each vertex.

▶ **Definition 7** (f-choosable graphs)**.** A graph G is called *f-choosable* for some function $f : V(G) \to \mathbb{N}$ (called *list-capacity function*) if G is $\mathcal{L}$-colorable for any list assignment $\mathcal{L} : V(G) \to 2^{\mathbb{N}}$ where $|\mathcal{L}(v)| = f(v)$ for all $v \in V(G)$.

As a direct consequence of the definition, we get that the null graph K_0 with no vertices is f-choosable for any list-capacity function f. The reason is the vacuous truth that, for any list assignment $\mathcal{L}$, the empty function $e : \emptyset \to \mathbb{N}$ is a proper $\mathcal{L}$-coloring of K_0.

▶ **Definition 8** (Color Compatibility)**.** Given a graph G, a list assignment $\mathcal{L}$, an ordered k-tuple $(u_1, u_2, \ldots, u_k) \in V(G)^k$ and an ordered k-tuple of colors $(c_1, c_2, \ldots, c_k)$ with $c_i \in \mathcal{L}(u_i)$ for $i \in [k]$, we say that $(c_1, c_2, \ldots, c_k)$ is *compatible* on $(u_1, u_2, \ldots, u_k)$ if there exists a proper $\mathcal{L}$-coloring g of G with $g(u_i) = c_i$. We may omit G and $\mathcal{L}$ if they are clear from the context.

We define the following algorithmic problems:

- (i, j)-Choosability: Given a graph G, integers $i, j \in \mathbb{N}$ where $i \leq j$, and a list-capacity function f for which $f(v) \in \{i, \ldots, j\}$ for any vertex $v \in V$, decide whether G is *f-choosable*.
- k-Choosability:= (k, k)-Choosability.
- (i, j)-Choosability Deletion: Given a graph G, integers $i, j, r \in \mathbb{N}$ with $i \leq j$, and a list-capacity function f for which $f(v) \in \{i, \ldots, j\}$ for any vertex $v \in V$, decide whether there exists $U \subset V$ with $|U| \leq r$ such that $G - U$ is *f-choosable*.
- k-Choosability Deletion:= (k, k)-Choosability Deletion.

Below are some known results about Choosability.

▶ **Theorem 9** (from [16])**.** $(2, 3)$-Choosability *is Π_2^p-complete.*

▶ **Theorem 10** (from [21])**.** k-Choosability *for $k \geq 3$ is Π_2^p-complete.*

▶ **Theorem 11** (from [17])**.** k-Choosability *parameterized by the treewidth of the input graph is in* FPT.

3 Double-exponential lower bound for k-Choosability

The topic of this section is proving Theorem 2(2), the double-exponential lower bound on 3-Choosability parameterized by treewidth. For the proof, we will find it convenient to start the reduction from Edge 3-Coloring, where given a graph G, the task is to decide if G has a proper edge 3-coloring (that is, whether there is an assignment $c : E(G) \to \{1, 2, 3\}$ such that $c(e_1) \neq c(e_2)$ for any two edges e_1 and e_2 sharing an endpoint). Holyer's proof [22] for the NP-hardness of Edge 3-Coloring on 3-regular graphs can be observed to give a tight lower bound under ETH. Note that a 3-regular graph on n vertices has exactly $3n/2 = O(n)$ edges. We state the lower bound in a slightly awkward way, but this type of bound is what we exactly need.

▶ **Theorem 12** (follows from [22]). *Assuming ETH,* Edge 3-Coloring *cannot be decided in time* $2^{2^{o(\log n)}}$*, where* n *is the number of vertices of the graph. Moreover, this remains true even if we consider only 3-regular graphs whose number of vertices is an integer power of* 2.

That is, if we reduce Edge 3-Coloring to some other problem by creating an equivalent instance with treewidth $O(\log n)$, then an algorithm of the target problem with running time $2^{2^{o(w)}} \cdot n^{O(1)}$ for graphs of treewidth w would yield an algorithm with running time $2^{2^{o(\log n)}} \cdot n^{O(1)}$ for Edge 3-Coloring, contradicting Theorem 12.

The reason why reducing from this problem is convenient for our proof is that Edge 3-Coloring involves constraints of the form "the three edges e_1, e_2, e_3 incident to a vertex u use all three colors from $\{1, 2, 3\}$," and this type of constraints will be easy to express in our reduction. However, there is an unfortunate presentation issue: when talking about colors, vertices, and edges, it may not be immediately clear if we refer to the source Edge 3-Coloring instance or to the target 3-Choosability instance, and this may cause confusion. Therefore, we prefer to treat Edge 3-Coloring as a constraint satisfaction problem and use terminology appropriate to that. Formally, an instance of Edge 3-Coloring can be interpreted as a problem where we have a set X of n variables (corresponding to the edges of G), each variable has the domain $\{1, 2, 3\}$, and we have a set Y of m constraints (corresponding to the vertices of G). Each constraint contains exactly three distinct variables, and the constraint is satisfied if these variables are assigned three different values. Then the proper edge 3-colorings of G are in one-to-one correspondence with the satisfying assignments of this constraint satisfaction problem. Note that each variable appears in exactly two constraints.

Given an instance H of the constraint satisfaction problem described above, we create an instance of $(2, 3)$-Choosability, i.e, a graph G and a list-capacity function $f : V \to \{2, 3\}$. Then we show that there is a satisfying assignment of I if and only if G is *not* $(2, 3)$-choosable. We will further make sure that $\mathrm{tw}(G) = O(\log m)$, which will imply the desired time lower bound. For the forward direction, all we need to do is, given a satisfying assignment $h : X \to \{1, 2, 3\}$ of H, use h in order to define a list assignment $\mathcal{L}$ for which G is not $\mathcal{L}$-colorable. The converse direction is somewhat more involved, as we need to start from the assumption that G is not f-choosable and, given some uncolorable list assignment $\mathcal{L}'$, show that H is satisfiable. However, since we do not know exactly how the list assignment $\mathcal{L}'$ which fails to proper $\mathcal{L}'$-color G looks like, we need to consider more general properties which apply to any such potential list assignment.

3.1 Gadgets

Before presenting the main reduction, we introduce three different types of gadgets and prove their properties. Corresponding to the two directions of the proof of correctness of the

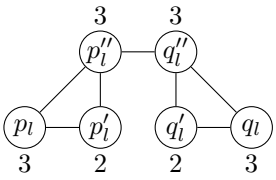


Figure 1 The *Bit-chooser* gadget B_l of the l^{th} bit.

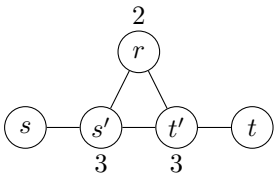


Figure 2 The *Weak-edge* gadget W_{st}, connecting vertices s, t.

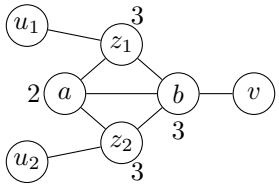


Figure 3 The *Weak-star* gadget S_i for variable x_i.

reduction, we show two types of properties for each gadget: an "$\exists$ list $\mathcal{L}$" property, which we use in order to define a list assignment $\mathcal{L}$ for the forward direction, and a "$\forall$ lists $\mathcal{L}$" property, which shall be useful in proving the converse direction.

1. Bit-chooser gadget B_l (Figure 1). Informally, a list assignment of the bit chooser gadget can enforce that a certain color appears on at least one of the two outputs p_l and q_l, but this is all it can do: in every list assignment, there is a "good" color on p_l and q_l that is compatible with every color on the other output.

▶ **$\exists\mathcal{L}$-Property 1.** There exists a list assignment $\mathcal{L}$ and a color c with $c \in \mathcal{L}(p_l) \cap \mathcal{L}(q_l)$ such that for any proper $\mathcal{L}$-coloring of B_l, at least one of p_l, q_l should receive color c.

▶ **$\forall\mathcal{L}$-Property 1.** For every list assignment $\mathcal{L}$, there exists a color $c \in \mathcal{L}(p_l)$ (resp., $\mathcal{L}(q_l)$) such that for every color $c' \in \mathcal{L}(q_l)$ (resp., $c' \in \mathcal{L}(p_l)$) the pair (c, c') is compatible on (p_l, q_l) (resp., on (q_l, p_l)).

2. Weak-edge gadget W_{st} (Figure 2). Informally, the Weak-edge gadget can prevent a certain combination (c, c') of colors appearing on the two outputs, but it cannot prevent more than one such combinations.

▶ **$\exists\mathcal{L}$-Property 2.** There exists a list assignment $\mathcal{L}$ and colors $c \in \mathcal{L}(s)$, $c' \in \mathcal{L}(t)$ such that the pair (c, c') is incompatible on (s, t).

▶ **$\forall\mathcal{L}$-Property 2.** For every list assignment $\mathcal{L}$, there exists at most one $(c, c') \in \mathcal{L}(s) \times \mathcal{L}(t)$ such that (c, c') incompatible on (s, t). In fact, if (c, c') is incompatible on (s, t), then the following hold: $c, c' \notin \mathcal{L}(r)$; $\mathcal{L}(s') = \{c\} \cup \mathcal{L}(r)$; and $\mathcal{L}(t') = \{c'\} \cup \mathcal{L}(r)$.

Because of their properties, the Weak-edges, if given an appropriate list assignment $\mathcal{L}$, can define an incompatible pair of colors being mutually assigned to their endpoints. Another way to view a Weak-edge is to consider it a directed edge between a precolored vertex s and an uncolored vertex t potentially forbidding a particular color from being assigned to t: if, for W_{st}, (c, c') is incompatible on (s, t) and s has already been assigned color c, then we know that t cannot receive color c'. Observe that, because of the $\forall\mathcal{L}$-Property, each Weak-edge defines at most one such incompatible pair between the endpoints.

3. Weak-star gadget S_i (Figure 3). Informally, the Weak-star gadget has the property that there is at most one color c on v that can forbid one specific color c_1 on u_1 and one specific color c_2 on u_2. When using this gadget on a vertex v whose list size is 2, it will act like an OR-gadget: if this specific color appears on u_1 or u_2, then it forbids color c on v, and hence forces it to take the other color in the list of v.

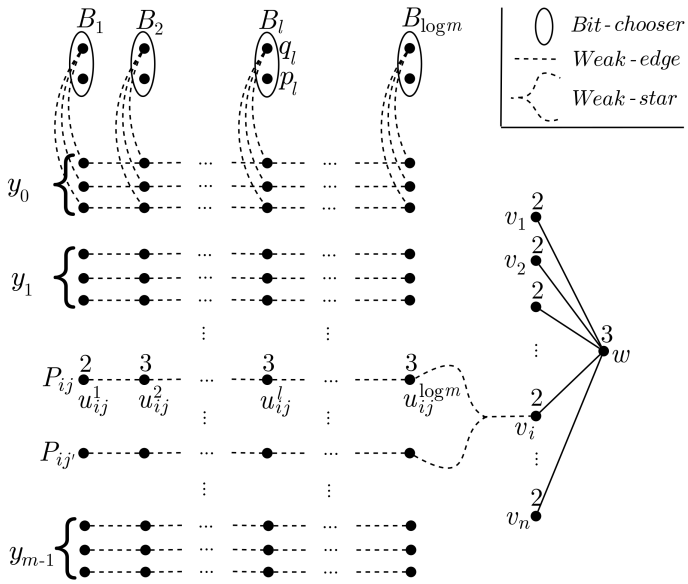


Figure 4 An overview of the construction.

▶ **$\exists\mathcal{L}$-Property 3.** There exist a list assignment $\mathcal{L}$ and a color $c \in \mathcal{L}(u_1) \cap \mathcal{L}(u_2) \cap \mathcal{L}(v)$ such that (c, c) is incompatible on both (u_1, v) and (u_2, v).

▶ **$\forall\mathcal{L}$-Property 3.** For every list assignment $\mathcal{L}$, there exist colors $c_1 \in \mathcal{L}(u_1), c_2 \in \mathcal{L}(u_2), c \in \mathcal{L}(v)$ such that if (d_1, d_2, d) is incompatible on (u_1, u_2, v) then $d = c$ and $(d_1 = c_1) \vee (d_2 = c_2)$.

3.2 Construction

We will now proceed with the description of the construction of G. Consider an arbitrary ordering of the m constraints of Y, assigning a unique index $\{0, \ldots, m-1\}$ to each of them. Now construct $\log m$ *Bit-chooser* gadgets $B_1, \ldots, B_{\log m}$ as shown in Figure 1. For gadget B_l, vertex p_l shall correspond to bit 1 and vertex q_l to 0.

Furthermore, for each appearance of a variable x_i in constraint y_j we construct a *Chain* P_{ij}, which is essentially a path on $\log m$ vertices $u_{ij}^1, \ldots, u_{ij}^{\log m}$, where we have substituted every edge (u_{ij}^l, u_{ij}^{l+1}) by a *Weak-edge* gadget $W_{u_{ij}^l u_{ij}^{l+1}}$ as the one shown in Figure 2, by identifying u_{ij}^l with s and u_{ij}^{l+1} with t. We also set $f(u_{ij}^1) = 2$ and $f(u_{ij}^l) = 3$ for $l > 1$.

Then we need to connect the Chains to the Bit-choosers. To do so, for each Chain P_{ij}, we write j in binary representation and for $l = 1, \ldots, \log m$, if the l^{th} bit is 1, we connect u_{ij}^l to p_l, else we connect it to q_l. The connection is by a Weak-edge $W_{p_l u_{ij}^l}$ or $W_{q_l u_{ij}^l}$ respectively.

In addition, we construct n variable-vertices $v_1, \ldots, v_n$, one for each variable $x_1, \ldots, x_n$ and we set $f(v_i) = 2$. Eventually, we need to connect the two Chains which correspond to the two appearances of x_i to vertex v_i. In order to do so, we use a *Weak-star* gadget S_i (see Figure 3). For Chains P_{ij} and $P_{ij'}$ with $j < j'$, corresponding to variable x_i in constraints y_j and $y_{j'}$, respectively, we identify $u_{ij}^{\log m}$ with u_1, $u_{ij'}^{\log m}$ with u_2, and v with variable-vertex v_i.

Last, we construct a *checker* vertex w with $f(w) = 3$ and connect it to all $v_1, \ldots, v_n$ (using an ordinary edge). This completes the construction. Our claim is that H is satisfiable if and only if G is not f-choosable.

It is easy to verify that the constructed graph G has pathwidth (and hence treewidth) bounded by $O(\log m)$.

▶ **Lemma 13.** *$pw(G) = O(\log m)$.*

3.3 Satisfying assignment ⇒ uncolorable list assignment

In this section, we prove the forward direction of the correctness of the reduction.

▶ **Lemma 14.** *If H is satisfiable, then G is not f-choosable.*

Proof. Assume that H is satisfiable. Then there should be an assignment $h : X \to \{1, 2, 3\}$ satisfying all the constraints in Y. We will produce a list assignment $\mathcal{L}$ for which the graph will not be $\mathcal{L}$-colorable.

First, we construct the lists of the v_i's according to the satisfying assignment of H, $\mathcal{L}(v_i) = \{h(x_i), c\}$. For the checker vertex w, we have $\mathcal{L}(w) = \{1, 2, 3\}$. For the main vertices of the Chains P_{ij}, we set $\mathcal{L}(u^1_{ij}) = \{c, c'\}$ and $\mathcal{L}(u^l_{ij}) = \{c, c', c''\}$, for $l \in \{2, \ldots, \log m\}$. Last, for the gadget vertices, we shall construct their lists according to those from the proofs of the $\exists\mathcal{L}$-Properties, as described below.

- For vertices in the Bit-choosers, lists match exactly those in the proof of $\exists\mathcal{L}$-Property 1.
- For vertices in the Weak-stars, lists match exactly those in the proof of $\exists\mathcal{L}$-Property 3.
- The Weak-edges should specify appropriate incompatible pairs of colors on their endpoints, and the list assignment should follow that of proof of $\exists\mathcal{L}$-Property 2: Weak-edges connecting Bit-choosers to Chains should forbid (c, c') on (p_l, u^l_{ij}) (resp., (q_l, u^l_{ij})); Weak-edges interconnecting Chain-vertices should forbid (c, c'') on $(u^{(l-1)}_{ij}, u^l_{ij})$.

Let us now explain why G is not $\mathcal{L}$-colorable, no matter which colors we pick from the lists. We are going to show that any possible coloring of the Bit-choosers accordant with $\mathcal{L}$ corresponds to selecting a constraint y, by selecting a 0-1 value for $\log m$ bits, which together select a constraint index in $\{0, \ldots, m-1\}$.

From $\exists\mathcal{L}$-Property 1 and for the particular list assignment which emerges from its proof, we are required to *select* (at least) one of the two endpoints p_l, q_l by giving it a special color c. Selecting p_l is interpreted as setting the l^{th} bit to 1, whereas selecting q_l is interpreted as setting it to 0. A selection of $\log m$ bit-vertices (one from each Bit-chooser) corresponds to picking a binary number, which shall represent the index of the constraint we select.

Suppose that a constraint $y_j = (x_{j_1}, x_{j_2}, x_{j_3})$ is selected this way. This means that for the 3 Chains $P_{j_1 j}, P_{j_2 j}, P_{j_3 j}$ we have color c' being removed from all $\mathcal{L}(u^l_{j_i j})$. In particular, this means that $u^1_{j_i j}$ has the unique color c available for it, which in turn forces color c to $u^2_{j_i j}$ by forbidding c'', and so on. This way, color c will propagate throughout the Chains, forcing color c on $u^{\log m}_{ij}$.

From $\exists\mathcal{L}$-Property 3 and for the list described in its proof, forcing color c on $u^{\log m}_{ij}$ forbids color c on v_i, forcing us to choose color $h(x_i)$, which in turn should forbid $h(x_i)$ from w. Since we have three variable-vertices $v_{j_1}, v_{j_2}, v_{j_3}$ with forced choices and $x_{j_1}, x_{j_2}, x_{j_3}$ belong to the same constraint y_j which h satisfies, $h(x_{j_1}), h(x_{j_2}), h(x_{j_3})$ should all be distinct values from $\{1, 2, 3\}$. Thus, all three colors should be forbidden in w, which is fatal since $\mathcal{L}(w) = \{1, 2, 3\}$. ◀

3.4 Uncolorable list assignment ⇒ satisfying assignment

Let us now proceed with the converse direction. First we determine some properties that every uncolorable list assignment $\mathcal{L}'$ of G needs to have. Then we extract an assignment

from $\mathcal{L}'$ for H and use the properties of the lists to show that it is a satisfying assignment.

The following definition will be convenient in the proofs. Let B' be the vertices of all the $\log m$ Bit-choosers. Given some list assignment $\mathcal{L}$, we say that a partial $\mathcal{L}$-coloring $g : B' \to \mathbb{N}$ *activates* vertex u^l_{ij} if $g(p_l) = c$ for some color $c \in \mathcal{L}(p_l)$ and there is a Weak-edge $W_{p_l u^l_{ij}}$ forbidding (c, c') on (p_l, u^l_{ij}) (or similarly for q_l). Thus, if we were to extend g to a proper $\mathcal{L}$-coloring of the whole graph, we would have one less choice for u^l_{ij}. In this case, vertex u^l_{ij} is called *active*. A Chain P_{ij} is called *active* if all its vertices are active. The crucial observation is that if a Chain P_{ij} is not active, no matter what color $c \in \mathcal{L}'(u^{\log m}_{ij})$ we assign to $u^{\log m}_{ij}$, this can be extended to a proper $\mathcal{L}'$-partial coloring of the chain P_{ij}. Suppose for example that vertex u^l_{ij} is not activated, that is, the coloring on the Bit-chooser B_l does not forbid any color on u^l_{ij}. Then we can start coloring the vertices u^1_{ij}, u^2_{ij}, …, u^{l-1}_{ij} in this order, then the vertices $u^{\log m}_{ij}$, $u^{\log m-1}_{ij}$, …, u^{l+1}_{ij} in this order, and there is still at least one available color left for u^l_{ij}. Furthermore, even if P_{ij} is active, it is possible to extend the partial coloring of the Bit-choosers to it, but this may force a certain color on $u^{\log m}_{ij}$.

▶ **Lemma 15.** *If G is not $\mathcal{L}'$-colorable, then any partial $\mathcal{L}'$-coloring of the Bit-choosers activates at least 3 of the Chains.*

Proof. Consider a partial $\mathcal{L}'$-coloring of the Bit-choosers. Suppose that only two Chains $P_{i_1 j_1}$, $P_{i_2 j_2}$ are activated, potentially forcing a coloring on $u^{\log m}_{i_1 j_1}$ and $u^{\log m}_{i_2 j_2}$. By $\forall\mathcal{L}$-Property 3, the $\mathcal{L}'$-coloring can be extended through the Weak-stars S_{i_1} and S_{i_2} even if $i_1 = i_2$, potentially reducing $|\mathcal{L}'(v_{i_1})|$ and $|\mathcal{L}'(v_{i_2})|$ to a unique choice. For every other vertex $u^{\log m}_{ij}$ vertex, any color can be extended to its Chain. Hence the Weak-star S_i for $i \notin \{i_1, i_2\}$ does not forbid any color on v_i.

Now observe that, even if both v_{i_1}, v_{i_2} have forced colors c_1, c_2, there is always a third color $c \in \mathcal{L}'(w)$, $c_1 \neq c \neq c_2$, which we can assign to w. Further observe that, since all other v_i have two available compatible choices with the rest of G, there will be at least one color in every v_i's list which should be different than c. Use these colors to complete a proper $\mathcal{L}'$-coloring for G. Of course this is a contradiction. Thus there should be exactly 3 active Chains. ◀

▶ **Lemma 16.** *For every list assignment $\mathcal{L}'$, there is some partial $\mathcal{L}'$-coloring of the Bit-choosers that activates exactly 3 Chains. Furthermore, for every constraint there exists a partial $\mathcal{L}'$-coloring that activates its 3 corresponding Chains.*

Proof. From $\forall\mathcal{L}$-Property 1, we know that there exists a color $c_p \in \mathcal{L}'(p_l)$ which is compatible with all colors in $\mathcal{L}'(q_l)$ and a color $c_q \in \mathcal{L}'(q_l)$ which is compatible with all colors in $\mathcal{L}'(p_l)$. Let c'_p be an arbitrary color of $\mathcal{L}'(p_l)$ different from c_p, and let c'_q be an arbitrary color of $\mathcal{L}'(q_l)$ different from c_q. Note that both (c_p, c'_q) and (c'_p, c_q) can be extended to the Bit-chooser B_l and we obtain two different colorings for B_l this way.

Let us consider all possible colorings of the Bit-choosers that arise from selecting one of these two colorings for each B_l; there are $2^{\log m} = m$ combinations. We claim that each Chain P_{ij} is activated by at most one of these colorings. Suppose that there are two colorings that both activate P_{ij}. The two colorings must differ on at least one of the Bit-choosers, say, on B_l. Suppose that P_{ij} is connected to p_l with a Weak-edge (the case when it is connected to q_l is analogous). Color c_p appears on p_l in one of the colorings and color c'_p appears in the other coloring. As $c_p \neq c'_p$, it is not possible that the Weak-edge $W_{p_l u^l_{ij}}$ forces a color on u^l_{ij} in both cases, a contradiction.

Since each of the m colorings activates at least 3 Chains by Lemma 15 and we have shown that each of the $3m$ Chains is activated by at most one of the colorings, a counting argument shows that each coloring activates exactly 3 Chains, and each Chain is activated by exactly one of the m colorings. That is, the Chains can be partitioned into m triples, and each triple can be activated by one of the colorings. To prove the second statement of the lemma, we need to show that each such triple contains Chains corresponding to one of the constraints. Suppose for a contradiction that one of the m colorings activates two Chains P_{ij} and $P_{i'j'}$ with $j \neq j'$. As the numbers j and j' are different, there is a bit l where they differ, which means that, without loss of generality that P_{ij} is connected to p_l, while $P_{i'j'}$ is connected to q_l. Suppose that this coloring assigns (c_p, c'_q) to (p_l, q_l) (the case when (c'_p, c_q) appears is similar). Recall that c_p on p_l is compatible with any color of $\mathcal{L}'(q_l)$ appearing on q_l. Let c''_q be the third color of $\mathcal{L}'(q_l)$, different from c_q and c'_q. We may modify the coloring on B_i such that (c_p, c''_q) appears on Bit-chooser B_l. Then $P_{i'j'}$ will no longer be activated: if color c'_q on q_l activated $u^l_{i'j'}$ via Weak-edge $W_{q_l u^l_{i'j'}}$, then color c''_q surely does not activate it. We observe that this modified coloring cannot activate any Chain that was not active before. Indeed, if some Chain $P_{i^*j^*}$ becomes activated by color c''_q on q_l, then $P_{i^*j^*}$ was not activated by any of the m colorings we considered before, as those colorings assigned only colors c_q and c'_q to q_l. This contradicts our earlier claim that each P_{ij} is activated by exactly one of the m colorings. Thus the modified coloring satisfies strictly less than 3 of the Chains, which contradicts Lemma 15. Thus we can conclude that each of the m colorings activates a (different) triple of Chains corresponding to one the m constraints. ◀

▶ **Lemma 17.** *If G is not f-choosable, then H is satisfiable.*

Proof. We may assume without loss of generality that $\mathcal{L}'(w) = \{1, 2, 3\}$. For every Weak-star S_i, consider the color $c_i \in \mathcal{L}'(v_i)$ given by $\forall\mathcal{L}$-Property 3. We define the assignment $h(x_i) := c^*_i$, where $\{c^*_i\} = \mathcal{L}(v_i) \setminus \{c_i\}$. We show that this gives a satisfying assignment: for every constraint $y_j = (x_{i_1}, x_{i_2}, x_{i_3})$, the colors appearing on the variables x_{i_1}, x_{i_2}, x_{i_3} form the set $\{1, 2, 3\}$; in particular, this will imply $h(i) \in \{1, 2, 3\}$ for every i.

Let us verify that that constraint $y_j = (x_{i_1}, x_{i_2}, x_{i_3})$ is satisfied. By Lemma 16, there is a partial assignment to the Bit-choosers that activates exactly the Chains $P_{i_1 j}$, $P_{i_2 j}$, and $P_{i_3 j}$. This means that the vertices $u^{\log m}_{i_1 j}$, $u^{\log m}_{i_2 j}$, and $u^{\log m}_{i_3 j}$ are forced to some color, but the other vertices $u^{\log m}_{ij}$ are unaffected. Thus for $i \notin \{i_1, i_2, i_3\}$, the Weak-star S_i does not prevent any color on v_{i_t}. But for $t = 1, 2, 3$, the Weak-star S_t may forbid the use of color c_{i_t} on v_{i_t}. In order to avoid any conflicts, we assign color $h(x_{i_t}) \neq c_{i_t}$ to v_i. If $\{h(x_{i_1}), h(x_{i_2}), h(x_{i_3})\} \neq \{1, 2, 3\}$, then we can assign to w a color not appearing on v_{i_1}, v_{i_2}, v_{i_3}, and then extend the coloring to v_i for each $i \notin \{i_1, i_2, i_3\}$ by choosing a color different from the color of w. This would contradict the assumption that $\mathcal{L}'$ is not colorable. Thus $\{h(x_{i_1}), h(x_{i_2}), h(x_{i_3})\} = \{1, 2, 3\}$, that is, constraint y_j is satisfied. ◀

3.5 Lower Bounds

Now we are ready to establish the lower bounds stated in Theorem 2(2).

▶ **Theorem 18.** (2, 3)-Choosability *cannot be decided in time $2^{2^{o(pw)}} \cdot n^{O(1)}$, where pw is the pathwidth of the input graph, under ETH.*

Proof. Suppose we could decide (2,3)-Choosability in time $2^{2^{o(\mathrm{pw})}} \cdot n^{O(1)}$. We know from Lemma 13 that $\mathrm{pw} = O(\log m)$.

Thus $2^{2^{o(\mathrm{pw})}} \cdot n^{O(1)} = 2^{2^{o(\log m)}} \cdot n^{O(1)}$. From Lemmas 14 and 17, this would imply solving Edge 3-Coloring in time $2^{2^{o(\log m)}}$. This contradicts Theorem 12. ◀

▶ **Corollary 19** (Theorem 2(2)). *Assuming ETH, k-Choosability for any $k \geq 3$ cannot be decided in time $2^{2^{o(pw)}} \cdot n^{O(1)}$, where pw is the pathwidth of the input graph.*

Proof. First observe that the problem for $k \geq \mathrm{pw} + 1$ is meaningless, since the answer is trivially yes: a graph G of pathwidth pw is pw-degenerate, and thus $(\mathrm{pw}+1)$-choosable.

Gutner and Tarsi [21] give a reduction from $(2,3)$-Choosability to k-Choosability, where the pathwidth of the constructed graph G' is $O(\mathrm{pw}(G))$. ◀

4 Triple-exponential lower bound for k-Choosability Deletion

The goal of this section is to prove Theorem 3(2): the triple-exponential lower bound for k-Choosability Deletion. For this proof, we choose a variant of list coloring as the source problem of the reduction.

Our reduction is from Bipartite List 3-Coloring: given a bipartite graph H with vertex set $X = X_1 \cup X_2, X_1 = \{x_{10}, x_{11}, \ldots x_{1(n-1)}\}, X_2 = \{x_{20}, x_{21}, \ldots, x_{2(n-1)}\}$, edge set Y with $|Y| = m$, and a list assignment $\mathcal{D} : X \to 2^{\{1,2,3\}}$ with $|\mathcal{D}(x)| \geq 2$ for all $x \in X$, the task is to find a proper 3-coloring $\phi : V(G) \to \{1,2,3\}$ of G where $\phi(v) \in \mathcal{D}(v)$ for every vertex v. This problem is known to be NP-hard [31] (to ensure $|\mathcal{D}(x)| \geq 2$, one needs to observe that any vertex with $|\mathcal{D}(x)| = 1$ can be removed from the graph after omitting its unique color from the lists of its neighbors). The NP-hardness proof can be observed to give a tight lower bound, which we state in the following form.

▶ **Lemma 20.** *Assuming ETH, there is no algorithm for* Bipartite List 3-Coloring *with running time $2^{2^{2^{o(\log\log n)}}}$ on bipartite graphs with n vertices on each side. This remains true if we consider only graphs where $\log\log n$ is integer.*

Proof. Given a 3SAT formula with n_0-variables and m_0-clauses, the reduction of Kratochvíl [31] creates an equivalent instance $(G, \mathcal{D})$ of Bipartite List 3-Coloring with $n_1 = O(n_0+m_0)$ vertices and $m_1 = O(n_0+m_0)$ edges. Let n be the smallest integer not smaller than n_1 such that $\log\log n$ is integer. Observe that $n \leq n_1^2$ (as $\log\log n \leq \log\log n_1 + 1$ and hence $\log n \leq 2\log n_1$), hence $\log\log n = \log\log n_1^2 = O(\log\log n_1) = O(\log\log(n_0+m_0))$. Let us add dummy vertices to G until each side has exactly n vertices. Using the assumed algorithm with running time $2^{2^{2^{o(\log\log n)}}}$, we would be able to solve the Bipartite List 3-Coloring instance and hence the equivalent 3SAT instance in time $2^{2^{2^{o(\log\log(n_0+m_0))}}} = 2^{o(n_0+m_0)}$, contradicting ETH. ◀

Given an instance of Bipartite List 3-Coloring, we construct an equivalent instance of $(1,4)$-Choosability Deletion consisting of a graph G and a list-capacity function f. More precisely, there exists a proper $\mathcal{D}$-coloring $h : X \to \{1,2,3\}$ of H if and only if there exists a subset $U \subseteq V(G)$ of vertices with $|U| \leq 4n$ such that $G - U$ is f-choosable. Moreover, graph G has treewidth $O(\log\log n)$. Thus an algorithm for $(1,4)$-Choosability Deletion on graphs of treewidth w with running time $2^{2^{2^{o(w)}}} \cdot n^{O(1)}$ would give an algorithm for Bipartite List 3-Coloring with running time $2^{2^{2^{o(\log\log n)}}} \cdot n^{O(1)}$, contradicting Lemma 20.

Similarly to Section 3, we reformulate Bipartite List 3-Coloring as a constraint satisfaction problem to improve the presentation: the appearance of colors and lists of colors in both the source and target problems would be a source of confusion. We can view Bipartite List 3-Coloring as a constraint satisfaction problem, where the variable set is X and constraints $y = (y_1, y_2) \in Y \subset X_1 \times X_2$ have arity 2. We call an assignment $h : X \to \{1,2,3\}$ a *legal assignment* if we have $h(x) \in \mathcal{D}(x)$ for every $x \in X$. We say

that H is satisfiable if there exists a legal assignment $h : X \to \{1, 2, 3\}$ such that we have $h(y_1) \neq h(y_2)$ for every $y = (y_1, y_2) \in Y$.

Observe that Choosability Deletion has inherently three levels of quantifier alternations in its definition: $\exists$ (set of deleted vertices) $\forall$ (list assignments $\mathcal{L}$) $\exists$ (choice of colors consistent with $\mathcal{L}$). The main idea of the reduction is that we can redefine Bipartite List 3-Coloring using three levels of quantifier alternations. Furthermore, in order to achieve the triple-exponential lower bound, we need to perform an even tighter compression than the one we achieved in Section 3. Keeping those two things in mind we proceed with re-defining Bipartite List 3-Coloring as follows.

From the original definition of Bipartite List 3-Coloring, we have that H is satisfiable if there exists legal assignment $h : X_1 \cup X_2 \to \{1, 2, 3\}$, such that for all $x_{1i} \in X_1$, $x_{2j} \in X_2$ with $h(x_{1i}) \neq h(x_{2j})$, we have $(x_{1i}, x_{2j}) \notin Y$. The latter requirement $(x_{1i}, x_{2j}) \notin Y$ can be re-written as $\forall y = (x_{1i'}, x_{2j'}) \in Y$, either $i' \neq i$ or $j' \neq j$. For some $i < n$, let $\mathcal{B}(i) = [\mathcal{B}(i)_0, \mathcal{B}(i)_1, \ldots, \mathcal{B}(i)_{\log n - 1}]$ be the binary representation of i. Then $i \neq i'$ can be expressed as saying that there exists a $k \in \{0, \ldots, \log n - 1\}$ such that $\mathcal{B}(i)_k \neq \mathcal{B}(i')_k$.

Putting everything together, we have:

▶ **Definition 21** (CSP Bipartite List 3-Coloring defined with 3 levels of quantifier alternations)**.** Given a set $X = X_1 \cup X_2$ of variables with $X_\xi = \{x_{\xi 0}, x_{\xi 1}, \ldots, x_{\xi(n-1)}\}$ for $\xi \in \{1, 2\}$ and a set $Y \subseteq X_1 \times X_2$ of constraints, the task is to decide if

- $\exists$ legal assignment $h : X_\xi \to \{1, 2, 3\}$, such that
- $\forall x_{1i} \in X_1$, $\forall x_{2j} \in X_2$, $\forall y = (x_{1i'}, x_{2j'}) \in Y$ with $h(x_{1i}) = h(x_{2j})$, we have
- $\exists k \in \{0, \ldots, \log n - 1\}$ such that either $\mathcal{B}(i)_k \neq \mathcal{B}(i')_k$ or $\mathcal{B}(j)_k \neq \mathcal{B}(j')_k$.

The reduction closely follows this equivalent definition of Bipartite List 3-Coloring: we use the three levels of alternation in the definition of $(1, 4)$-Choosability Deletion to express these three levels of alternation. Note also that the last level of alternation, when quantifying over $k \in \{0, \ldots, \log n - 1\}$ can be described as the selection of $\log \log n$ bits. This choice will be expressed by the introduction of $\log \log n$ Bit-choosers, which will be the dominating factor in the treewidth of the constructed instance.

References

1. Mohammad Hossein Bateni, Mohammad Taghi Hajiaghayi, and Dániel Marx. Approximation schemes for Steiner forest on planar graphs and graphs of bounded treewidth. *J. ACM*, 58(5):21, 2011. doi:10.1145/2027216.2027219.
2. M. Biró, Mihály Hujter, and Zsolt Tuza. Precoloring extension. I. interval graphs. *Discrete Mathematics*, 100(1-3):267–279, 1992. doi:10.1016/0012-365X(92)90646-W.
3. Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In David S. Johnson and Uriel Feige, editors, *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 67–74. ACM, 2007. doi:10.1145/1250790.1250801.
4. Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michal Pilipczuk. Subexponential parameterized algorithm for interval completion. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1116–1131. SIAM, 2016. doi:10.1137/1.9781611974331.ch78.
5. Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Inf. Process. Lett.*, 58(4):171–176, 1996. doi:10.1016/0020-0190(96)00050-6.
6. Yixin Cao. Linear recognition of almost interval graphs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms,*

SODA 2016, Arlington, VA, USA, January 10-12, 2016, pages 1096–1115. SIAM, 2016. doi:10.1137/1.9781611974331.ch77.

7 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 214–225. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.214.

8 Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21:1–21:35, 2015. doi:10.1145/2629595.

9 Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008. doi:10.1016/j.jcss.2008.05.002.

10 Janka Chlebíková and Klaus Jansen. The d-precoloring problem for k-degenerate graphs. *Discrete Mathematics*, 307(16):2042–2052, 2007. doi:10.1016/j.disc.2005.12.049.

11 Bruno Courcelle. The monadic second-order logic of graphs. I. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.

12 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

13 Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. Hitting forbidden subgraphs in graphs of bounded treewidth. In Erzsébet Csuhaj-Varjú, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 189–200. Springer, 2014. doi:10.1007/978-3-662-44465-8_17.

14 Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Johan M. M. van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 150–159. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.23.

15 Frank K. H. A. Dehne, Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Kim Stevens. An $o(2^{O(k)}n^3$ FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.*, 41(3):479–492, 2007. doi:10.1007/s00224-007-1345-z.

16 Paul Erdős, Arthur L Rubin, and Herbert Taylor. Choosability in graphs. *Congr. Numer*, 26:125–157, 1979.

17 Michael R Fellows, Fedor V Fomin, Daniel Lokshtanov, Frances Rosamond, Saket Saurabh, Stefan Szeider, and Carsten Thomassen. On the complexity of some colorful problems parameterized by treewidth. *Information and Computation*, 209(2):143–153, 2011.

18 Fedor V. Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM J. Comput.*, 42(6):2197–2216, 2013. doi:10.1137/11085390X.

19 Markus Frick and Martin Grohe. The complexity of first-order and monadic second-order logic revisited. *Ann. Pure Appl. Logic*, 130(1-3):3–31, 2004. doi:10.1016/j.apal.2004.01.007.

20 Elisabeth Gassner. The Steiner forest problem revisited. *J. Discrete Algorithms*, 8(2):154–163, 2010. doi:10.1016/j.jda.2009.05.002.

21 Shai Gutner and Michael Tarsi. Some results on $(a : b)$-choosability. *Discrete Mathematics*, 309(8):2260–2270, 2009.

22 Ian Holyer. The NP-completeness of edge-coloring. *SIAM Journal on computing*, 10(4):718–720, 1981.

23 M. Hujter and Zs. Tuza. Precoloring extension. II. Graphs classes related to bipartite graphs. *Acta Math. Univ. Comenian. (N.S.)*, 62(1):1–11, 1993.

24 Mihály Hujter and Zsolt Tuza. Precoloring extension III: Classes of perfect graphs. *Combinatorics, Probability & Computing*, 5:35–56, 1996. doi:10.1017/S0963548300001826.

25 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62:367–375, 2001.

26 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

27 Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time FPT algorithms via network flow. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1749–1761. SIAM, 2014. doi:10.1137/1.9781611973402.127.

28 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1802–1811. SIAM, 2014. doi:10.1137/1.9781611973402.130.

29 Klaus Jansen and Petra Scheffler. Generalized coloring for tree-like graphs. *Discrete Applied Mathematics*, 75(2):135–155, 1997. doi:10.1016/S0166-218X(96)00085-6.

30 Haim Kaplan, Ron Shamir, and Robert Endre Tarjan. Tractability of parameterized completion problems on chordal, strongly chordal, and proper interval graphs. *SIAM J. Comput.*, 28(5):1906–1922, 1999. doi:10.1137/S0097539796303044.

31 J. Kratochvíl. Precoloring extension with fixed color bound. *Acta Math. Univ. Comenian. (N.S.)*, 62(2):139–153, 1993.

32 Jan Kratochvíl and Zsolt Tuza. Algorithmic complexity of list colorings. *Discrete Applied Mathematics*, 50(3):297–302, 1994. doi:10.1016/0166-218X(94)90150-3.

33 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Known algorithms on graphs on bounded treewidth are probably optimal. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 777–789. SIAM, 2011. doi:10.1137/1.9781611973082.61.

34 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the Exponential Time Hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96.

35 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 760–776. SIAM, 2011. doi:10.1137/1.9781611973082.60.

36 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms*, 11(2):15:1–15:31, 2014. doi:10.1145/2566616.

37 Dániel Marx. NP-completeness of list coloring and precoloring extension on the edges of planar graphs. *Journal of Graph Theory*, 49(4):313–324, 2005. doi:10.1002/jgt.20085.

38 Dániel Marx. Precoloring extension on unit interval graphs. *Discrete Applied Mathematics*, 154(6):995–1002, 2006. doi:10.1016/j.dam.2005.10.008.

39 Dániel Marx. Chordal deletion is fixed-parameter tractable. *Algorithmica*, 57(4):747–768, 2010. doi:10.1007/s00453-008-9233-8.

40 Dániel Marx and Ildikó Schlotter. Obtaining a planar graph by vertex deletion. *Algorithmica*, 62(3-4):807–822, 2012. doi:10.1007/s00453-010-9484-z.

41 Takao Nishizeki, Jens Vygen, and Xiao Zhou. The edge-disjoint paths problem is NP-complete for series-parallel graphs. *Discrete Applied Mathematics*, 115(1-3):177–186, 2001. doi:10.1016/S0166-218X(01)00223-2.

42 Guoqiang Pan and Moshe Y. Vardi. Fixed-parameter hierarchies inside PSPACE. In *LICS*, pages 27–36. IEEE Computer Society, 2006.

43 Michal Pilipczuk. Problems parameterized by treewidth tractable in single exponential time: A logical approach. In Filip Murlak and Piotr Sankowski, editors, *Mathematical Foundations of Computer Science 2011 – 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, volume 6907 of *Lecture Notes in Computer Science*, pages 520–531. Springer, 2011. doi:10.1007/978-3-642-22993-0_47.

44 M. S. Ramanujan and Saket Saurabh. Linear time parameterized algorithms via skew-symmetric multicuts. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1739–1748. SIAM, 2014. doi:10.1137/1.9781611973402.126.

45 Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32(4):299–301, 2004. doi:10.1016/j.orl.2003.10.009.

46 Zsolt Tuza. Graph colorings with local constraints – a survey. *Discussiones Mathematicae Graph Theory*, 17(2):161–228, 1997. doi:10.7151/dmgt.1049.

47 Johan M. M. van Rooij, Hans L. Bodlaender, and Peter Rossmanith. Dynamic programming on tree decompositions using generalised fast subset convolution. In Amos Fiat and Peter Sanders, editors, *Algorithms – ESA 2009, 17th Annual European Symposium, Copenhagen, Denmark, September 7-9, 2009. Proceedings*, volume 5757 of *Lecture Notes in Computer Science*, pages 566–577. Springer, 2009. doi:10.1007/978-3-642-04128-0_51.

Do Distributed Differentially-Private Protocols Require Oblivious Transfer?*

Vipul Goyal[1], Dakshita Khurana[2], Ilya Mironov[3], Omkant Pandey[4], and Amit Sahai[5]

1 **Microsoft Research India, Bangalore, India**
vipul@microsoft.com
2 **UCLA and Center for Encrypted Functionalities, Los Angeles, USA**
dakshita@cs.ucla.edu
3 **Google, USA**
mironov@google.com
4 **Drexel University, Philadelphia, USA**
omkant@drexel.edu
5 **UCLA and Center for Encrypted Functionalities, Los Angeles, USA**
sahai@cs.ucla.edu

Abstract

We study the cryptographic complexity of two-party differentially-private protocols for a large natural class of boolean functionalities. Information theoretically, McGregor et al. [FOCS 2010] and Goyal et al. [Crypto 2013] demonstrated several functionalities for which the maximal possible accuracy in the distributed setting is significantly lower than that in the client-server setting. Goyal et al. [Crypto 2013] further showed that "highly accurate" protocols in the distributed setting for any non-trivial functionality in fact imply the existence of one-way functions. However, it has remained an open problem to characterize the exact cryptographic complexity of this class. In particular, we know that semi-honest oblivious transfer helps obtain optimally accurate distributed differential privacy. But we do not know whether the reverse is true.

We study the following question: *Does the existence of optimally accurate distributed differentially private protocols for any class of functionalities imply the existence of oblivious transfer (or equivalently secure multi-party computation)?* We resolve this question in the affirmative for the class of boolean functionalities that contain an XOR embedded on adjacent inputs. We give a reduction from oblivious transfer to:

- *Any distributed optimally accurate ϵ-differentially private protocol with $\epsilon > 0$* computing a functionality with a boolean XOR embedded on adjacent inputs.
- *Any distributed non-optimally accurate ϵ-differentially private protocol with $\epsilon > 0$, for a constant range of non-optimal accuracies and constant range of values of ϵ,* computing a functionality with a boolean XOR embedded on adjacent inputs.

Enroute to proving these results, we demonstrate a connection between optimally-accurate two-party differentially-private protocols for functions with a boolean XOR embedded on adjacent inputs, and noisy channels, which were shown by Crépeau and Kilian [FOCS 1988] to be sufficient for oblivious transfer.

* Research of the second and fifth authors supported in part from a DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C-0205. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government. The work of the third author was done while at Microsoft Research. The full version of this paper is available on-line at http://eprint.iacr.org/2015/1090.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 29; pp. 29:1–29:15

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

1998 ACM Subject Classification E.3 Data Encryption

Keywords and phrases Oblivious Transfer, Distributed Differential Privacy, Noisy Channels, Weak Noisy Channels

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.29

1 Introduction

Differential privacy [7, 8, 11, 14] is a very well-studied and popular privacy notion of recent years[1]. It provides powerful input privacy guarantees to participants of a statistical query database. Informally a randomized function computed on a database is said to be differentially private, if the output distribution of the function evaluated on the database, induced by the presence of a particular record, is statistically close to the output distribution induced when the record is absent. Note that this can be trivially achieved by computing an output that is independent of the entries in the database. Therefore, to be useful, a non-trivial differentially private algorithm must compute outputs that satisfy some meaningful notion of accuracy.

Consider a confidential dataset owned by a trusted server. The server must release the output of some statistic evaluated on the dataset, to an untrusted client. Even in this setting, where privacy is a concern only at the server's end, there is an evident tradeoff between privacy and accuracy. In fact, for any given privacy parameter ϵ, there is a maximum possible accuracy (which we call the *optimal accuracy*) such that any algorithm with better than optimal accuracy will fail to remain differentially private. Such privacy-accuracy tradeoffs are reasonably well-understood in the client-server setting [7, 12, 15, 23]. There has also been a huge body of work in designing algorithms that achieve close to optimal accuracies for various functionalities and data mining tasks in the client-server setting.

The focus of this work is the distributed setting, where a database is jointly hosted by multiple mutually distrusting servers. This was first studied by Dwork et al. [10]. As an illustrative example, consider two hospitals which together wish to compute the correlation between the occurrence of smoking and lung cancer by taking into account their combined patient records. In this setting, we require the servers to engage in a protocol, at the end of which the privacy of each record of both the servers must be guaranteed without a significant loss in accuracy. Note that the privacy requirements must be met for both servers, *given their entire view of the protocol transcript*, not just the computed output; possibly necessitating an additional loss in accuracy (over and above the loss in the client-server setting).

The intuition that the distributed setting would necessitate a greater accuracy loss than the client-server setting has been proved to be correct in the information theoretic world for different classes of functions. Beimel, Nissim and Omri [1] showed accuracy limits for distributed differentially-private protocols for n parties each holding their own inputs. McGregor, Mironov, Pitassi, Reingold, Talwar and Vadhan [36] showed large accuracy gaps in the two-party setting for several natural functionalities with n-bit inputs. Goyal, Mironov, Pandey and Sahai [19] demonstrated a constant gap between the maximal achievable accuracies in the client-server and distributed settings for any non-trivial boolean functionality.

In the computational setting this gap vanishes, if a semi-honest protocol for oblivious transfer exists. In this case, both servers can use secure multi-party computation [18] to simulate the client-server differentially private function evaluation, thereby obtaining the

[1] See [9] for a survey of results.

optimally accurate output evaluated on the union of their databases. Although oblivious transfer is sufficient to close this gap, it is not clear whether this is a *necessary* assumption.

Indeed, there has been a separate line of work, starting with Haitner, Omri and Zarosim [21] demonstrating black-box separations between one-way functions and distributed differentially private algorithms with optimal accuracies, for two-party functionalities with long outputs. Khurana, Maji and Sahai [25] showed a black-box separation between public-key encryption and distributed differentially private algorithms with optimal accuracies for two-party boolean functionalities. These separations also extend to a range of non-optimal accuracies that are information theoretically impossible to achieve in the distributed setting. These results provide evidence that some "strong" cryptographic assumption is likely necessary for optimally accurate (or close to optimally accurate) distributed differentially private function evaluation. Despite this research, the following question has remained elusive:

> *"Does there exist any class of functionalities whose distributed differentially private evaluation with optimal accuracy – necessitates the existence of oblivious transfer?"*

We prove that any protocol computing the boolean XOR functionality in a distributed differentially private manner with optimal accuracy and overwhelming probability of agreement (on the output) between both parties, implies the existence of oblivious transfer. Our result also directly lends itself to *any* boolean functionality that contains an embedded XOR on two adjacent inputs. Roughly, a function f is said to contain an embedded XOR if and only if the ideal functionality for f can be used to compute the boolean XOR functionality in the semi-honest setting. We give a formal definition of what it means for a function to contain an embedded XOR in the technical sections of the paper.

Interestingly, in the setting of secure computation, the ideal XOR functionality is known to be trivial. This is because the output of this functionality combined with the input of any individual party completely reveals the input of the other party. Thus, parties can simply send each other their inputs – and this corresponds to a secure evaluation of the XOR functionality. However, an optimally accurate distributed differentially private (noisy) protocol for XOR is not trivial, in fact we show that it implies oblivious transfer. Our proof makes use of the fact that an ideal (non-noisy) XOR is fully informative about the input of the other party.

Finally, it is interesting to observe the "philosophical" differences between differential privacy and secure computation:

- In (computationally) differentially-private protocols, "privacy comes first." We would like to first ensure privacy of each individual input and then with this constraint, would like to compute an output which is as accurate as possible.
- In secure computation, "accuracy comes first." We would like to release an accurate output to the function we are computing – and with this constraint, would like to ensure privacy of the inputs to the extent possible. Here, we require the transcript to leak no information about the inputs beyond what can be deduced from the output itself.

Nevertheless, as already mentioned, general secure computation immediately helps achieve the same (optimal) level of accuracy in distributed differentially-private protocols as the best achievable accuracy in the client-server setting. By relying completely on oblivious transfer for secure computation [26], our results show that the reverse is true as well (at least for the differentially private evaluation of any two-party functionality with an embedded XOR).

1.1 Our Contribution

Before elaborating upon our results, we briefly summarize what is known so far about accuracy gaps in the distributed differentially private computation of boolean functionalities.

Alice and Bob with inputs x and y, respectively, wish to compute $f(x, y)$ in a differentially private manner in the distributed setting. An ϵ-differentially private protocol for some functionality f ensures that the probability of Alice's views conditioned on y and y' are $\lambda := e^{\epsilon}$ multiplicatively-close to each other, where y and y' represented as bit-strings differ only in one coordinate (i.e. they are adjacent inputs). A protocol between them is α-accurate if for any x and y, the output of the protocol agrees with $f(x, y)$, with probability at least α.

For boolean functionalities, the optimal accuracy (in the client-server model) is $\alpha^*_\epsilon := \frac{\lambda}{\lambda+1}$, where $\lambda = e^{\epsilon}$. Goyal et al. [19] showed that in the information theoretic setting, $f = \mathsf{AND}$ can only be computed ϵ-differentially privately up to accuracy $\alpha^{(\mathsf{AND})}_\epsilon := \frac{\lambda(\lambda^2+\lambda+2)}{(\lambda+1)^3}$. Similarly, for $f = \mathsf{XOR}$ the maximal achievable accuracy in the information theoretic setting is $\alpha^{(\mathsf{XOR})}_\epsilon := \frac{\lambda^2+1}{(\lambda+1)^2}$. Note that $\alpha^{(\mathsf{XOR})}_\epsilon < \alpha^{(\mathsf{AND})}_\epsilon < \alpha^*_\epsilon$, for any finite $\epsilon > 0$.

We say that a function f contains an embedded XOR if there exist inputs x_0, x_1, y_0, y_1 and outputs z_0, z_1 such that $f(x_a, y_b) = z_{\mathsf{XOR}(a,b)}$ for all $a, b \in \{0, 1\}$. Similarly, we can define an embedded AND (equivalently, an embedded OR). By observing that any boolean function f which is sensitive to both parties' inputs either contains an embedded XOR or AND *on adjacent inputs* [3], the maximal achievable accuracy becomes

$$\alpha^{(f)}_\epsilon := \begin{cases} \alpha^{(\mathsf{XOR})}_\epsilon, & \text{if } f \text{ contains an embedded } \mathsf{XOR} \text{ on adjacent inputs} \\ \alpha^{(\mathsf{AND})}_\epsilon, & \text{otherwise.} \end{cases} \tag{1}$$

Given a semi-honest secure protocol for oblivious transfer, the optimal accuracy α_ϵ is achievable for any boolean f. With respect to the necessity of cryptographic assumptions, Goyal et al. [19] showed that achieving any accuracy between α_ϵ and $\alpha^{(f)}_\epsilon$ for any function f in the distributed setting implies the existence of one-way functions. We strengthen their result to show that any two-party differentially private protocol that computes the XOR functionality in a differentially private manner with accuracy close to α_ϵ implies the existence of semi-honest secure oblivious transfer. Our result also extends to a weaker variant of differential privacy, namely *computational differential privacy* [37]. All our results hold for two-party functionalities where both parties obtain the same output with overwhelming probability. Our results can be summarized as follows (with k denoting the security parameter):

▶ **Informal Theorem 1.** *Semi-honest oblivious transfer reduces to any two-party ϵ DP protocol for XOR with accuracy $\rho(> 1/2)$ such that $\rho \geq \alpha_\epsilon = \frac{e^\epsilon}{1+e^\epsilon}$ (the optimal accuracy).*

▶ **Informal Theorem 2.** *Semi-honest OT reduces to any two-party ϵ_k computationally DP protocol for XOR with accuracy $\rho_k \geq \alpha_{\epsilon_k} = \frac{e^{\epsilon_k}}{1+e^{\epsilon_k}}$ (the optimal accuracy).*

▶ **Informal Theorem 3.** *A $(\rho_k, \frac{\lambda_k}{m_k} - 1, \frac{\lambda}{m_k} - 1)$ weak noisy channel [6, 43] reduces to any two-party ϵ_k computationally DP protocol for XOR with (possibly non-optimal) accuracy $\rho_k(> 1/2)$ where $\lambda_k = e^{\epsilon_k}$ and $m_k = \rho_k/(1 - \rho_k)$.*

We prove the first two theorems via a reduction from (standard) noisy channels, which are known to imply semi-honest OT [5]. The first theorem is just a restriction of the second to the information-theoretic setting. The first two can also be viewed as special cases of the third. Furthermore, for a range of non-optimal accuracies we also show a reduction to weak noisy channels [6, 43]. Invoking known reductions of OT to weak binary symmetric channels, we obtain that for a small range of values of ϵ_k and possibly non-optimal accuracies ρ_k such that $\alpha^{(f)}_{\epsilon_k} \geq \rho_k >> \alpha^*_{\epsilon_k}$, there exist constants $c_1, \left\{c_2 < \frac{e^{c_1}}{1+e^{c_1}}\right\}$ such that for all $\epsilon_k > c_1$ and $\rho_k > c_2$, any two-party ϵ_k-private ρ_k-accurate computational DP protocol for XOR (or more generally any functionality with an embedded XOR on adjacent inputs) implies OT.

1.2 Related Work

The tradeoff between privacy and accuracy is quite central in designing differentially private algorithms. As mentioned before, in the client-server setting (where a single trusted server owns the entire database), the work of Dinur and Nissim [7] first showed limitations for a wide class of private algorithms. These limitations were further explored in [12, 15, 23].

The work of Dwork et al. [8, 11] proposed generic techniques for differentially private function evaluation, based on adding noise as a function of the sensitivity of database queries. The optimality of such techniques was studied in various settings in [13, 22, 41]. Variants of these techniques were shown to be optimal for certain classes of queries by [17, 20], and were shown to be non-optimal for other classes by Brenner and Nissim [2].

As mentioned before, there has also been a significant amount of work characterizing the accuracy of two-party differentially-private protocols. McGregor et al. [36] first showed that information theoretically, a large accuracy loss is inherent to the distributed differentially private computation of functionalities such as the inner product and hamming distance over n-bit inputs. This was followed by the work of Goyal et al. [19] who showed large gaps in the client-server and two-party accuracies for the differentially-private computation of boolean functionalities. Finally, the works of Haitner et al. [21] and Khurana et al. [25] showed that it is impossible to use one-way functions or even key-agreement in a black-box way to bridge any of these accuracy gaps. Our work subsumes these results for the case of XOR.

There has also been a bulk of work on the complexity of two-party finite functionalities in the information theoretic setting [26, 3, 32, 27, 28, 29, 31, 33]. Chor and Kushilevitz [4] established that all Boolean functions either reduce to SFE or can be trivially simulated. In the computationally bounded setting, Maji et al. [34] give a complete characterization of deterministic two-party finite functionalities while a series of works [35, 30, 24] give an information-theoretic characterization of (randomized, fixed-role) two-party functionalities.

Note that all constant communication protocols for Boolean functionalities can be viewed as two-party ideal finite functionalities, and therefore characterized according to [35, 30, 24]. Yet, our characterization extends to any polynomial-round *protocols* for differentially private computation with optimal accuracy, of certain classes of Boolean functionalities. This requires extra techniques to account for the entire transcript of protocol execution, which may leak information over and above the output of the ideal functionality.

1.3 Technical Overview

We consider the simple setting of distributed differentially private evaluation of boolean functions. Alice and Bob, with inputs x and y respectively, execute a protocol to compute a Boolean function $f(x, y)$. The protocol must preserve privacy (according to the differential privacy guarantee) of the input of each party. From [19], we know that any non-trivial Boolean function must embed an AND or an XOR minor on adjacent inputs. In this work, we focus on the XOR functionality; and our proof directly extends to any functionality with an embedded XOR on adjacent inputs. We also consider protocols with perfect agreement, that is, where Alice and Bob always get the same output at the end of the protocol (which is equivalent to saying that the output is part of the transcript). However, our proof also extends to protocols where parties agree on the output with overwhelming probability.[2]

[2] Note that if we relax the requirement of output agreement, then there is a simple information theoretically secure protocol achieving optimal accuracy.

Our main idea will be to use any protocol that implements the XOR functionality to construct an ideal noisy channel. An ideal noisy channel with flip probability $p < 1/2$ is a functionality that takes input a bit X from the sender, samples an independent bernoulli random variable (the 'error') E, where $E \sim \mathsf{Ber}(p)$, computes $\tilde{X} = (X \oplus E)$ and outputs it to the receiver.

Consider an optimally accurate differentially private evaluation of the boolean XOR functionality, where both parties agree on the output with overwhelming probability (and can then publish the output). In this case, the output of the differentially private functionality can be interpreted as a "noisy" version of the correct output. In the optimally accurate setting, the probability that the output is correct is exactly $\alpha_\epsilon = \frac{e^\epsilon}{1+e^\epsilon}$. In other words, let Z denote the output of the protocol, then for all inputs X, Y; the output $Z = (X \oplus Y) \oplus E$, where E is a bernoulli random variable $E \sim \mathsf{Ber}(\frac{1}{1+e^\epsilon})$.

Our protocol to realize a noisy channel is simple: the sender (Alice) and receiver (Bob) sample independent random (private) input bits $X \xleftarrow{\$} \{0,1\}$ and $Y \xleftarrow{\$} \{0,1\}$. They invoke the differentially private protocol for XOR with inputs (X, Y) and obtain output Z, where $Z = (X \oplus Y) \oplus E$, and E is the error as defined above. The sender outputs X and the receiver outputs $Z \oplus Y (= X \oplus E)$. It is easy to see that this protocol *correctly* implements a noisy channel with noise $E \sim \mathsf{Ber}(\frac{1}{1+e^\epsilon})$. However observe that the underlying differentially private protocol for XOR may not be an ideal secure computation protocol for the noisy XOR functionality. In particular, the protocol transcript may leak extra information over the official output. Thus, it remains to prove that the above protocol implements a secure noisy channel – while only relying on the security guarantee of the differential privacy condition.

In the computational setting, the ideal noisy channel functionality can be realized by a protocol with the following security property [5]: roughly, no efficient distinguisher on the sender's end, or on the receiver's end respectively, should be able to distinguish the cases when the error E was 0 from when E was 1. More formally, let $\mathcal{D}_R$ denote a distinguisher that obtains the entire view of the receiver, and $\mathcal{D}_S$ denote a distinguisher that obtains the entire view of the sender at the end of the protocol. Then, for any non-uniform PPT distinguisher $\mathcal{D}_R$, the following security guarantee is required to hold: $\Pr[\mathcal{D}_R = 1|E = 0] - \Pr[\mathcal{D}_R = 1|E = 1] = \mathsf{negl}(k)$ over the randomness of the protocol. Symmetrically, at the sender's end, for any non-uniform PPT distinguisher $\mathcal{D}_S$, $\Pr[\mathcal{D}_S = 1|E = 0] - \Pr[\mathcal{D}_S = 1|E = 1] = \mathsf{negl}(k)$ over the randomness of the protocol. Here $\mathsf{negl}(\cdot)$ denotes some function that is asymptotically smaller than the inverse of any polynomial function, and k denotes the security parameter.

Now, the challenge is to prove that the protocol outlined above satisfies these security properties – that is, no efficient distinguisher on the sender side, or on the receiver side can distinguish the case when $E = 0$ from when $E = 1$. Here, we use the following properties of the optimally accurate differentially private XOR functionality.

- Because of optimal accuracy, the protocol output is correct with probability exactly $\frac{e^\epsilon}{1+e^\epsilon}$.
- The (ideal, non-noisy) XOR functionality is fully informative: its output along with any of the parties' inputs, can be used to correctly compute the input of the other party.

Since the protocol is optimally accurate, the protocol output is correct – that is, $Z = X \oplus Y$ with probability exactly $\alpha_\epsilon = \frac{e^\epsilon}{1+e^\epsilon}$. Moreover, by the fully-informative property of XOR, the correct output, together with the input of any party can be used to correctly compute the other party's input. In other words, for all X, Y, the noisy output Z of the differentially private protocol, together with the input Y, helps compute a guess for the other party's input that is correct with probability at least α_ϵ ($Z \oplus Y$ equals X with probability α_ϵ).

Note that if a party could guess the other party's input with probability any better than α_ϵ, this would directly violate differential privacy. Therefore, the output already allows

computing the best possible guess (upto differential privacy limits) for the other party's input. Informally, this means that any extra information about the error (say, leaked from the transcript) could be used to obtain a better guess of the other party's input and directly violate differential privacy. To prove security of our noisy channel, we must formalize these arguments. This is done in Section 3 and forms the core of our proof of security.

2 Preliminaries

Notation. Let $\pi := \langle A, B \rangle$ be a two-party protocol. Let $\mathsf{view}_\pi^P(x_A, x_B)$ be the random variable which, in a random execution of π with inputs (x_A, x_B) consists of $(x_P, R_P, \mathsf{trans})$, where R_P is the randomness used by party P and trans is the sequence of messages exchanged between the parties in the sampled execution. Let out_P be the function applied by party P on $\mathsf{view}_\pi^P(x_A, x_B)$ to obtain the output for P, $\mathsf{out}_P(\mathsf{view}_\pi^P(x_A, x_B))$. We say that the protocol is *symmetric* if both parties receive the same output, i.e., for every x, y: $\mathsf{out}_A(\mathsf{view}_\pi^A(x_A, x_B)) = \mathsf{out}_B(\mathsf{view}_\pi^B(x_A, x_B))$. This is called the *official* output of the protocol, denoted by $\mathsf{out}_\pi(x_A, x_B)$. For the rest of this paper, we consider only symmetric protocols, however we note that our results can be easily extended to protocols in which both parties agree on the output with overwhelming probability.

In the computational setting, we consider a family of protocols $\{\pi_k\}_{k\in\mathbb{N}}$, where k is the security parameter. Then, the view of party $P \in \{A, B\}$ is denoted by $\mathsf{view}_\pi^P(k, x_A, x_B)$.

2.1 Noisy Channels

Informally, a noisy channel takes as input a bit b and outputs bit $b' = b \oplus e$ where *error bit* $e \sim \mathsf{Ber}(1-\rho)$ is sampled independently, and $\oplus$ is the bitwise exclusive-or operation. Roughly, the security requirement is that the error e remains "semantically secure" for both the sender and the receiver. Somewhat counterintuitively, we consider the flip probability of a ρ-noisy channel to be $(1-\rho)$. This is done deliberately to match DP protocols.

A (ρ, α, β)-weak binary symmetric channel [6, 43] is a noisy channel where the error is no longer "semantically secure". In particular, a malicious sender or receiver obtains partial leakage on the error added by the channel, and (α, β) denote sender and receiver leakage respectively. We defer the formal definitions to the full version.

Any protocol implementing a noisy channel is sufficient to implement the semi-honest OT functionality. A reduction between these primitives was first given by Crépeau and Kilian [5]. Furthermore, [43] showed that any protocol implementing the weak binary symmetric channel for a certain range of parameters of $\rho, \alpha, \beta)$, is sufficient to implement OT. Although these reductions are information-theoretic, they also carry over to the computational setting. We use the following corollary from [43]:

▶ **Corollary 1.** *Let ρ, α, β be constants, and let $\bar{\epsilon} = \frac{(1-\rho)^2}{(1-\rho)^2+\rho^2}$. If at least one of the conditions $2\alpha+\beta+\bar{\epsilon} \le 0.12$, or $\beta+\bar{\epsilon} < \frac{(1-\alpha)^4}{44}$, or $88\alpha+44\bar{\epsilon} < (1-\beta)^2$, or $196\alpha+98\beta+\frac{49}{2} < (1-2\bar{\epsilon})^2$ holds, then there exists a protocol that uses a (ρ, α, β)-passive weak BSC and efficiently implements OT secure in the semi-honest model.*

2.2 Differential Privacy

We give the formal definition of a weak notion of computational differential privacy. Assuming dense sets, this is a strictly weaker definition than other (simulation-based) definitions of CDP [37]. Therefore, our reductions automatically extend to other definitions.

▶ **Definition 2** (ϵ-Indistinguishable-Computational Differential Privacy, [37])**.** We say that an ensemble $\{M_k\}_{k\in\mathbb{N}}$ of randomized functions $M_k : \{0,1\}^n \mapsto \mathcal{R}_k$ with finite range $\mathcal{R}_k$, provides ϵ_k-IND-CDP if there exists a negligible function $\mathsf{negl} : \mathbb{N} \mapsto \mathbb{R}$ such that for every non-uniform PPT algorithm ("distinguisher") D, every polynomial $p(\cdot)$, every sufficiently large $k \in \mathbb{N}$, every $(x, x') \in \{0,1\}^n \times \{0,1\}^n$ satisfying $|x - x'|_h = 1$, and every advice string z_k of size at most $p(k)$ it holds that: $\Pr[D_k(M_k(x)) = 1] \leq e^{\epsilon_k} \times \Pr[D_k(M_k(x')) = 1] + \mathsf{negl}(k)$, where we write $D_k(y)$ for $D(1^k, z_k, y)$ and the probability is over the mechanism M_k and D_k.

▶ **Definition 3** (Differential Privacy over a subset of transcripts)**.** We say that an ensemble $\{M_k\}_{k\in\mathbb{N}}$ of randomized functions $M_k : \{0,1\}^n \mapsto \mathcal{R}_k$ with finite range $\mathcal{R}_k$, provides ϵ_k-IND-CDP over some subset of executions $\mathbb{S}$ if there exists a negligible function $\mathsf{negl} : \mathbb{N} \mapsto \mathbb{R}$ such that for every non-uniform PPT algorithm ("distinguisher") D, every polynomial $p(\cdot)$, every sufficiently large $k \in \mathbb{N}$, every adjacent pair $(x, x') \in \{0,1\}^n \times \{0,1\}^n$, and every advice string z_k of size $\leq p(k)$ $\Pr[D_k(M_k(x)) = 1 \wedge (M_k(x) \in \mathbb{S}_k)] \leq e^{\epsilon_k} \times \Pr[D_k(M_k(x')) = 1 \wedge (M_k(x') \in \mathbb{S}_k)] + \mathsf{negl}(k)$ where we write $D_k(y)$ for $D(1^k, z_k, y)$ and the probability is taken over the randomness of mechanism M_k and distinguisher D_k.

▶ **Definition 4** (Two-Party Differential Privacy)**.** Let $\pi : \langle A, B\rangle$ be a protocol with inputs of A and B in $\{0,1\}^n$. Then π provides ϵ-DP if: (1) for every $x \in \{0,1\}^n$ the mechanism represented by the function $\mathsf{view}^A_\pi(x, \cdot)$ over the inputs $y \in \{0,1\}^n$ is ϵ-DP, and (2) for every $y \in \{0,1\}^n$ the mechanism represented by $\mathsf{view}^B_\pi(\cdot, y)$ over the inputs $x \in \{0,1\}^n$ is ϵ-DP.

In the two-party computational setting, ϵ_k-IND-CDP is defined analogously. Formally, let $\{\pi_k := \langle A, B\rangle(1^k)\}_{k\in\mathbb{N}}$ be an ensemble of interactive functions where the inputs of A and B are in $\{0,1\}^n$. We say that $\{\pi_k\}_{k\in\mathbb{N}}$ provides ϵ_k-IND-CDP if: (1) for every $x \in \{0,1\}^n$ $\{\mathsf{view}^A_\pi(k, x, \cdot)\}_k$ provides ϵ_k-IND-CDP over the inputs $y \in \{0,1\}^n$, and (2) for every $y \in \{0,1\}^n$ $\{\mathsf{view}^A_\pi(k, \cdot, y)\}_k$ provides ϵ_k-IND-CDP over the inputs $x \in \{0,1\}^n$.

▶ **Definition 5** (Accuracy in Differential Privacy [19])**.** The *accuracy* of a randomized Boolean mechanism $M : \{0,1\}^n \mapsto \{0,1\}$ with respect to a Boolean function $f : \{0,1\}^n \mapsto \{0,1\}$ is defined as: $\mathsf{Acc}_f(M) = \min_x\{\Pr[M(x) = f(x)]\}$, where the probability is taken over the randomness of M.

The accuracy of a *symmetric two-party protocol* $\pi := \langle A, B\rangle$ w.r.t. $f : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$ is the accuracy of the (Boolean) mechanism $\mathsf{out}_\pi : \{0,1\}^n \times \{0,1\}^n \mapsto \{0,1\}$; where out_π returns the official output. Accuracy in the computational setting is defined analogously.

- For every Boolean mechanism $M : \{0,1\}^n \mapsto \{0,1\}$ and Boolean function $f : \{0,1\}^n \mapsto \{0,1\}$, if M is ϵ-DP then: $\mathsf{Acc}_f(M) \leq \frac{\lambda}{1+\lambda}$ where $\lambda = e^\epsilon$.[3] We call the bound $\rho = \frac{\lambda}{1+\lambda}$, the *optimal* accuracy, achieved by setting $M(x) = f(x) \oplus e$ such that $\Pr[e = 0] = \frac{\lambda}{1+\lambda}$.
- If M satisfies ϵ-IND-CDP, then $\mathsf{Acc}_f(M) \leq \frac{\lambda}{1+\lambda} + \mathsf{negl}(k)$ for a negligible function $\mathsf{negl}(\cdot)$.
- If a symmetric protocol ensemble $\{\pi_k\}_{k\in\mathbb{N}}$ provides ϵ-IND-CDP for a constant $\epsilon > 0$, then the accuracy of this ensemble w.r.t. the XOR function is at most $\frac{\lambda+\mathsf{negl}(k)}{1+\lambda} = \rho + \mathsf{negl}'(k)$ for constant ϵ. The accuracy ρ can be achieved using secure two-party computation [37].

3 Noisy Channels Reduce to Optimal Two-Party IND-CDP

▶ **Theorem 6.** *If there exists a two-party ϵ_k-IND-CDP protocol with accuracy $\rho_k(> 1/2)$ such that $\rho_k \geq \frac{e^{\epsilon_k}}{1+e^{\epsilon_k}}$ with respect to the* exclusive-or *function for a constant $\epsilon_k > 0$, then there exists a protocol implementing the ρ_k-noisy-channel functionality.*

[3] Informally, if this is not the case, there exists a distinguisher such that the ratio between the probability that it guesses the input correctly versus incorrectly is greater than e^ϵ, thereby violating ϵ-DP.

Proof. Let $\{\pi_k\}_k$ where $\pi_k = \langle A, B\rangle(1^k)$ be an ensemble of ϵ_k-IND-CDP protocols for computing the XOR function with accuracy $\rho_k \geq \frac{\lambda}{1+\lambda}$ where $\lambda = e^{\epsilon_k}$, and $\epsilon_k > 0$ is a constant. Note that since the protocol is ϵ_k-IND-CDP and $\epsilon_k > 0$, we have that $\rho_k \leq \frac{\lambda}{1+\lambda} + \mathsf{negl}(k)$ for some negligible function $\mathsf{negl}(k)$. For the rest of the proof, we denote ϵ_k by ϵ, and ρ_k by ρ.

The following protocol ensemble $\{\pi_k := \langle S, R\rangle(1^k)\}_k$ implements a ρ-noisy-channel: S receives bit x as input, and R has no input. R samples a random bit y and the parties execute the ϵ-IND-CDP protocol $\langle A(x), B(y)\rangle(1^k)$ and obtain the (same) bit z as official output of this protocol. R outputs $\tilde{x} = z \oplus y$ and S outputs $\perp$. The correctness of this protocol follows directly from the accuracy of the ϵ-IND-CDP protocol. We now show that it satisfies sender-security.

Sender security. Assume to the contrary, that the protocol does not satisfy sender-security. That is, there exists a non-uniform PPT distinguisher $\mathcal{D}_R$, a fixed polynomial $q(\cdot)$, and infinitely many values k for which (there exists a polynomial-sized advice string z_k such that) $\mathsf{Adv}^R_\pi(k) \geq 1/q(k)$. Fix one such k from now on and let:

$$p_k = \Pr[\mathcal{D}_R(z_k, \mathsf{view}^R_{\pi_k}) = 1 | E = 0] - \Pr[\mathcal{D}_R(\mathsf{view}^R_{\pi_k}) = 1 | E = 1]. \tag{2}$$

Note that $\mathsf{Adv}^R_\pi(k) = |p_k|$. Without loss of generality, let $p_k > 0$ for this k, and therefore by assumption $p_k \geq 1/q(k)$. We abuse notation and write $\mathcal{D}_R = 1$ to denote the event that $\mathcal{D}_R(1^k, z_k, \mathsf{view}^R_{\pi'_k}) = 1$.[4] Since $p_k \neq 0$ we must have that $0 < \Pr[\mathcal{D}_R = 1] < 1$.

Let E be the random variable denoting the *error bit* for the ϵ-IND-CDP protocol. That is, for the ϵ-IND-CDP protocol, $E = \tilde{x} \oplus x$. Since we are in the computational setting, the accuracy of the protocol may be different for each input, denoted by: $\rho_{00}, \rho_{01}, \rho_{10}, \rho_{11}$. However, they must all be within a negligible distance from each other and therefore lie within the interval $[\rho - \mathsf{negl}(k), \rho + \mathsf{negl}(k)]$, where ρ denotes ρ_{00}. Since a correct output is equivalent to $E = 0$, and each input is selected with equal probability, $\Pr[E = 0]$ (which is equivalent to "average" accuracy) also lies in the same interval. We show that if p_k is noticeable then differential privacy is violated on the set of transcripts where $\mathcal{D}_R$ outputs 1.

▶ **Claim 7.** $\Pr[E = 0 \wedge \mathcal{D}_R = 1] > e^\epsilon \times \Pr[E = 1 \wedge \mathcal{D}_R = 1] + \frac{p_k}{2}$.

Proof. Let $\Pr[E = 0] = \rho^*$, and $\mu(k)$ be a negligible function so that $\rho^* = \frac{\lambda}{1+\lambda} + \mu(k) > 1/2$. Also, $\Pr[E = 0]/\Pr[E = 1]$ is equal to $\rho^*/(1-\rho^*) = \lambda + \mu'(k)$ for some negligible function μ'. Now, since $\Pr[\mathcal{D}_R = 1] \neq 0$, we can write (using Bayes' rule):

$$\begin{aligned}
&\Pr[E = 0 \wedge \mathcal{D}_R = 1] = \Pr[\mathcal{D}_R = 1 | E = 0] \times \Pr[E = 0] \\
&= \big(p_k + \Pr[\mathcal{D}_R = 1 | E = 1]\big) \times \Pr[E = 0] && \text{(By equation 2)} \\
&= \left(p_k + \frac{\Pr[E = 1 | \mathcal{D}_R = 1] \times \Pr[\mathcal{D}_R = 1]}{\Pr[E = 1]}\right) \times \Pr[E = 0] && \text{(Bayes' rule)} \\
&= p_k \cdot \Pr[E = 0] + \Pr[E = 1 \wedge \mathcal{D}_R = 1] \times \frac{\Pr[E = 0]}{\Pr[E = 1]}
\end{aligned}$$

Note that: $p_k \cdot \Pr[E = 0] = p_k \rho^* > p_k/2$, and $\frac{\Pr[E=0]}{\Pr[E=1]} = \frac{\rho^*}{1-\rho^*} = \lambda + \mu'(k) > \lambda$. Therefore, $\Pr[E = 0 \wedge \mathcal{D}_R = 1] > \frac{p_k}{2} + \lambda \cdot \Pr[E = 1 \wedge \mathcal{D}_R = 1]$. ◀

[4] Note that the input of the sender in sampling view $\mathsf{view}^R_{\pi'_k}$ is uniformly chosen by definition of sender-security; and further, since k has been fixed, letting $\mathcal{D}_R := \mathcal{D}_R(1^k, z_k, \mathsf{view}^R_{\pi'_k})$ is unambiguous and well defined. Note that now, $p_k = \Pr[\mathcal{D}_R = 1 | e = 0] - \Pr[\mathcal{D}_R = 1 | e = 1]$.

- Obtain inputs $M_k(x)$, S_k, D'_k
- If $S_k(M_k(x)) = 1$, $D_k(M_k(x)) = D'_k(M_k(x))$
- If $S_k(M_k(x)) \neq 1$, $D_k(M_k(x)) = 0$

Figure 1 Algorithm for ϵ-IND-CDP Distinguisher D_k.

We say that a subset $\mathbb{S}$ of transcripts is PPT-checkable, if there exists a probabilistic poly-time "checking" algorithm for deciding membership of a transcript in $\mathbb{S}$.

▶ **Claim 8.** *If* $\Pr[E = 0 \wedge \mathcal{D}_R = 1] > e^\epsilon \times \Pr[E = 1 \wedge \mathcal{D}_R = 1] + \frac{p_k}{2}$ *is such that* $\frac{p_k}{2}$ *is non-negligible over uniformly chosen sender input, then the protocol ensemble* $\{\pi_k\}_k$ *does not preserve* ϵ*-IND-CDP on the PPT-checkable subset of transcripts satisfying* $\mathcal{D}_R = 1$.

Proof. From $\Pr[E = 0 \wedge \mathcal{D}_R = 1] > e^\epsilon \Pr[E = 1 \wedge \mathcal{D}_R = 1] + \frac{p_k}{2}$ it follows that $\Pr[\tilde{x} = x \wedge \mathcal{D}_R = 1] > e^\epsilon \Pr[\tilde{x} \neq x \wedge \mathcal{D}_R = 1] + \frac{p_k}{2}$, over the randomness of x where $\tilde{x}$ denotes the output of the receiver. Since x is *uniformly chosen* in $\{0, 1\}$,

$$\Pr[\tilde{x} = 1 \wedge \mathcal{D}_R = 1 | x = 1] + \Pr[\tilde{x} = 0 \wedge \mathcal{D}_R = 1 | x = 0]$$
$$> e^\epsilon \left(\Pr[\tilde{x} = 0 \wedge \mathcal{D}_R = 1 | x = 1]\right) + e^\epsilon \left(\Pr[\tilde{x} = 1 \wedge \mathcal{D}_R = 1 | x = 0]\right) + \frac{p_k}{2}$$

Now, it is easy to observe that *either* of the following statements are true.

1. $\Pr[\tilde{x} = 1 \wedge \mathcal{D}_R = 1 | x = 1] > e^\epsilon \times \Pr[\tilde{x} = 1 \wedge \mathcal{D}_R = 1 | x = 0] + \frac{p_k}{4}$ OR,
2. $\Pr[\tilde{x} = 0 \wedge \mathcal{D}_R = 1 | x = 0] > e^\epsilon \times \Pr[\tilde{x} = 0 \wedge \mathcal{D}_R = 1 | x = 1] + \frac{p_k}{4}$

In either case, it is possible to claim the existence of a distinguisher. If p_k is noticeable and statement 1 holds, then there exists a distinguisher $D_k^{1\prime}$ with output equal to receiver output $\tilde{x}$, which violates IND-CDP over the PPT checkable subset corresponding to $\mathcal{D}_R = 1$. On the other hand, if p_k is noticeable and statement 2 is true, then there exists a distinguisher $D_k^{2\prime}$ with output equal to $1 - \tilde{x}$, which violates IND-CDP over the PPT checkable subset corresponding to $\mathcal{D}_R = 1$.

It follows from this claim that if p_k is noticeable, then the protocol ensemble $\{\pi_k\}_k$ does not preserve ϵ-IND-CDP on the PPT-checkable subset of transcripts on which $\mathcal{D}_R = 1$, because there exists distinguisher $D'_k \in \{D_k^{1\prime}, D_k^{2\prime}\}$ and a corresponding pair of inputs $(x^*, x^{*\prime}) \in (\{0, 1\} \times \{0, 1\})$ such that $\Pr[D'_k = 1 \wedge \mathcal{D}_R = 1 | x = x^*] > e^\epsilon \times \Pr[D'_k = 0 \wedge \mathcal{D}_R = 1 | x = x^{*\prime}] + \frac{p_k}{4}$. In other words, there exists a non-uniform distinguisher that violates ϵ-IND-CDP on this subset. This proves the claim. ◀

▶ **Claim 9.** *A two-party protocol ensemble that provides* ϵ*-IND-CDP over all executions also provides* ϵ*-IND-CDP over any PPT-checkable subset of executions.*

Proof. Assume to the contrary that there exists a two-party ϵ-IND-CDP protocol for which there is a non-uniform PPT distinguisher D'_k that violates ϵ-IND-CDP over some PPT-checkable subset of executions (denoted by $\mathbb{S}_k$). Let S_k denote the code of a PPT-checking algorithm that returns 1 if some execution $M_k(x) \in \mathbb{S}_k$, and 0 otherwise.

Then, we construct a non-uniform PPT distinguisher D_k (Figure 1) that accepts S_k, D'_k as advice z_k, and violates ϵ-IND-CDP for the protocol.

We know that for some polynomial $p(\cdot)$, some sufficiently large $k \in \mathbb{N}$, some $(x^*, x^{*\prime}) \in \{0, 1\} \times \{0, 1\}$, some advice string z'_k of size at most $p(k)$ and all functions $\mathsf{negl} : \mathbb{N} \mapsto \mathbb{R}$, it holds that: $\Pr[D'_k(M_k(x^*)) = 1 \wedge (M_k(x^*) \in \mathbb{S}_k)] > e^{\epsilon_k} \Pr[D'_k(M_k(x^{*\prime})) = 1 \wedge (M_k(x^{*\prime}) \in$

$\mathbb{S}_k)] + \mathsf{negl}(k)$, where the probability is taken over the randomness of mechanism M_k and distinguisher D'_k, and $D'_k(y)$ represents $D'(1^k, z_k, y)$. Then, by a simple manipulation:

$$\begin{aligned}
&\Pr[D_k(M_k(x^*)) = 1] = \Pr[D_k(M_k(x^*)) = 1 \wedge (M_k(x^*) \in \mathbb{S}_k)] \\
&> e^{\epsilon} \Pr[D'_k(M_k(x^{*\prime})) = 1 \wedge (M_k(x^{*\prime}) \in \mathbb{S}_k)] + \mathsf{negl}(k) \\
&= e^{\epsilon} \big(\Pr[D_k(M_k(x^{*\prime})) = 1 \wedge (M_k(x^{*\prime}) \in \mathbb{S}_k)] + \Pr[D_k(M_k(x^{*\prime})) = 1 \wedge (M_k(x^{*\prime}) \notin \mathbb{S}_k)] \big) \\
&+ \mathsf{negl}(k), = e^{\epsilon} \Pr[D_k(M_k(x^{*\prime})) = 1] + \mathsf{negl}(k).
\end{aligned}$$

Therefore, we have a non-uniform PPT distinguisher D_k such that for some polynomial $p(\cdot)$, some sufficiently large $k \in \mathbb{N}$, for the same $(x^*, x^{*\prime}) \in (\{0,1\} \times \{0,1\})$, some advice string z_k of size at most $p(k)$ and all functions $\mathsf{negl} : \mathbb{N} \mapsto \mathbb{R}$ it holds that $\Pr[D'_k(M_k(x^*)) = 1] > e^{\epsilon_k} \times \Pr[D'_k(M_k(x^{*\prime})) = 1] + \mathsf{negl}(k)$. This completes the proof of this claim. ◀

From these claims, it follows that if p_k is noticeable, then $\{\pi_k\}_k$ does not preserve ϵ-IND-CDP. This is a contradiction, therefore $p_k = \mathsf{negl}(k)$, and the noisy channel is sender-secure.

Receiver security. The output z of the ϵ-IND-CDP-protocol, obtained by both parties, is symmetric with respect to the input of each party. Moreover, since the inputs of both parties are chosen uniformly at random, the security of the receiver follows in a manner similar to sender security. This completes the proof of the theorem. ◀

Combining this with Crépeau-Kilian's reduction [5] of OT to noisy channels, we obtain:

▶ **Corollary 10.** *If there exists a two-party ϵ_k-IND-CDP protocol with accuracy ρ_k such that $\rho_k \geq \frac{e^{\epsilon_k}}{1+e^{\epsilon_k}}$ with respect to the* exclusive-or *function for a constant $\epsilon_k > 0$, then there exists an ensemble of protocols implementing the semi-honest oblivious-transfer functionality in the computational setting.*

4 Noisy Channels Reduce to Non-Optimal Two-Party IND-CDP

▶ **Theorem 11.** *If there exists a two-party ϵ_k-IND-CDP protocol with non-optimal accuracy $\rho^1_k \leq \frac{e^{\epsilon_k}}{1+e^{\epsilon_k}}$ with respect to the* exclusive-or *function for a constant $\epsilon_k > 0$, then there exists a protocol implementing the $(\rho^1_k, \frac{\lambda}{m} - 1, \frac{\lambda}{m} - 1)$-passive weak binary symmetric channel functionality where $\rho^1_k > 1/2$, $\lambda = e^{\epsilon_k}$ and $m = \frac{\rho^1_k}{1-\rho^1_k}$.*

The proof of this theorem follows in a similar manner as Theorem 6, and can be found in the full version of our paper. While our reduction to weak noisy channels holds for all parameters $\epsilon > 0$ and accuracies ρ^1_k, the range of parameters for which such channels give OT is small. The following corollary follows from Theorem 11, and Corollary 1.

▶ **Corollary 12.** *If there exists a two-party ϵ_k-IND-CDP protocol with non-optimal accuracy $\rho^1_k \leq \frac{e^{\epsilon_k}}{1+e^{\epsilon_k}}$ with respect to the* exclusive-or *function for a constant $\epsilon > 0$, then there exist constants $c_1, \left\{c_2 < \frac{e^{c_1}}{1+e^{c_1}}\right\}$, such that for all $\epsilon_k > c_1$ and $\rho^1_k > c_2$, there is a protocol implementing the semi-honest oblivious transfer functionality.*

5 Conclusion and Open Problems

5.1 Extension to Functionalities with an Embedded XOR

Recall that we say that a function f contains an embedded XOR on *adjacent* inputs if there exist adjacent inputs x_0, x_1, y_0, y_1 and outputs z_0, z_1 such that $f(x_1, y_b) = z_{\mathsf{XOR}(a,b)}$ for all

$a, b \in \{0, 1\}$. It is easy to observe that any finite functionality f with an embedded XOR, which can be computed with optimal accuracy restricted to its embedded XOR on adjacent inputs, can be used to obtain a differentially private optimally accurate XOR functionality over boolean inputs. Accuracy of XOR follows from the accuracy of the original functionality f, and privacy of XOR follows because differential privacy is a worst-case guarantee which must be maintained even when restricted to a single bit of the adjacent inputs. The resulting differentially private optimally accurate XOR protocol can then be used to obtain a secure noisy channel and therefore, perform oblivious transfer.

5.2 Open Problems

Characterizing All Functionalities. It remains an intriguing open problem to obtain a complete characterization of functionalities whose differentially private evaluation with optimal accuracy in a distributed setting, is cryptographically complete. It is interesting to obtain a complete characterization even for boolean functionalities, since the differentially private evaluation of any non-trivial functionality with optimal accuracy (such as the inner product and hamming distance functionalities considered by McGregor et al. [36]) implies the differentially private evaluation of a non-trivial boolean functionality with optimal accuracy.

Consider, the case of boolean AND. This functionality is interesting, because any non-trivial boolean functionality must contain embedded AND or XOR on adjacent inputs [3]. Therefore, for instance, showing that any (possibly polynomial round) protocol that gives a differentially private protocol for the boolean AND functionality with optimal accuracy, is cryptographically complete – would imply the completeness of an optimally accurate distributed differentially private protocol for any non-trivial boolean functionality. However, unlike XOR, the AND functionality is not completely informative about the other party's input. In case the input of a party is 0, even a non-noisy output of the ideal AND functionality conveys absolutely no information about the input of the other party. In case the input is 1, the output allows to exactly compute the other party's input. Therefore, if a party has input 0, the differentially-private output would be completely useless for this party, while there could be additional leakage from the transcript (allowed by differential privacy) that we do not know how to use. Such functionalities appear to have interesting connections to weak oblivious transfer, from which it is not completely known how to obtain oblivious transfer.

Characterizing non-optimal accuracies. From the work of McGregor et al. [36] and Goyal et al. [19] in the information theoretic setting, it is clear that for any privacy parameter ϵ, there is a constant gap in the maximal achievable accuracies of any ϵ differentially private protocol in the client-server and distributed settings. Goyal et al. [19] additionally showed that any hope of bridging this gap would imply the existence of one-way functions. The black box separation results of [21, 25] also hold for differentially private protocols with any accuracy in this range. Yet, it is unclear whether all protocols with accuracies in this range must imply the existence of oblivious transfer.

Our techniques for non-optimal accuracies give rise to weak noisy channels and weak versions of oblivious transfer, which for a constant range of parameters, do imply full-fledged oblivious transfer. Yet, there is a large gap between the upper and lower bounds for weak oblivious transfer amplification, and since our reductions go via noisy channels – this gap lends itself to our setting. We believe that this provides additional motivation to revive (and continue) research on the characterization of weak noisy channels.

References

1 Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In Wagner [42], pages 451–468. doi:10.1007/978-3-540-85174-5_25.

2 Hai Brenner and Kobbi Nissim. Impossibility of differentially private universally optimal mechanisms. In focs2010 [16], pages 71–80. doi:10.1109/FOCS.2010.13.

3 Benny Chor and Eyal Kushilevitz. A zero-one law for boolean privacy (extended abstract). In David S. Johnson, editor, *STOC*, pages 62–72. ACM, 1989. doi:10.1145/73007.73013.

4 Benny Chor and Eyal Kushilevitz. A zero-one law for Boolean privacy. *SIAM J. Discrete Math.*, 4(1):36–47, 1991. doi:10.1137/0404004.

5 Claude Crépeau and Joe Kilian. Weakening security assumptions and oblivious transfer (abstract). In Shafi Goldwasser, editor, *CRYPTO*, volume 403 of *Lecture Notes in Computer Science*, pages 2–7. Springer, 1988. doi:10.1007/0-387-34799-2_1.

6 Ivan Damgård, Joe Kilian, and Louis Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 1999. doi:10.1007/3-540-48910-X_5.

7 Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In Frank Neven, Catriel Beeri, and Tova Milo, editors, *PODS*, pages 202–210. ACM, 2003. doi:10.1145/773153.773173.

8 Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006. doi:10.1007/11787006_1.

9 Cynthia Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011. doi:10.1145/1866739.1866758.

10 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006. doi:10.1007/11761679_29.

11 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006. doi:10.1007/11681878_14.

12 Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of LP decoding. In David S. Johnson and Uriel Feige, editors, *STOC*, pages 85–94. ACM, 2007. doi:10.1145/1250790.1250804.

13 Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In Mitzenmacher [38], pages 381–390. doi:10.1145/1536414.1536467.

14 Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 528–544. Springer, 2004. doi:10.1007/978-3-540-28628-8_32.

15 Cynthia Dwork and Sergey Yekhanin. New efficient attacks on statistical disclosure control mechanisms. In Wagner [42], pages 469–480. doi:10.1007/978-3-540-85174-5_26.

16 *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010.

17 Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In Mitzenmacher [38], pages 351–360. doi:10.1145/1536414.1536464.

18 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *STOC*, pages 218–229. ACM, 1987. doi:10.1145/28395.28420.

19 Vipul Goyal, Ilya Mironov, Omkant Pandey, and Amit Sahai. Accuracy-privacy tradeoffs for two-party differentially private protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 298–315. Springer, 2013. doi:10.1007/978-3-642-40041-4_17.

20 Mangesh Gupte and Mukund Sundararajan. Universally optimal privacy mechanisms for minimax agents. In Jan Paredaens and Dirk Van Gucht, editors, *PODS*, pages 135–146. ACM, 2010. doi:10.1145/1807085.1807105.

21 Iftach Haitner, Eran Omri, and Hila Zarosim. Limits on the usefulness of random oracles. *Theory of Cryptography Conference (TCC, to appear)*, 2013.

22 Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In Schulman [40], pages 705–714. doi:10.1145/1806689.1806786.

23 Shiva Prasad Kasiviswanathan, Mark Rudelson, Adam Smith, and Jonathan Ullman. The price of privately releasing contingency tables and the spectra of random matrices with correlated rows. In Schulman [40], pages 775–784. doi:10.1145/1806689.1806795.

24 Dakshita Khurana, Daniel Kraschewski, Hemanta K. Maji, Manoj Prabhakaran, and Amit Sahai. All complete functionalities are reversible, 2015.

25 Dakshita Khurana, Hemanta K. Maji, and Amit Sahai. Black-box separations for differentially private protocols. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014 – 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 386–405. Springer, 2014. doi:10.1007/978-3-662-45608-8_21.

26 Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *STOC*, pages 20–31. ACM, 1988. doi:10.1145/62212.62215.

27 Joe Kilian. A general completeness theorem for two-party games. In *STOC*, pages 553–560. ACM, 1991. doi:10.1145/103418.103475.

28 Joe Kilian. More general completeness theorems for secure two-party computation. In F. Frances Yao and Eugene M. Luks, editors, *STOC*, pages 316–324. ACM, 2000. doi:10.1145/335305.335342.

29 Joe Kilian, Eyal Kushilevitz, Silvio Micali, and Rafail Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput.*, 29(4):1189–1208, 2000. doi:10.1137/S0097539797321742.

30 Daniel Kraschewski, Hemanta K. Maji, Manoj Prabhakaran, and Amit Sahai. A full characterization of completeness for two-party randomized function evaluation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014 – 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 659–676. Springer, 2014. doi:10.1007/978-3-642-55220-5_36.

31 Robin Künzler, Jörn Müller-Quade, and Dominik Raub. Secure computability of functions in the it setting with dishonest majority and applications to long-term security. In Reingold [39], pages 238–255. doi:10.1007/978-3-642-00457-5_15.

32 Eyal Kushilevitz. Privacy and communication complexity. In *FOCS*, pages 416–421. IEEE, 1989. doi:10.1109/SFCS.1989.63512.

33 Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Complexity of multi-party computation problems: The case of 2-party symmetric secure function evaluation. In Reingold [39], pages 256–273. doi:10.1007/978-3-642-00457-5_16.

34 Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. A zero-one law for cryptographic complexity with respect to computational uc security. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 595–612. Springer, 2010. doi:10.1007/978-3-642-14623-7_32.

35 Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. A unified characterization of completeness and triviality for secure function evaluation. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT*, volume 7668 of *Lecture Notes in Computer Science*, pages 40–59. Springer, 2012. doi:10.1007/978-3-642-34931-7_4.

36 Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil P. Vadhan. The limits of two-party differential privacy. In focs2010 [16], pages 81–90. doi:10.1109/FOCS.2010.14.

37 Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil P. Vadhan. Computational differential privacy. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 126–142. Springer, 2009. doi:10.1007/978-3-642-03356-8_8.

38 Michael Mitzenmacher, editor. *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*. ACM, 2009.

39 Omer Reingold, editor. *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*. Springer, 2009. doi:10.1007/978-3-642-00457-5.

40 Leonard J. Schulman, editor. *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. ACM, 2010.

41 Jonathan Ullman and Salil P. Vadhan. PCPs and the hardness of generating private synthetic data. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 400–416. Springer, 2011. doi:10.1007/978-3-642-19571-6_24.

42 David Wagner, editor. *Advances in Cryptology – CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*. Springer, 2008.

43 Jürg Wullschleger. Oblivious transfer from weak noisy channels. In *TCC*, pages 332–349, 2009.

Functional Commitment Schemes: From Polynomial Commitments to Pairing-Based Accumulators from Simple Assumptions*

Benoît Libert[†1], Somindu C. Ramanna[‡2], and Moti Yung[§3]

1 **ENS de Lyon, LIP Laboratory, Lyon, France**
benoit.libert@ens-lyon.fr

2 **ENS de Lyon, LIP Laboratory, Lyon, France**
somindu.ramanna@ens-lyon.fr

3 **Snapchat, Los Angeles, CA, USA; and Columbia University, New York, USA**
moti@cs.columbia.edu

Abstract

We formalize a cryptographic primitive called functional commitment (FC) which can be viewed as a generalization of vector commitments (VCs), polynomial commitments and many other special kinds of commitment schemes. A non-interactive functional commitment allows committing to a message in such a way that the committer has the flexibility of only revealing a function of the committed message during the opening phase. We provide constructions for the functionality of linear functions, where messages consist of vectors over some domain and commitments can later be opened to a specific linear function of the vector coordinates. An opening for a function thus generates a witness for the fact that the function indeed evaluates to a given value for the committed message. One security requirement is called *function binding* and requires that no adversary be able to open a commitment to two different evaluations for the same function.

We propose a construction of functional commitment for linear functions based on constant-size assumptions in composite order groups endowed with a bilinear map. The construction has commitments and openings of constant size (i.e., independent of n or function description) and is *perfectly hiding* – the underlying message is information theoretically hidden. Our security proofs build on the Déjà Q framework of Chase and Meiklejohn (Eurocrypt 2014) and its extension by Wee (TCC 2016) to encryption primitives, thus relying on constant-size subgroup decisional assumptions. We show that FC for linear functions are sufficiently powerful to solve four open problems. They, first, imply polynomial commitments, and, then, give cryptographic accumulators (i.e., an algebraic hash function which makes it possible to efficiently prove that some input belongs to a hashed set). In particular, specializing our FC construction leads to the first pairing-based polynomial commitments and accumulators for large universes known to achieve security under simple assumptions. We also substantially extend our pairing-based accumulator to handle subset queries which requires a non-trivial extension of the Déjà Q framework.

1998 ACM Subject Classification E.3 Data Encryption, K.6.5 Security and Protection

* Full version available at https://hal.inria.fr/hal-01306152.

† The first author was funded by the "Programme Avenir Lyon Saint-Etienne de l'Université de Lyon" in the framework of the programme "Investissements d'Avenir" (ANR-11-IDEX-0007).

‡ The second author was funded by the "Programme Avenir Lyon Saint-Etienne de l'Université de Lyon" in the framework of the programme "Investissements d'Avenir" (ANR-11-IDEX-0007).

§ Part of this work was done while the third author was with Google Inc. and visiting the Simons Institute for Theory of Computing at U.C. Berkeley.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 30; pp. 30:1–30:14

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

Keywords and phrases Cryptography, commitment schemes, functional commitments, accumulators, provable security, pairing-based, simple assumptions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.30

1 Introduction

Commitment schemes are fundamental primitives used as building blocks in a number of cryptographic protocols. A commitment scheme emulates a publicly observed safe; it allows a party to commit to a message m so that this message is not revealed until a later moment when the commitment is opened and the receiver gets convinced that the message was indeed m. Two important security properties are called hiding and binding. The former requires that no information about the message is revealed to an observer. The latter property means that the committing party cannot change the message after committing to it.

Several works considered commitment schemes where the committer has the flexibility of only revealing some partial information about the message (rather than the entire message) during the opening phase. In vector commitments [22, 10], messages are *vectors* and commitments are only opened with respect to specific positions. Another example is polynomial commitments, where users commit to a polynomial and only reveal evaluations of this polynomial on certain inputs.

In this work, we consider functional commitments (FC) for linear functions. Namely, messages consist of vectors $(m_1, \ldots, m_n)$ and commitments can be partially opened by having the sender verifiably reveal a linear combination $\sum_{i=1}^{n} x_i \cdot m_i$, for public coefficients $\{x_i\}_{i=1}^{n}$. We show that this functionality implies many other natural functionalities, including vector commitments, polynomial commitments and cryptographic accumulators. We provide an efficient FC realization for linear functions based on well-studied assumptions in groups with a bilinear map. In turn, our scheme implies solutions to past natural questions. We give the first constructions under constant-size assumptions of two important primitives: polynomial commitments and cryptographic accumulators. In both cases, earlier solutions were based on non-standard assumptions where the number of input elements (and thus the strength of the assumption) depended on specific features of the schemes (like the maximal degree of committed polynomials). Our third result is a solution to an accumulator supporting subset queries, which is also based on constant size assumptions.

1.1 Related Works and the Open Problems

Functional commitments. Functional commitments can be seen as the natural commitment analogue of functional encryption [31, 6]. The latter primitive allows restricting what the receiver learns about encrypted data: when decrypting using a secret key SK_F for the function F, the decryptor learns $F(x)$ and nothing else. Likewise, FC schemes allow the committer to accurately control what the opening phase can reveal about the message.

Functional commitments were implicitly suggested by Gorbunov, Vaikuntanathan and Wichs [17] who described a statistically-hiding commitment scheme for which the sender is able to only reveal a circuit evaluation $C(x)$ when x is the committed input. While their solution supports arbitrary circuits and relies on well-studied lattice assumptions, its input x must be committed to in a bit-by-bit manner (or at least by splitting x into small blocks). We remark that, assuming a common reference string, non-interactive FC for general functionalities can be realized by combining ordinary statistically-hiding commitments with non-interactive zero-knowledge (NIZK) proofs [3]. Here, we focus on the problem of achieving

a better efficiency for more restricted (yet, sufficiently powerful for many applications) functionalities. Assuming a common reference string (as in all non-interactive perfectly hiding commitments), we aim at efficient constructions supporting short witnesses without resorting to the machinery of NIZK proofs. In particular, we aim at constant-size commitment strings (regardless of how long the message is) supporting concise witnesses.

In the literature, a number of earlier works consider settings where a sender is given the flexibility of revealing only a partial information about committed data. A verifiable random function [25], for example, can be seen as a perfectly binding commitment to a pseudo-random function key for which the committer can convince a verifier about the correct function evaluation for the committed key on a given input. Selective-opening security [16] addresses the problem of proving the security of un-opened commitments when an adversary gets to see the opening of other commitments to possibly correlated messages.

Zero-knowledge sets, as introduced by Micali, Rabin and Kilian [24], are another prominent example where users commit to a set S or an elementary database and subsequently prove the (non-)membership of some elements without revealing any further information (not even the cardinality of the committed set S). Ostrovsky, Rackoff and Smith [27] envisioned committed databases for which the sender can demonstrate more general statements than just membership and non-membership.

Vector commitments. Concise vector commitments were first suggested by Libert and Yung [22] and further developed by Catalano and Fiore [10]. They basically consist of Pedersen-like [30] commitments to vectors $(m_1, \ldots, m_n)$ where a constant-size opening (where "constant" means independent of n) allows the sender to open the commitment for only one coordinate m_i without revealing anything on other coordinates. The initial motivation of vector commitments was the design of zero-knowledge databases with short proofs [11, 22] via mercurial commitments [12] supporting short coordinate-wise openings [22]. While concise vector commitments can be based on long-lived hardness assumptions like RSA or Computational Diffie-Hellman [10], they either require groups of hidden order or public keys of size $O(n^2)$ if n is the dimension of committed vectors. In contrast, solutions based on variable-size assumptions allow for public keys of size $O(n)$, which leaves open the following problem.

Problem 1: *Is there a concise vector commitment scheme achieving linear-size public keys under constant-size assumptions in groups with a bilinear map?*

Polynomial commitments. As introduced by Kate, Zaverucha and Goldberg [19], polynomial commitments are a mechanism whereby a sender can generate a constant-size commitment to a polynomial $P[Z]$ (where "constant" means independent of the degree) in such a way that a constant-size witness can convince a verifier that the committed $P[Z]$ indeed evaluates to $P(i)$ for a given i. Polynomial commitments find natural applications in the context of verifiable secret sharing [14], anonymous credentials with attributes [7] or in optimized flavors of zero-knowledge databases which do not seek to hide the size of the committed set. They also imply vector commitments, as observed in [7]. Camenisch *et al.* [7] used vector commitments in a modular design of anonymous credentials where users' credentials are associated with descriptive attributes. While the commitments in [19, 7] were based on parameterized assumptions, the problem described below has been open.

Problem 2: *Design a polynomial commitment based on constant-size assumptions.*

Accumulators. Cryptographic accumulators can be interpreted as commitments, especially when the hashing algorithm is randomized. Accumulators [2] are closely related to zero-knowledge sets in that they make it possible to hash a set S while efficiently generating witnesses guaranteeing the inclusion of certain elements in the hashed set. Unlike zero-knowledge sets, they do not hide the cardinality of the underlying set but usually achieve a better efficiency via short membership witnesses. The first family of accumulators based on number theoretic techniques relies on groups of hidden order [2, 1, 23, 4] and includes proposals based on the Strong RSA assumption [1, 21]. The second family [26, 8], which was first explored by Nguyen [26], appeals to bilinear maps (a.k.a. pairings) and assumptions whose hardness depends on a parameter q determined by features of the scheme or the number of adversarial queries.

Solutions based on the Strong RSA assumption feature short public parameters and readily extend into universal accumulators [21] (where non-membership witnesses can show that a given input was not accumulated) or dynamic accumulators [9] (where witnesses can be autonomously updated when the hashed set is modified). On the other hand, they usually require expensive operations to injectively encode set elements as prime numbers. While pairing-based schemes [26, 8] do not need such a prime-number-encoding, they require linear-size public parameters in the maximal number of accumulated elements. On the positive side, they are useful in applications where the number of hashed elements cannot exceed a pre-determined bound. Pairing-based accumulators also proved useful in the context of authenticated data structures. Papamanthou *et al.* [29] used them to authenticate set operations and notably prove (using a constant-size witness) the inclusion of a given set in the accumulated set. The same technique was extended [29] to provide evidence that two accumulated sets have a given intersection.

A third family of accumulators [28, 4] builds on hash trees rather than number theoretic assumptions. Its disadvantage is that witnesses have size $O(\log N)$ (where N denote the cardinality of hashed sets) whereas number-theoretic solutions enable $O(1)$-size witnesses.

The security properties of accumulators were recently re-formalized by Derler *et al.* [15] who showed connections with other primitives. It was notably showed that, when endowed with an indistinguishability property, accumulators imply non-interactive commitment schemes and are implied by zero-knowledge sets.

Despite their numerous applications, cryptographic accumulators still have relatively few assumptions to rely on. So far, known candidates based on standard assumption arise from a generic construction from vector commitments [10]. While implying solutions based on RSA or Diffie-Hellman, the generic construction of [10] only supports inputs living in a small domain: the public key size is indeed linear in the size of the input universe, which prevents from hashing elements consisting of arbitrary strings. This leaves open Problem 3.

Problem 3: *Does there exist a pairing-based accumulator for large input universes secure under constant-size assumptions?*

As mentioned earlier, accumulators are applicable in authenticating set operations ([29]) and a useful extension would allow creating witnesses for set inclusion and intersection that are of constant size. Namely, a short witness can serve as evidence that some set X is a subset of the accumulated set or that two sets X_1, X_2 have a particular intersection I. In this domain, the following problem still remains open.

Problem 4: *Construct a pairing-based accumulator supporting set operations with constant-size witnesses achieving security under simple assumptions.*

1.2 Our Contributions

We formalize the notion of *functional commitments* (FCs) for linear functions, a generalization of vector commitments (VCs). Similar to VCs, such a commitment scheme allows committing to vectors of messages which can later be opened to specific function evaluations. While possible [17], the design of FCs for arbitrary functionalities seems unlikely to lead to truly efficient solutions. Instead, we aim at FCs for linear function families $\{F_{\vec{x}} : \mathcal{D}^n \times \mathcal{D}^n \to \mathcal{D}\}_{\vec{x}\in\mathcal{D}^n}$ defined by $F_{\vec{x}}(\vec{m}) = \langle \vec{x}, \vec{m}\rangle = \sum_{i=1}^n x_i m_i$ for $\vec{m} \in \mathcal{D}^n$ that suffice for many important applications. An FC scheme for a family of linear functions $\{F_{\vec{x}} : \mathcal{D}^n \to \mathcal{D}\}_{\vec{x}\in\mathcal{D}^n}$ produces commitments to messages of the form $\vec{m} = (m_1, \ldots, m_n) \in \mathcal{D}^n$ over the domain $\mathcal{D}$. Fixing a specific $\vec{x} \in \mathcal{D}^n$, such that $y = \sum_{i=1}^n x_i m_i \in \mathcal{D}$, an opening for $F_{\vec{x}}$ demonstrates that $F_{\vec{x}}(\vec{m})$ indeed evaluates to y. The security notions of hiding and binding extend to our setting in a natural way. In addition, we require the commitments and witnesses to be *concise* i.e., their size should be independent of the length of messages or function description.

Our first contribution is a construction of functional commitment for linear functions based on well-studied assumptions in composite order bilinear groups. The scheme is perfectly hiding and computationally binding under subgroup decision assumptions. The construction can be seen as a variant of the vector commitment scheme of Izabachène *et al.* [18] which was only proved secure under a non-standard variable-size assumption. We show that the composite-order setting makes it possible to use the Déjà Q framework of [13] so as to obtain security from constant size assumptions. As FC for linear functions implies vector commitments, our construction provides a positive answer to *Problem 1*.

As a second contribution, we show that our FC scheme implies polynomial commitments and large-universe accumulators supporting subset queries. The resulting schemes are secure under subgroup decision assumptions of constant-size thus settling *Problem 2* and *Problem 3*. We finally extend our accumulator into a scheme supporting subset queries while retaining security from constant size assumptions, partially answering *Problem 4* in the affirmative.

Overview of our Construction. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map with common group order $N = p_1p_2p_3$ and let $\mathbb{G}_q$ denote the subgroup of $\mathbb{G}$ of order q (here q would be of the form $p_1^{e_1}p_2^{e_2}p_3^{e_3}$ for $e_1, e_2, e_3 \in \{0, 1\}$). The linear functions will be defined over $\mathbb{Z}_N$. The commitment key consists of $\{g^{\alpha^j}\}_{j=1}^n$, $\{U_j = u^{\alpha^j}\}_{j\in[1,2n]\setminus\{n+1\}}$ for some $g, u \in \mathbb{G}_{p_1}$. The trapdoor is $U_{n+1} = u^{\alpha^{n+1}}$. A commitment to $\vec{m}$ consists of $C = g^\gamma \cdot \prod_{j=1}^n g^{\alpha^j m_j}$. Witness for a function evaluation $\langle \vec{x}, \vec{m}\rangle = y$ is defined as $W_y = \prod_{i=1}^n W_i^{x_i}$ with the $\mathbb{G}_{p_1}$ component of W_i being $u^{\alpha^{n-i+1}\gamma} \cdot \prod_{j=1, j\neq i}^n u^{\alpha^{n+1+j-i}m_j}$ for each $i = 1, \ldots, n$. The absence of U_{n+1} in the witness allows verifying that $y = \langle \vec{x}, \vec{m}\rangle$ by testing if $e(C, \prod_{i=1}^n u^{\alpha^{n-i+1}x_i}) = e(g^\alpha, u^{\alpha^n})^y \cdot e(g, W_y)$. The u-components are randomized with elements of $\mathbb{G}_{p_3}$. This modification does not affect verification since the $\mathbb{G}_{p_3}$ components get cancelled upon pairing with $\mathbb{G}_{p_1}$ elements. The scheme is a composite-order analogue of the one proposed in [22].

Proof Idea. A $(q_1 \to q_2)$ subgroup decision assumption requires random elements of $\mathbb{G}_{q_1}$ to be indistinguishable from random elements of $\mathbb{G}_{q_2}$. Using Wee's adaptation [32] of the Déjà Q framework [13], we prove that our FC scheme is computationally binding based on $(p_1 \to p_1p_2)$ and $(p_1p_3 \to p_1p_2p_3)$ subgroup decision assumptions. An adversary breaking the binding property is successful if it can produce a commitment C and two conflicting witnesses W_y and $W_{y'}$ for evaluation of a function $\vec{x}$. Given that both witnesses satisfy the verification equations, one can say that the adversary can essentially produce $\Delta W = \left(W_{y'}/W_y\right)^{1/(y-y')}$ which is of the form $u^{(\alpha^{n+1})} \cdot g_2^{r_2} \cdot g_3^{r_3}$ for some $r_2, r_3 \in \mathbb{Z}_N$ and generators $g_2 \in \mathbb{G}_{p_2}$ and

$g_3 \in \mathbb{G}_{p_3}$. The $\mathbb{G}_{p_1}$ component of ΔW is identical to that of the trapdoor key. Define two types of keys (resp. attacks) according to $\{U_j\}_{j=1}^{2n}$ (resp. ΔW) containing a $\mathbb{G}_{p_2}$ component or not. We argue that the attacker cannot mount an attack of a type different from that of the key based on the $(p_1 \to p_1 p_2)$. The distribution of $\mathbb{G}_{p_2}$ components for the keys are changed gradually via the transition described below.

$$u^{\alpha^i} R_{3,i} \xrightarrow{\text{subgroup}} u^{\alpha^i} g_2^{r_1 \alpha^i} R_{3,i} \xrightarrow{\text{CRT}} u^{\alpha^i} g_2^{r_1 \alpha_1^i} R_{3,i},$$

where α_1 is uniformly distributed over $\mathbb{Z}_N$. The first transition uses the $p_1 p_3 \to p_1 p_2 p_3$ subgroup decision assumptions and the second transition is based on the Chinese remainder theorem (CRT) that states that $\alpha \bmod p_1$ and $\alpha \bmod p_2$ are uncorrelated. We can thus replace $\alpha \bmod p_2$ by $\alpha_1 \bmod p_2$ as long as the former is unconditionally hidden from the attacker. By repeating the transition $2n$ times, we obtain the transformation: $u^{\alpha^i} \to u^{\alpha^i} g_2^{\sum_{j=1}^{2n} r_j \alpha_j^i} R'_{3,i}$.

The exponent of g_2 is a pseudorandom function [13, 32] and hence can be replaced by a random exponent, $RF(i)$ for U_i in particular. After the final transition, creating ΔW consistent with these keys amounts to predicting the value of the random function evaluated at $n+1$ (for the trapdoor U_{n+1}), which is statistically infeasible.

Polynomial Commitments from Simple Assumptions. We wish to commit to a polynomial $P[Z] = a_0 + a_1 Z + \cdots + a_{n-1} Z^{n-1}$ of degree n over $\mathcal{D}$ and reveal an opening for $P(x)$ for $x \in \mathcal{D}$. Using the FC scheme for linear functions, we can commit to $(a_0, \ldots, a_{n-1}) \in \mathcal{D}^n$ so that an opening to $P(x)$ is a witness for $\langle \vec{x}, \vec{m} \rangle = P(x)$ where $\vec{x} = (1, x, \ldots, x^{n-1})$.

Accumulators for Large Universes. An accumulator allows hashing a set to a single element so that one can prove the membership of a value in the set. Vector commitments are known to imply accumulators [10], but via a construction that only supports a small universe of values. Our polynomial commitment naturally leads to an accumulator for large universes (i.e., the domain size can be exponential in the security parameter). To accumulate a set of values $S = \{y_1, \ldots, y_{n-1}\}$, use a polynomial commitment to $P[Z] = \prod_{i=1}^{n-1} (Z - y_i)$. A witness for $x \in S$ (or $x \notin S$) is generated based on the fact $P[x] = 0$ if and only if $x \in S$.

Tackling Subset Queries. Polynomial commitments and universal accumulators can be seen as direct consequences of the FC for linear functions. On the other hand, proving security for accumulators with concise subset witnesses requires a novel extension of the Déjà Q framework. We now provide a brief outline of the same.

Let n be the maximal number of accumulated values and let d be the maximal size of "provable" subsets. In the commitment scheme, keys consisted of powers of α in the exponent over the interval $[1, 2n]$ with a hole at position $n+1$. We extend this interval to $[1, (d+1)n]$ keeping $n+1, 2n+1, \ldots, (d+1)n$ powers of α as part of the trapdoor. The witness component for a specific position i of the linear function was defined as $W_i = u^{\alpha^{n-i+1}\gamma} \cdot \prod_{j=1, j\neq i}^{n} u^{\alpha^{n+1+j-i} m_j}$. To combine witnesses for several (at most d) values into a constant-size witness, we define the witness for the i-th position of the ℓ-th element as a "shift" of W_i by n. More precisely, $W_{\ell,i}$ is defined to have $u^{\alpha^{\ell n - i + 1}\gamma} \cdot \prod_{j=1, j\neq i}^{n} u^{\alpha^{\ell n + 1 + j - i} m_j}$ as its $\mathbb{G}_{p_1}$ component.

Security for accumulators is captured by the notion of *collision-freeness* which asserts that it is computationally infeasible for an attacker to produce a set S and a witness W_X for a subset $X = \{x_1, \ldots, x_k\} \not\subseteq S$ that verifies correctly with an accumulated value for S (generated using randomness specified by the adversary). Given the randomness, the

reduction can compute valid witnesses of membership and non-membership for individual values in X (as in the normal accumulator scheme). Combining appropriate "shifts" of these witnesses gives us $W_{X\cap S}$ (combined membership witness) and $W_{X\setminus S}$ (combined non-membership witness). We then observe that $W/(W_{X\cap S}W_{X\setminus S})$ has a $\mathbb{G}_{p_1}$-component of the form $u^{\sum_{\ell\in[1,k],x_\ell\notin S} w_\ell \alpha^{\ell n+1}}$ ($w_\ell \neq 0$) which means that the attacker essentially produces a linear combination of the discrete logarithms of trapdoor keys in the exponent. The rest of the reduction proceeds similar to the FC scheme with the pseudorandom function now extending to the larger interval. Using this pseudorandom function, the distribution of the keys is gradually modified until the $\mathbb{G}_{p_2}$ components of all U_i's are truly random. We argue that generating such a witness requires the adversary to predict a linear combination of at most d specific evaluations of a random function which is clearly infeasible.

2 Background

2.1 Bilinear Maps and Complexity Assumptions

We use groups $(\mathbb{G}, \mathbb{G}_T)$ of composite order $N = p_1p_2p_3$ endowed with an efficiently computable map (a.k.a. pairing) $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ such that: (1) $e(g^a, h^b) = e(g, h)^{ab}$ for any $(g, h) \in \mathbb{G}\times\mathbb{G}$ and $a, b \in \mathbb{Z}$; (2) if $e(g, h) = 1_{\mathbb{G}_T}$ for each $h \in \mathbb{G}$, then $g = 1_{\mathbb{G}}$. Also, pairing two elements of order p_i and p_j, with $i \neq j$, always gives the identity element $1_{\mathbb{G}_T}$.

In the following, for each $i \in \{1, 2, 3\}$, we denote by $\mathbb{G}_{p_i}$ the subgroup of order p_i. For all distinct $i, j \in \{1, 2, 3\}$, we call $\mathbb{G}_{p_ip_j}$ the subgroup of order p_ip_j. We rely on the following assumptions introduced in [20], which are non-interactive, falsifiable. In both of them, the number of input elements is constant (regardless of the number of adversarial queries).

Assumption 1. Given a description of $(\mathbb{G}, \mathbb{G}_T)$ as well as $g \xleftarrow{R} \mathbb{G}_{p_1}, X_3 \xleftarrow{R} \mathbb{G}_{p_3}$ and $T \in \mathbb{G}$, it is infeasible to efficiently decide if $T \in \mathbb{G}_{p_1p_2}$ or $T \in \mathbb{G}_{p_1}$.

Assumption 2. Let $g, X_1 \xleftarrow{R} \mathbb{G}_{p_1}, X_2, Y_2 \xleftarrow{R} \mathbb{G}_{p_2}, Y_3, Z_3 \xleftarrow{R} \mathbb{G}_{p_3}$. Given a description of $(\mathbb{G}, \mathbb{G}_T)$, (g, X_1X_2, Z_3, Y_2Y_3) and T, it is hard to decide if $T \in_R \mathbb{G}_{p_1p_3}$ or $T \in_R \mathbb{G}$.

2.2 Vector Commitment Schemes

In prime order groups, Libert and Yung [22] introduced concise vector commitment schemes, which are commitments that can be opened with a short de-commitment string for each individual coordinate. Such commitments were described in [22, 10]. In [22], the commitment key is $CK = (g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}) \in \mathbb{G}^{2n}$, where $g_i = g^{(\alpha^i)}$ for each i. The trapdoor is g_{n+1}. To commit to $\vec{m} = (m_1, \ldots, m_n)$, one picks $r \xleftarrow{R} \mathbb{Z}_p$ and computes $C = g^r \cdot \prod_{j=1}^{n} g_{n+1-j}^{m_\kappa}$. A single element $W_i = g_i^r \cdot \prod_{j=1, j\neq i}^{n} g_{n+1-j+i}^{m_j}$ provides evidence that m_i is the i-th component of $\vec{m}$ as it satisfies $e(g_i, C) = e(g, W_i) \cdot e(g_1, g_n)^{m_i}$. The infeasibility of opening C to two distinct messages for some i relies on a parametrized assumption [5].

2.3 Functional Commitments for Linear Functions: Definitions

In [18], Izabachène *et al.* implicitly showed that the vector commitment scheme of [22] can be generalized into a commitment scheme allowing to commit to a vector $\vec{m}$ while proving – via a partial opening made of a short piece of information – that the committed vector $\vec{m}$ satisfies $\langle\vec{m}, \vec{x}\rangle = y$, for some public $\vec{m}$ and y. We call such a primitive *functional* commitment for *linear* functions. In this section, we formally define this primitive and its security.

▶ **Definition 1** (Functional Commitments)**.** Let $\mathcal{D}$ be a domain and consider linear functions $\langle\cdot,\cdot\rangle : \mathcal{D}^n \times \mathcal{D}^n \to \mathcal{D}$ defined by $\langle \vec{x}, \vec{m}\rangle = \sum_{i=1}^{n} x_i m_i$ for $\vec{x}, \vec{m} \in \mathcal{D}^n$ with $\vec{x} = (x_1, \ldots, x_n), \vec{m} = (m_1, \ldots, m_n)$. A functional commitment scheme FC for $(\mathcal{D}, n, \langle\cdot,\cdot\rangle)$ is a tuple of four (possibly probabilistic) polynomial time algorithms – (Setup,Commit,Open,Verify).

Setup$(1^\lambda, 1^n)$: takes in a security parameter $\lambda \in \mathbb{N}$, a desired message length $n \in \mathsf{poly}(\lambda)$ and outputs a commitment key CK and, optionally, a trapdoor TK.

Commit$(CK, \vec{m})$: takes as input the commitment key CK, a message vector $\vec{m} \in \mathcal{D}^n$ and outputs a commitment C for $\vec{m}$ and auxiliary information denotes aux.

Open$(CK, C, \mathsf{aux}, \vec{x})$: takes as input the commitment key CK, a commitment C (to $\vec{m}$), auxiliary information (possibly containing $\vec{m}$) and a vector $\vec{x} \in \mathcal{D}^n$; computes a witness W_y for $y = \langle \vec{x}, \vec{m}\rangle$ *i.e.,* W_y is a witness for the fact that the linear function defined by $\vec{x}$ when evaluated on $\vec{m}$ gives y.

Verify$(CK, C, W_y, \vec{x}, y)$: takes as input the commitment key CK, a commitment C, a witness W_y, a vector $\vec{x} \in \mathcal{D}^n$ and $y \in \mathcal{D}$; outputs 1 if W_y is a witness for C being a commitment for some $\vec{m} \in \mathcal{D}^n$ such that $\langle \vec{x}, \vec{y}\rangle = y$ and outputs 0 otherwise.

The correctness condition for a functional commitment scheme requires that for every $(CK, TK) \leftarrow \mathsf{Setup}(\lambda, n)$, for all $\vec{m}, \vec{x} \in \mathcal{D}^n$, if $(C, \mathsf{aux}) \leftarrow \mathsf{Commit}(CK, \vec{m})$ and $W_y \leftarrow \mathsf{Open}(CK, C, \mathsf{aux}, \vec{x})$, then $\mathsf{Verify}(CK, C, W_y, \vec{x}, y) = 1$ with probability 1.

The security requirements of functional commitments are formalized as follows. The perfect hiding property mandates that the distribution of the commitment string $\mathsf{Commit}(CK, \vec{m})$ be independent of the message $\vec{m}$.

▶ **Definition 2** (Perfectly Hiding)**.** A commitment scheme is perfectly hiding if for a key CK generated by an honest setup, for all $\vec{m}_1, \vec{m}_2 \in \mathcal{D}^n$ with $\vec{m}_1 \neq \vec{m}_2$, the two distributions $\{CK, \mathsf{Commit}(CK, \vec{m}_1)\}$ and $\{CK, \mathsf{Commit}(CK, \vec{m}_2)\}$ are identical given that the random coins of Commit are chosen according to the uniform distribution from the respective domain.

The binding property requires the infeasibility of generating a commitment C and accepting witnesses for two distinct values y, y' without knowing the trapdoor TK.

▶ **Definition 3** (Function Binding)**.** A functional commitment scheme $\mathsf{FC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ for $(\mathcal{D}, n, \langle\cdot,\cdot\rangle)$ is said to be computationally binding if any PPT adversary $\mathcal{A}$ has negligible advantage in winning the following game.

1. The challenger generates (CK, TK) by running $\mathsf{Setup}(\lambda, n)$ and gives CK to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ outputs a commitment C, a vector $\vec{x} \in \mathcal{D}^n$, two values $y, y' \in \mathcal{D}$ and two witnesses $W_y, W_{y'}$. We say that $\mathcal{A}$ wins the game if the following conditions hold.
   - **(i)** $y \neq y'$;
   - **(ii)** $\mathsf{Verify}(CK, C, W_y, \vec{x}, y) = \mathsf{Verify}(CK, C, W_{y'}, \vec{x}, y') = 1$.

2.4 Cryptographic Accumulators

The basic functionality of an accumulator is to combine a set S of values into a single value V so that for any $x \in S$ it is possible to prove that x is accumulated in V.

▶ **Definition 4** (Accumulator)**.** Let $\mathcal{D}$ be a domain. An accumulator scheme Acc for $\mathcal{D}$ is a tuple $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{WitCreate}, \mathsf{Verify})$ of PPT algorithms defined as follows.

Setup$(1^\lambda, 1^n)$: takes as input a security parameter λ and an integer $n \in \mathbb{N}$ upper bounding the number of elements that can be accumulated; outputs a pair of keys (PK, SK).

Eval(PK, S): inputs a key PK, a set $S \subset \mathcal{D}$ of elements (with $|S| \leq n$) to be accumulated and outputs an accumulated value V along with some auxiliary information aux.

WitCreate$(PK, S, V, \mathsf{aux}, x, \mathsf{type})$**:** inputs a public key PK, a set S, a pair of accumulated value and state information (V, aux) generated by $\mathsf{Eval}(PK, S)$, an element $x \in \mathcal{D}$ and a boolean value $\mathsf{type} \in \{0, 1\}$ indicating whether the output should be membership or non-membership witness according as its value is 1 or 0 respectively.
Case type $= 1$**:** If $x \notin S$, it returns $\perp$. Otherwise, a membership witness W is returned.
Case type $= 0$**:** It returns $\perp$ if $x \in S$ and a non-membership witness W otherwise.

Verify$(PK, V, W, x, \mathsf{type})$**:** takes as input the public key PK, an accumulator V for set S, a witness W, an element $x \in \mathcal{D}$ and a boolean value type. Returns 1 if and only if either
- W is a valid witness for $x \in S$ and $\mathsf{type} = 1$
- W is a valid witness for $x \notin S$ and $\mathsf{type} = 0$.

The above definition consider static accumulators. In dynamic accumulators, the accumulated value as well as witnesses can be publicly updated whenever an element is added to or deleted from the set. In this work, we only consider static accumulators.

The correctness condition requires that for all honestly generated keys, all honestly computed accumulators and witnesses, the Verify algorithm always accepts. An accumulator scheme is deemed secure if it is at least *collision-free*. Collision-freeness ensures the computational infeasibility of producing either a membership witness for a non-accumulated value or a non-membership witness for an accumulated value.

In accumulators supporting subset queries, witnesses can be generated for a subset of the accumulated set rather than individual elements. While accumulators have been defined in the universal setting, i.e., both membership and non-membership witnesses can be generated, here we only consider the non-universal setting.

▶ **Definition 5** (Accumulator with subset queries)**.** Let $\mathcal{D}$ be a domain. An accumulator scheme Acc for $\mathcal{D}$ is defined by a tuple (Setup, Eval, WitCreate, Verify) of probabilistic polynomial time algorithms defined as follows.

Setup$(1^\lambda, 1^n, 1^d)$**:** takes as input a security parameter λ, an upper bound $n \in \mathbb{N}$ on the number of elements that can be accumulated and an integer $d \in \mathbb{N}$ denoting the maximum size of a set for which a witness can be created; outputs a pair of keys (PK, SK).

Eval(PK, S)**:** takes in a public key PK, a set $S \subset \mathcal{D}$ of elements (with $|S| \leq n$) to be accumulated and outputs an accumulated value V with some auxiliary information aux.

WitCreate$(PK, S, V, \mathsf{aux}, X)$**:** inputs a public key PK, a set S, a pair of accumulated value and state information (V, aux) generated by $\mathsf{Eval}(PK, S)$, a set $X \subseteq S$ with $|X| \leq d$ and outputs a witness W_X.

Verify(PK, V, W_X, X)**:** takes as input the public key PK, an accumulator V for set S, a witness W_X, a set $X \subseteq S$. Returns 1 if W_X is a witness for $X \subseteq S$ and $\perp$ otherwise.

In the above syntax, we assume that the auxiliary information aux includes the randomness that was used to compute V when Eval is a probabilistic algorithm.

3 A Functional Commitment from Subgroup Decision Assumptions

Here, we prove that the Déjà Q framework [13] allows proving the security of the functional commitment of [18] under constant size assumptions by switching to composite order groups.

Setup$(1^\lambda, 1^n)$**:** Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of composite order $N = p_1p_2p_3$, where $p_i > 2^{l(\lambda)}$ for each $i \in \{1, 2, 3\}$, for a suitable polynomial $l : \mathbb{N} \to \mathbb{N}$. Choose $g, u \xleftarrow{R} \mathbb{G}_{p1}$, $R_3 \xleftarrow{R} \mathbb{G}_{p_3}$ and $\alpha \xleftarrow{R} \mathbb{Z}_N$ at random in order to define $G_j = g^{\alpha^j}$ for each $j \in [1, n]$ and

$$U_1 = u^\alpha \cdot R_{3,1}, \quad \ldots \quad U_n = u^{(\alpha^n)} \cdot R_{3,n},$$
$$U_{n+2} = u^{(\alpha^{n+2})} \cdot R_{3,n+2}, \quad \ldots \quad U_{2n} = u^{(\alpha^{2n})} \cdot R_{3,2n},$$

where $R_{3,j} \overset{R}{\leftarrow} \mathbb{G}_{p_3}$ for each $j \in [1, 2n]\backslash\{n+1\}$. Define the commitment key to consist of $CK := \big(g,\ \{G_j\}_{j=1}^n,\ \{U_j\}_{j\in[1,2n]\backslash\{n+1\}},\ R_3\big)$. The trapdoor consists of $TK := U_{n+1} = u^{(\alpha^{n+1})} \cdot R_{3,n+1}$, where $R_{3,n+1} \overset{R}{\leftarrow} \mathbb{G}_{p_3}$.

Commit($CK, \vec{m}$): Given $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_N^n$, compute $C = g^\gamma \cdot \prod_{j=1}^n G_j^{m_j}$ for a random choice of $\gamma \overset{R}{\leftarrow} \mathbb{Z}_N$ and output C with the auxiliary information $\mathsf{aux} = (m_1, \ldots, m_n, \gamma)$.

Open($CK, C, \mathsf{aux}, \vec{x}$): Given $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_N^n$, the auxiliary information $\mathsf{aux} = (m_1, \ldots, m_n, \gamma)$ allows generating a witness for the function $\langle \vec{m}, \vec{x} \rangle = \sum_{i=1}^n m_i \cdot x_i$ by computing

$$W_i = U_{n-i+1}^\gamma \cdot \prod_{j=1, j\neq i}^{n} U_{n+1+j-i}^{m_j} \qquad \forall i \in \{1, \ldots, n\}, \tag{1}$$

and outputting $W_y = \prod_{i=1}^n W_i^{x_i}$.

Verify($CK, C, W_y, \vec{x}, y$): Given $C \in \mathbb{G}$ and $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_N^n$, accept $W_y \in \mathbb{G}$ as evidence that C is a commitment to $\vec{m} \in \mathbb{Z}_N^n$ such that $y = \langle \vec{m}, \vec{x} \rangle$ if and only if it holds that $e(C, \prod_{i=1}^n U_{n-i+1}^{x_i}) = e(G_1, U_n)^y \cdot e(g, W_y)$. If so, output 1. Otherwise, return 0.

The correctness is verified by observing that, for each $i \in \{1, \ldots, n\}$, (1) implies that

$$e(C, U_{n-i+1}) = e(g,u)^{(\alpha^{n+1})m_i} \cdot e\big(g, U_{n-i+1}^\gamma \prod_{j=1, j\neq i}^{n} U_{n+j-i+1}^{m_j}\big) = e(G_1, U_n)^{m_i} \cdot e(g, W_i).$$

By raising both members of the above equality to the power $x_i \in \mathbb{Z}_N$ and taking the product over all $i \in [1, n]$, we find that W_y satisfies $e(C, \prod_{i=1}^n U_{n-i+1}^{x_i}) = e(G_1, U_n)^{\langle \vec{m}, \vec{x} \rangle} \cdot e(g, W_y)$.

It is clear that that the commitment is perfectly hiding: since C lives in the cyclic subgroup $\mathbb{G}_{p_1}$, any vector $(m_1, \ldots, m_n) \in \mathbb{Z}_N^n$ has a corresponding opening $\gamma \in \mathbb{Z}_N$ (and even $p_2 p_3$ openings since only $\gamma \bmod p_1$ is fixed by $\vec{m}$).

We now prove it computationally binding under subgroup assumptions. While this property can be proved via a reduction from the one-wayness of Wee's broadcast encryption [32, Section 4], we found it interesting to give a direct proof under the underlying assumptions for two reasons. First, this proof allows relying on a computational (rather than decisional) analogue of Assumption 1. Second, the proof provides insights allowing to prove the security of variants of this commitment or the other primitives it implies. For example, by adapting the proof of Theorem 6, we design an accumulator supporting subset queries in section 5. Since the latter scheme has a public key containing more elements than in [32], it can hardly be proved secure via a reduction from the security of Wee's broadcast encryption [32].

The proof involves two computationally indistinguishable distributions of parameters (CK, TK). The normal distribution is as in the real scheme whereas the semi-functional distribution allows CK and TK to have a $\mathbb{G}_{p_2}$ component. As in [32, Theorem 2], we use the Déjà Q framework so as to gradually move to a game where the $\{U_i\}_{i=1}^{2n}$ all contain a $\mathbb{G}_{p_2}$ component $g_2^{R(i)}$ which is determined by a random function $R : [1, 2n] \to \mathbb{Z}_{p_2}$. As in [22, 18], we rely on the fact that any attack against the binding property publicly reveals a value U_{n+1} which contains $u^{(\alpha^{n+1})}$ as its $\mathbb{G}_{p_1}$ component. Depending on whether U_{n+1} contains a $\mathbb{G}_{p_2}$ component or not, we speak of Type B or Type A attacks. The proof uses a subsequence of $2n$ games where, in the k-th game, the $\mathbb{G}_{p_2}$ component of U_i is of the form $g_2^{F_k(i)}$, where $F_k : [1, 2n] \to \mathbb{Z}_{p_2}$ is a k-wise independent function. The strategy of the proof is to show that, unless either Assumption 1 or Assumption 2 can broken, the attack on the binding property also reveals a U_{n+1} of the form $U_{n+1} = u^{(\alpha^{n+1})} \cdot g_2^{F_k(n+1)} \cdot \mathcal{R}_3$, for some $R_3 \in \mathbb{G}_{p_3}$ in the k-th game. Said otherwise, the attack reveals a trapdoor U_{n+1} which mimics the distribution of the commitment key CK. When we reach the $2n$-th game, the $\mathbb{G}_{p_2}$ component of each U_i is determined by $F_{2n}(i)$. Since $F_{2n}(.)$ is a $2n$-wise independent function, the $\mathbb{G}_{p_2}$ of U_{n+1}

is thus statistically independent of those of $\{U_i\}_{i\in[1,2n]\setminus\{i\}}$, which appear in the public key. The proof of Theorem 6 is given in the full version of the paper.

▶ **Theorem 6.** *The scheme is binding if Assumption 1 and Assumption 2 both hold.*

4 Further Constructions

4.1 Polynomial Commitments from Constant-Size Assumptions

It is easy to see that any functional commitment for linear functions implies a polynomial commitment. Indeed, in order to commit to a polynomial $P[Z] = a_0 + a_1 Z + \cdots + a_{n-1}Z^{n-1}$ of degree $n-1$, we can simply commit to the vector of coefficients $\vec{m} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{Z}_N^n$. When the sender wants to convince a verifier that $P(x) = y$, for some public $x, y \in \mathbb{Z}_N$, it is sufficient to generate a witness W_y showing that $\langle \vec{m}, \vec{x} \rangle = y$, where $\vec{x} = (1, x, x^2, \ldots, x^{n-1})$. Our construction of Section 3 thus implies the first polynomial commitment based on constant-size assumptions. Indeed, the schemes of [19, 7] rely on q-type assumptions where q is proportional to the maximal degree of committed polynomials.

4.2 Large-Universe Pairing-Based (Universal) Accumulators from Constant-Size Assumptions

Catalano and Fiore [10] designed cryptographic accumulators from vector commitments. While their construction yields an accumulator based on the Diffie-Hellman assumption, it only supports small universes. Namely, accumulated values should come from a polynomial-size domain since the public key has linear size in the cardinality of this domain.

It is easy to see that polynomial commitments imply accumulators for exponential-size universes. While the size of the public key is linear in the maximal number of accumulated values (as in Nguyen's accumulator [26]), it does not depend of the universe size. As a result, we can accumulate inputs consisting of arbitrary strings of polynomial length.

In order to accumulate a set $S = \{x_1, \ldots, x_{n-1}\}$, one can commit to the vector $(a_0, a_1, \ldots, a_{n-2}, 1)$ that contains the coefficients of the polynomial $P[Z] = \prod_{j=1}^{n-1}(Z - x_j)$ and rely on the fact that $x \in S$ if and only if $P(x) = 0$. A witness that $x_i \in S$ (resp. $x_i \notin S$) is obtained by generating a witness that the committed polynomial satisfies $P(x_i) = 0$ (resp. $P(x_i) \neq 0$)). A concrete construction based on Assumptions 1 and 2 is described in the the full version.

5 Accumulators Supporting Subset Queries

We now generalize the accumulator of Section 4.2 so that a constant-size witness $W \in \mathbb{G}$ can provide evidence that a purported set X is contained in the hashed set S. Such a commitment was previously designed by Papamanthou *et al.* [29] under a non-standard q-type assumption. Our construction is thus the first realization based on fixed-size assumptions.

Gen$(1^\lambda, 1^n)$: Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of composite order $N = p_1p_2p_3$, where $p_i > 2^{l(\lambda)}$ for each $i \in \{1,2,3\}$, for a suitable polynomial $l : \mathbb{N} \to \mathbb{N}$. Choose $g, u \xleftarrow{R} \mathbb{G}_{p_1}$, $R_3 \xleftarrow{R} \mathbb{G}_{p_3}$ and $\alpha \xleftarrow{R} \mathbb{Z}_N$ at random. Let $d \leq n$ be the bound placed on size of a subset (also polynomial in the security parameter). Define $G_i = g^{(\alpha^i)}$ for each $i \in [1, n]$ and $U_j = u^{(\alpha^j)} \cdot R_{3,j}$, where $R_{3,j} \xleftarrow{R} \mathbb{G}_{p_3}$ for each $j \in [1, (d+1)n] \setminus \{n+1, 2n+1, \ldots, dn+1\}$.

The secret key is $SK := \{U_{\ell n+1}\}_{\ell=1}^{d}$, where $U_{\ell n+1} = u^{(\alpha^{\ell n+1})} \cdot R_{3,\ell n+1}$ with $R_{3,\ell n+1} \xleftarrow{R} \mathbb{G}_{p_3}$ for all $\ell \in [1, d]$. The public key is

$$PK := \big(g, \{G_j\}_{j=1}^{n}, \{U_j\}_{j\in[1,(d+1)n]\setminus\{n+1,2n+1,\ldots,dn+1\}}, R_3\big).$$

Eval(PK, S): To hash a set $S = \{y_1, \ldots, y_{n'}\}$ of cardinality $n' \leq n-1$, expand the polynomial $P_S[Z] = \prod_{j=1}^{n'}(Z - y_j) = \sum_{j=0}^{n'} m_j \cdot Z^j$. Choose $\gamma \xleftarrow{R} \mathbb{Z}_N$ to compute and output

$$V = g^{\gamma} \cdot \prod_{j=1}^{n'+1} G_j^{m_{j-1}} = g^{\gamma+\alpha\cdot P_S(\alpha)}, \qquad \mathsf{aux} = (S, \gamma) \tag{2}$$

WitCreate($PK, V, S, \mathsf{aux}, X$): Given a set $S = \{y_1, \ldots, y_{n'}\}$, a subset $X = \{x_1, \ldots, x_k\} \subseteq S$ of size $k \leq d$ (we assume w.l.o.g. that $x_1, \ldots, x_k$ are arranged in some pre-determined lexicographical order), and the state information $\mathsf{aux} = (S, \gamma)$ such that (V, aux) was produced by $\mathsf{Acc}(PK, S)$, compute $P_S[Z] = \prod_{j=1}^{n'}(Z - y_j) = \sum_{j=0}^{n'} m_j \cdot Z^j$ and define the corresponding vector $\vec{m} = (m_0, m_1, \ldots, m_{n'}, 0, \ldots, 0) \in \mathbb{Z}_N^n$. For each $\ell \in [1, k]$, define $\vec{x}_\ell = (x_{\ell,1}, \ldots, x_{\ell,n}) = (1, x_\ell, x_\ell^2, \ldots, x_\ell^n) \in \mathbb{Z}_N^n$ which satisfies $P_S(x_\ell) = \vec{m} \cdot \vec{x}_\ell = 0$. For $\ell \in [1, k]$, generate a witness that $\langle \vec{m}, \vec{x}_\ell \rangle = 0$ by first using $\{U_{\ell n+1+j-i}\}_{j\neq i}$ to compute

$$W_{\ell,i} = U_{\ell n-i+1}^{\gamma} \cdot \prod_{j=1, j\neq i}^{n} U_{\ell n+1+j-i}^{m_j} \qquad \forall i \in \{1, \ldots, n\}, \tag{3}$$

which satisfies $e(V, \prod_{i=1}^{n} U_{\ell n+1-i}^{x_{\ell,i}}) = e(g, W_{\ell,i})$ for all $\ell \in [1, k]$ since $\vec{m} \cdot \vec{x}_\ell = 0$. Then, compute and output the witness $W_X = \prod_{\ell=1}^{k} \prod_{i=1}^{n} W_{\ell,i}^{x_{\ell,i}}$.

Verify(PK, V, W_X, X): Given an accumulator value $V \in \mathbb{G}$, a subset $X = \{x_1, \ldots, x_k\}$, where $x_i \in \mathbb{Z}_N$ for each $i \in [1, k]$, and a candidate a witness W_X, do the following.

1. For each $\ell \in [1, k]$, define $\vec{x}_\ell = (x_{\ell,1}, \ldots, x_{\ell,n}) = (1, x_\ell, \ldots, x_\ell^n) \in \mathbb{Z}_N^n$.
2. Return 1 if and only if $e(V, \prod_{\ell=1}^{k} \prod_{i=1}^{n} U_{\ell n+1-i}^{x_{\ell,i}}) = e(g, W_X)$.

From an efficiency standpoint, the size of PK is quadratic in n when $d \approx n$ so as to handle queries for arbitrary subsets of size $\leq n$. In comparison with [29], we thus achieve security under simple assumptions at the expense of a somewhat larger public key. We see it as an interesting open problem to retain $O(n)$-size public keys under simple assumptions.

We prove that the scheme provides collision-freeness (a detailed definition is given in the full version) in that no PPT adversary can output a set S (of size $\leq n$) along with a verifying witness W_X for another set X which is *not* contained in S. We thus use a natural analogue of the definition of collision-freeness used in [15]: since our evaluation algorithm is randomized, we assume that the adversary outputs the set S and the random coins γ of the evaluation algorithm that lead to the accumulator value for which W_X verifies.

The proof relies on the fact that the adversary outputs both the hashed set S and the random coins γ of the hashing algorithm. It allows the reduction to use W_X in order to extract a membership witness for the difference $X \setminus S$ using the homomorphic properties of the underlying commitment. Having obtained $W_{X\setminus S}$, the reduction is also able to compute an aggregation of non-membership witnesses for the same difference $X \setminus S$. From these two conflicting witnesses, it is possible to extract some linear combination of the secret key components $\{U_{\ell n+1}\}_{\ell=1}^{d}$. In turn, this forces the adversary to predict a linear combination of random function evaluations in the final step of the sequence of games. The proof is given in the full version of the paper.

▶ **Theorem 7.** *The scheme is collision-free if Assumption 1 and Assumption 2 hold.*

References

1 Niko Baric and Birgit Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 480–494. Springer, 1997.

2 Josh Cohen Benaloh and Michael de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Sinatures (Extended Abstract). In *EUROCRYPT'93*, volume 765 of *LNCS*, pages 274–285. Springer, 1993.

3 Manuel Blum, Paul Feldman, and Silvio Micali. Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract). In *STOC 1988*, pages 103–112. ACM, 1988.

4 Dan Boneh and Henry Corrigan-Gibbs. Bivariate Polynomials Modulo Composites and Their Applications. In *ASIACRYPT 2014*, volume 8873 of *LNCS*, pages 42–62. Springer, 2014.

5 Dan Boneh, Craig Gentry, and Brent Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *CRYPTO 2005*, volume 3621 of *LNCS*, pages 258–275. Springer, 2005.

6 Dan Boneh, Amit Sahai, and Brent Waters. Functional Encryption: Definitions and Challenges. In *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, 2011.

7 Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and Modular Anonymous Credentials: Definitions and Practical Constructions. In *ASIACRYPT 2015*, volume 9453 of *LNCS*, pages 262–288. Springer, 2015.

8 Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *PKC 2009*, volume 5443 of *LNCS*, pages 481–500. Springer, 2009.

9 Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.

10 Dario Catalano and Dario Fiore. Vector Commitments and Their Applications. In *PKC 2013*, volume 7778 of *LNCS*, pages 55–72. Springer, 2013.

11 Dario Catalano, Dario Fiore, and Mariagrazia Messina. Zero-Knowledge Sets with Short Proofs. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 433–450. Springer, 2008.

12 Melissa Chase, Alexander Healy, Anna Lysyanskaya, Tal Malkin, and Leonid Reyzin. Mercurial Commitments with Applications to Zero-Knowledge Sets. In *EUROCRYPT 2005, Proceedings*, volume 3494 of *LNCS*, pages 422–439. Springer, 2005.

13 Melissa Chase and Sarah Meiklejohn. Déjà Q: Using Dual Systems to Revisit q-Type Assumptions. In *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 622–639. Springer, 2014.

14 Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *FOCS 1985*, pages 383–395. IEEE Computer Society, 1985.

15 David Derler, Christian Hanser, and Daniel Slamanig. Revisiting Cryptographic Accumulators, Additional Properties and Relations to Other Primitives. In *CT-RSA 2015*, volume 9048 of *LNCS*, pages 127–144. Springer, 2015.

16 Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic Functions. *J. ACM*, 50(6):852–921, 2003.

17 Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled Fully Homomorphic Signatures from Standard Lattices. In *STOC 2015*, pages 469–477. ACM, 2015.

18 Malika Izabachène, Benoît Libert, and Damien Vergnaud. Block-Wise P-Signatures and Non-interactive Anonymous Credentials with Efficient Attributes. In *IMACC 2011*, volume 7089 of *LNCS*, pages 431–450. Springer, 2011.

19 Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, 2010.

20 Allison B. Lewko and Brent Waters. New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In *TCC 2010, Proceedings*, volume 5978 of *LNCS*, pages 455–479. Springer, 2010.

21 Jiangtao Li, Ninghui Li, and Rui Xue. Universal Accumulators with Efficient Nonmembership Proofs. In *ACNS 2007*, volume 4521 of *LNCS*, pages 253–269. Springer, 2007.

22 Benoît Libert and Moti Yung. Concise Mercurial Vector Commitments and Independent Zero-Knowledge Sets with Short Proofs. In *TCC 2010, Proceedings*, volume 5978 of *LNCS*, pages 499–517. Springer, 2010.

23 Helger Lipmaa. Secure Accumulators from Euclidean Rings without Trusted Setup. In *ACNS 2012*, volume 7341 of *LNCS*, pages 224–240. Springer, 2012.

24 Silvio Micali, Michael O. Rabin, and Joe Kilian. Zero-Knowledge Sets. In *FOCS 2003*, pages 80–91. IEEE Computer Society, 2003.

25 Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable Random Functions. In *FOCS'99*, pages 120–130. IEEE Computer Society, 1999.

26 Lan Nguyen. Accumulators from Bilinear Pairings and Applications. In *CT-RSA 2005*, volume 3376 of *LNCS*, pages 275–292. Springer, 2005.

27 Rafail Ostrovsky, Charles Rackoff, and Adam D. Smith. Efficient Consistency Proofs for Generalized Queries on a Committed Database. In *ICALP 2004*, volume 3142 of *LNCS*, pages 1041–1053. Springer, 2004.

28 Charalampos Papamanthou, Elaine Shi, Roberto Tamassia, and Ke Yi. Streaming Authenticated Data Structures. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 353–370. Springer, 2013.

29 Charalampos Papamanthou, Roberto Tamassia, and Nikos Triandopoulos. Optimal Verification of Operations on Dynamic Sets. In *CRYPTO 2011*, volume 6841 of *LNCS*, pages 91–110. Springer, 2011.

30 Torben P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, 1991.

31 Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In *EUROCRYPT 2005, Proceedings*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

32 Hoeteck Wee. Déjà Q: Encore! Un Petit IBE. In *TCC 2016-A*, volume 9563 of *LNCS*, pages 237–258. Springer, 2016.

Block-Wise Non-Malleable Codes*

Nishanth Chandran[1], Vipul Goyal[2], Pratyay Mukherjee†[3], Omkant Pandey[4], and Jalaj Upadhyay‡[5]

1 **Microsoft Research, India**
nichandr@microsoft.com
2 **Microsoft Research, India**
vipul@microsoft.com
3 **University of California, Berkeley, USA**
pratyay85@berkeley.edu
4 **Drexel University, Philadelphia, USA**
omkant@drexel.edu
5 **Pennsylvania State University, State College, USA**
jalaj@psu.edu

Abstract

Non-malleable codes, introduced by Dziembowski, Pietrzak, and Wichs (ICS '10) provide the guarantee that if a codeword c of a message m, is modified by a tampering function f to c', then c' either decodes to m or to "something unrelated" to m. In recent literature, a lot of focus has been on explicitly constructing such codes against a large and natural class of tampering functions such as *split-state* model in which the tampering function operates on different parts of the codeword *independently*.

In this work, we consider a stronger adversarial model called *block-wise tampering* model, in which we allow tampering to depend on more than one block: if a codeword consists of two blocks $c = (c_1, c_2)$, then the first tampering function f_1 could produce a tampered part $c'_1 = f_1(c_1)$ and the second tampering function f_2 could produce $c'_2 = f_2(c_1, c_2)$ depending on *both* c_2 and c_1. The notion similarly extends to multiple blocks where tampering of block c_i could happen with the knowledge of all c_j for $j \leq i$. We argue this is a natural notion where, for example, the blocks are sent one by one and the adversary must send the tampered block before it gets the next block.

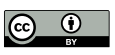
A little thought reveals that it is impossible to construct such codes that are non-malleable (in the standard sense) against such a powerful adversary: indeed, upon receiving the last block, an adversary could decode the entire codeword and then can tamper depending on the message. In light of this impossibility, we consider a natural relaxation called *non-malleable codes with replacement* which requires the adversary to produce not only related but also a valid codeword in order to succeed. Unfortunately, we show that even this relaxed definition is not achievable in the information-theoretic setting (i.e., when the tampering functions can be unbounded) which implies that we must turn our attention towards computationally bounded adversaries.

As our main result, we show how to construct a block-wise non-malleable code (BNMC) from sub-exponentially hard one-way permutations. We provide an interesting connection between BNMC and non-malleable commitments. We show that any BNMC can be converted into a non-malleable (w.r.t. opening) commitment scheme. Our techniques, quite surprisingly, give rise to a

* Part of this work was done while Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay were visiting Microsoft Research, India.

† Pratyay Mukherjee was supported in part from a DARPA/ARL SAFEWARE award, AFOSR Award FA9550-15-1-0274, and NSF CRII Award 1464397. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

‡ Jalaj Upadhyay was supported by the NSF Award IIS-1447700. Part of this work was done while the author was at the University of Waterloo. The views expressed are those of the author and do not reflect the official policy or position of the National Science Foundation.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 31; pp. 31:1–31:14

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

non-malleable commitment scheme (secure against so-called synchronizing adversaries), in which *only* the committer sends messages. We believe this result to be of independent interest. In the other direction, we show that any non-interactive non-malleable (w.r.t. opening) commitment can be used to construct BNMC only with 2 blocks. Unfortunately, such commitment scheme exists only under highly non-standard assumptions (adaptive one-way functions) and hence can not substitute our main construction.

1998 ACM Subject Classification E.2 Public key cryptosystems

Keywords and phrases Non-malleable codes, Non-malleable commitments, Block-wise Tampering, Complexity-leveraging

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.31

1 Introduction

Non-malleable codes. Error correcting codes allow a message m to be encoded into a codeword c, such that m can always recovered even from a tampered codeword c', only if the tampering is done in a specific way. More formally, the class of tampering functions, $\mathcal{F}_{\mathsf{frac}}$, tolerated by traditional error correction codes are ones that erase or modify only a constant fraction of the codeword c. In particular, no guarantees are provided on the output of the decoding algorithm when the tampering function $f \notin \mathcal{F}_{\mathsf{frac}}$. A more relaxed notion, error detecting codes, allow the decoder to also output a special symbol $\perp$, when m is unrecoverable from c', but here too, the codes can not tolerate simple tampering functions $f \in \mathcal{F}_{\mathsf{const}}$ where $\mathcal{F}_{\mathsf{const}}$ contains all constant functions[1]. To address this shortcoming of error correction/detection codes, Dziembowski, Pietrzak, and Wichs [12], introduced a more flexible notion of *non-malleable codes* (NMC). Informally, an encoding scheme $\mathsf{Code} := (\mathsf{Enc}, \mathsf{Dec})$ is a NMC against a class of tampering functions, $\mathcal{F}$, if the following holds: the decoded message $m' = \mathsf{Dec}(c')$ is either equal to the original message m or is completely unrelated to m, when $c' = f(\mathsf{Enc}(m))$ for some $f \in \mathcal{F}$. In general, NMC cannot exist for the set of all tampering functions $\mathcal{F}_{\mathsf{all}}$. To see this, observe that a tampering function that simply runs the decode algorithm to retrieve m, and then encodes a message related to m, trivially defeats the requirement above. However, somewhat surprisingly, Dziembowski *et al.* [12] showed the (probabilistic) existence of a NMC against a function family, $\mathcal{F}_{\mathsf{almost}}$, that is only slightly smaller than the set of all functions. They also constructed an efficient NMC against the class of tampering functions, $\mathcal{F}_{\mathsf{bit}}$, that can tamper each bit of the codeword independently. NMC has found important applications in tamper-resilient cryptography [12, 21, 13, 14].

Split-state Tampering. Arguably, one of the strongest class of tampering functions for which explicit constructions of NMC are known, is in the so called *split-state model*. Informally, a split-state model with ℓ states has the following attributes: (i) the codeword is assumed to be partitioned into ℓ-disjoint blocks $(c_1, \cdots, c_\ell)$, and (ii) the class of tampering functions, $\mathcal{F}^{\ell}_{\mathsf{split}}$, consists of all the functions $(f_1, \cdots, f_\ell)$ where f_i operates *independently* on c_i[2]. Dziembowski *et al.* [12] gave a construction of a NMC against the tampering class $\mathcal{F}^{2}_{\mathsf{split}}$ in the random oracle

[1] In particular if f always outputs some valid codeword c', then it is impossible to detect the error. For some cryptographic application like protecting against memory tampering attack this is too restrictive.
[2] Note that the class $\mathcal{F}_{\mathsf{bit}}$ can be viewed as $\mathcal{F}^{n}_{\mathsf{split}}$, where n is the length of the codeword c.

model. Constructions of NMC against $\mathcal{F}^2_{\mathsf{split}}$ are now known both in the computational [21][3] and information-theoretic settings [2, 7, 11]. Recently, Chattopadhyay and Zuckerman [6] gave an explicit information-theoretic NMC against $\mathcal{F}^{10}_{\mathsf{split}}$ and Aggarwal *et al.* [1] showed how to construct explicit information-theoretic NMC against $\mathcal{F}^2_{\mathsf{split}}$.

Going beyond split-state: Block-wise Tampering. A severe restriction of the split-state model is that every block of the codeword can only be tampered *independently* of all other blocks. In particular f_i modifies c_i with absolutely no knowledge about c_j, for any $j \neq i$. In this work, we address this restriction by allowing modification of *each* block *depending* on more than one-block. In particular, each c_i can be modified in any arbitrary way based on the first i blocks $(c_1, \ldots, c_i)$. Such a code is called *block-wise* NMC. More formally a code is called a *block-wise* NMC if it is a NMC against the class of tampering functions $\mathcal{F}^\ell_{\mathsf{block}}$: a set of functions $(f_1, \cdots, f_\ell) \in \mathcal{F}^\ell_{\mathsf{block}}$ if each f_i modify c_i to some c'_i depending on the first i-blocks. We also consider a stronger class of functions where the tampering can be done in *any* order. In particular f_i can modify any c_j depending on any i blocks. A natural scenario is a synchronous streaming model when the blocks are coming in one by one and the adversary on the channel sends across each modified blocks before the next block arrives.

NMC for $\mathcal{F}^\ell_{\mathsf{block}}$ is impossible. One can see that it is impossible to construct NMC against $\mathcal{F}^\ell_{\mathsf{block}}$ (for any ℓ): consider a tampering function, where the first $\ell - 1$ functions, $(f_1, \ldots, f_{\ell-1})$ are identity functions and the function f_ℓ (which gets the entire codeword as input) simply decodes the message and depending on the message, keeps it the same or overwrites it to something "invalid" (i.e., the modified codeword decodes to $\perp$). Note that, in this case the distribution of the (decoding of the) tampered codeword will indeed depend on the message, thereby violating non-malleability. In particular, such a tampering attack makes the decoder output $\perp$ with a probability distribution that depends on the input message. Therefore, we seek for a natural relaxation of the traditional definition of NMC such that it is achievable for the class $\mathcal{F}^\ell_{\mathsf{block}}$ and at the same time sufficient for interesting applications. In particular, we show that such relaxed NMC is sufficient to construct a simple non-malleable commitment scheme in a black-box manner. We note that the traditional application to tamper-resilient cryptography does not work with the relaxed version for obvious reason..

NMC with replacement (NMCwR). Essentially in the above attack the adversary breaks non-malleability by making the codeword "invalid". So, we take the most natural direction to relax the definition, in that the adversary is considered to be successful only if it produces some *valid and related* codeword via tampering. In particular, the adversary may selectively "destroy" a codeword depending upon the message we encode, however we show that in some sense, this is the "only attack" it can perform. Intuitively the guarantee provided by such an encoding scheme is that any adversary, by tampering with some encoded data can not produce a related encoded data without destroying it. However, formalizing such intuition turns out to be non-trivial. We take inspiration from the literature of non-malleable commitment w.r.t. replacement (introduced by Goyal [16]) and formalize such a relaxation by introducing an algorithm (possibly inefficient) called *replacer* which comes into play only when the tampered codeword is invalid, and in that case it replaces the $\perp$ by "anything" of his choice. Essentially, the idea is that if the invalidity depends on the input message (like

[3] In the computational setting, the functions f_i are assumed to run in polynomial time.

described in the above attack) then the replacer would rectify the output to remove such dependency. We call the new notion *non-malleable codes with replacement (NMCwR).*

1.1 Our results

In this paper we explore the properties, constructions and applications of NMCwR with respect to the class of block-wise tampering functions $\mathcal{F}^{\ell}_{\mathsf{block}}$. We call such code *block-wise non-malleable codes* (BNMC). Below we provide an overview of the results presented in this paper.

Information theoretic impossibility. Similar to the notion of continuous non-malleable codes [13](CNMC), any BNMC must possess a *uniqueness* property (a slightly different one than CNMC). For two blocks, uniqueness means that there can not exists two different valid codewords of the form (c_1, c_2) and (c_1, c_2') which decodes to different messages, i.e., for every valid code c_1, there is a unique decoding. If not, then an attack similar to the CNMC is possible without making the codeword invalid – the adversary can always tampers the first block to c_1 and depending on the message (since f_2 gets the entire codeword) tampers to one of c_2 or c_2' hence making the output distribution depend on the message. Consequently, just like CNMC, an information theoretic impossibility is evident with the only difference that in the setting of CNMC, the functions are unbounded, and, therefore (for two blocks) the function f_1 can derive the unique message corresponding to c_1 by brute-force and thus break the scheme. We show the following in the full version [4].

▶ **Theorem 1.** *It is impossible to construct an information-theoretic BNMC.*

Henceforth, in this paper we focus on constructing BNMC based on computational assumptions. We stress that even we are in the computationally bounded setting, we do not put any restriction on the efficiency of the replacer. In particular, the replacer is allowed to run in super-polynomial (or even exponential) time. In fact, later in this paper, we often encounter a replacer which runs in exponential time. Nonetheless, we must restrict the reduction to be probabilistic polynomial time (PPT). We are indeed able to overcome this technical hurdle by constructing such "efficient" reductions which can correctly simulate behavior of "highly inefficient" replacers.

Connection to Non-malleable Commitment. Since BNMC satisfies a definition weaker (that is NMC with replacement) than the traditional NMC, it is not possible to use such a code to build a tamper-resilient compiler as described in [12, 21] for obvious reason. In fact, it is nevertheless impossible to protect a system against memory tampering attack (see [8, 15, 18] for formal expositions on such attack) against any block-wise tampering. However we are able to show connections with non-malleable commitment with respect to opening (NMCom). To the best of our knowledge this is the first attempt to bridge these two non-malleability notions[4].

[4] In a recent work, Agrawal et al. [3] showed how to use NMC to construct non-malleable string-commitment from non-malleable bit-commitment. In their work, NMC is used as a tool, and, no relations are shown between non-malleable commitments and NMC. Recently, another work by Goyal et al. [17] constructs round-optimal NMCom from split-state NMC, the full version of which appears after the first version of this work.

1. Given an ℓ-block BNMC we can construct (in a black-box way) a simple $(\ell-1)$ round commitment protocol which is non-malleable with respect to opening (against synchronizing adversaries) as follows: the committer sends the block c_i in the i-th round and sends the last block c_ℓ as the opening. The receiver sends only acknowledgements after receiving each message. The non-malleability essentially follows from the non-malleability of the underlying BNMC and the perfect binding follows from the uniqueness property described above. To best of our knowledge, this is the first NMCom protocol where the receiver is not required to send any message (e.g. challenge) except for acknowledgement.

▶ **Theorem 2.** *Suppose there is an ℓ-block BNMC. Then there is a $(\ell-1)$ round perfectly binding non-malleable commitment scheme with respect to opening against a synchronizing man-in-the-middle adversary.*

2. We also show that from any non-interactive NMCom one can easily construct an BNMC even for only $\ell = 2$ blocks (i.e. optimal for $\mathcal{F}^{\ell}_{\mathsf{block}}$). Unfortunately, the only assumptions under which we know how to construct such commitments are either in the (non-tamperable) CRS model [9] or under the highly non-standard assumption of *adaptive* one-way functions [22]. Evidently this construction can not substitute our main construction which is based on much more standard assumption like sub-exponentially hard OWP.

▶ **Theorem 3.** *Suppose there is a perfectly binding non-interactive non-malleable commitment scheme (w.r.t. opening) whose input is a k-bit message and output is an n-bit commitment. Then there is a 2-block BNMC.*

Combining Theorem 2 and Theorem 3, we can conclude that when $\ell = 2$ the NMCom and BNMC are equivalent. The details of these results are elaborated in the full version [4].

Constructing BNMC. As the main result we provide a construction of BNMC from a standard assumption in the plain model. Precisely, we show that, for any arbitrary constant $\varphi > 0$, how to construct a BNMC against $\mathcal{F}^{\ell}_{\mathsf{block}}$ for $\ell = O(\kappa^{2+\varphi})$ (where κ is the security parameter). The security (i.e. non-malleability) of the construction is based on "sub-exponentially" hard one-way permutations which says that there exists one-way permutations (OWP) which are "hard-to-invert" even against an adversary running in sub-exponential time, precisely in time $O(2^{\kappa_{\mathsf{s}}})$ such that $\kappa_{\mathsf{s}} = O(\kappa^{\epsilon}/2)$ for some $0 < \epsilon < 1$. In particular, our construction uses any perfectly binding commitment scheme that is computationally hiding against such sub-exponential adversary (and this primitive can be constructed from the above assumption). The key technical challenge, as remarked earlier, is that BNMC is *not* an interactive primitive that allows bi-directional communication. This limitation renders the previously proposed techniques for designing non-malleable protocols inherently unusable. This is because these previous techniques are based on having "challenge-response" rounds similar to the type also used in designing zero-knowledge protocols. Thus, techniques like rewinding the sender are not useful in this setting at all: since there are no receiver messages, one would end up with the same transcript every time. Thus, apriori, it seems unclear what advantage one could get by having multiple blocks. Our final construction is quite clean and in fact, also gives arguably one of the simplest known constructions of non-malleable commitments. We show the following theorem.

▶ **Theorem 4.** *Assume the existence of sub-exponentially hard one-way permutations. Then for any $\varphi > 0$ of our choice, and any message length $k \in \mathbb{N}$, there exists an explicit construction of ℓ-block* BNMC *of codeword length $n = O(k\kappa^{6+\varphi})$, where $\ell = O(\kappa^{2+\varphi})$.*

We give an overview of the construction used to prove Theorem 4 in Section 1.2.

Strong BNMC. Additionally, we also consider a strictly stronger model of tampering: assume any permutation $\pi : [\ell] \to [\ell]$ chosen by the adversary. Then each function f_i takes i blocks $(c_{\pi(1)}, \ldots, c_{\pi(i)})$ as input and modifies the $\pi(i)$-th block. We call this family of function strong block-wise and denote it by $\mathcal{F}^{\ell}_{\mathsf{s\text{-}block}}$. We also provide a definition of strong BNMC which is essentially an explicit presentation of NMCwR for $\mathcal{F}^{\ell}_{\mathsf{s\text{-}block}}$. We provide an unconditional generic transformation to construct strong BNMC from any BNMC which, along with the earlier results imply that any construction of BNMC can be transformed to a strong BNMC (with some blow up in the length of codeword). We show the following, the details of which appears in the full version [4].

▶ **Theorem 5.** *If there is an ℓ-block BNMC with codeword length n, then there is an (ℓ)-block SBNMC with codeword length $\Theta(\ell n)$.*

1.2 Overview of our techniques

We now give a brief overview of our main construction of BNMC along with intuition as to why it works. The detailed construction is provided in Sec. 3.

First fix a parameter μ (such that $\mu = O(\kappa^{2+\varphi})$ for any arbitrary constant $\varphi > 0$ of our choice where κ is the security parameter) such that we encode a message m using $\ell = (2\mu + 1)$-blocks of codeword for some parameter μ . At a very high level, our encoding is as follows. Let us first fix some index (or *tag*) for the encoder $i \in [\mu]$. The encoder then chooses a *perfectly binding* commitment scheme COM.

Let $\mathsf{COM}_{\kappa_{\mathsf{s}}}(\cdot)$ and $\mathsf{COM}_{\kappa}(\cdot)$ denote that COM is computationally hidden with respect to security parameters κ_{s} and κ respectively, where κ_{s} is as mentioned above. The encoder then computes commitments to the message using $\mathsf{COM}_{\kappa_{\mathsf{s}}}$ and COM_{κ}. The first 2μ blocks of the encoding of m are blocks of all zeroes, except for block i and block $(2\mu - i)$ which are the commitments COM_{κ} and $\mathsf{COM}_{\kappa_{\mathsf{s}}}$, respectively. The $(2\mu + 1)^{\text{th}}$ block of the encoding contains the openings to $\mathsf{COM}_{\kappa_{\mathsf{s}}}$ and COM_{κ}. The decoding algorithm checks if (i) all the openings are consistent with the commitments and (ii) the messages committed are equal. Now, for a moment, assume that adversary's index i' is not equal to i (this can be removed later on). Then if $i' < i$, then the adversary has to output its first commitment without seeing the first commitment in the input codeword (rather only seeing on the string of zeros). Thus, the first commitment in the output is independent of the first commitment in the input. Moreover, our definition (NMCwR) puts the additional restriction that the adversary has to output a valid codeword in order to succeed. Combining one can see that the output codeword, if valid, must contain a message independent of the message encoded in the input. On the other hand, if $i' > i$, then the second commitment of the adversary has to be independent of the second commitment in the input. In this case, we rely on complexity leveraging to prove non-malleability. Using this key-observation one can prove the non-malleability except in one case: when the index chosen by the adversary i' is equal to i. To prevent mauling in this case we use one-time signatures. The encoder signs the entire codeword using i as a public-key and thus leaving the adversary either to forge the signature or change the index. However, one problem still remains. To use i as a public-key we need it to be sufficiently long, in particular for a concrete instance of such OTS (we consider variant of Lamport [19]) the length needed to be $O(\kappa^{2+\varphi})$ for any arbitrary constant $\varphi > 0$ of our choice. But note that, we have $i \in [\mu]$ and $\ell = 2\mu + 1$. Trying to set the size of the index $|i| = \log(\mu)$ to even $\Omega(k)$ would result in an "inefficient" construction with $\ell = 2^{\Omega(k)}$ blocks which is not acceptable. We solve this problem by using a "well-known" technique from non-malleable commitment, so-called DDN-XOR trick. Through that, it is possible to use a long tag of

size $t = O(\kappa^{2+\varphi})$ keeping the number of blocks also $O(\kappa^{2+\varphi})$ just by computing t shares (XOR's) of messages and and applying the above construction independently on the shares. So, our final construction would require a one-time signature which works with a public-key of bit-length $\mu = O(\kappa^{2+\varphi})$.

2 Definitions

We introduce the "relaxed" definition of non-malleable codes which is same as the NMC except there is a so-called *replacer* $\mathbf{R_f}$ which is an "all powerful" algorithm and comes into play only when the modified codeword is invalid (i.e. decodes to $\perp$). In that case, the replacer may replace the $\perp$ by any message in the message space or the symbol $\mathsf{same}^\star$. (The replacer can also keep the $\perp$ in case when it not harmful (i.e. does not depend on the input) e.g. when the tampering function always tampers to something invalid). Since the idea of replacer is similar in spirit with the notion of *non-malleable commitment with replacement* as introduced in [16] we call this relaxed version *non-malleable codes with replacement* (NMCwR in short). We present the formal definition below.

▶ **Definition 6** (Non-malleable codes with replacement)**.** Let $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ be an (k, n)-encoding scheme. Let $\mathcal{F}$ be some family of tampering functions. Then Code is called (k, n)-non-malleable code with replacement (NMCwR) if for every $f \in \mathcal{F}$ there exists an algorithm called the replacer $\mathbf{R_f}$ such that for any pair of messages $m_0, m_1 \in \{0,1\}^k$, $\mathsf{TampWR}^f_{m_0} \approx \mathsf{TampWR}^f_{m_1}$, where for any $m \in \{0,1\}^k$, TampWR^f_m is defined as

$$\mathsf{TampWR}^f_m \equiv \left\{ \begin{array}{c} c \leftarrow \mathsf{Enc}(m); c' \leftarrow f(c); \\ \text{If } c' = c \text{ set } m' := \mathsf{same}^\star \text{ else } m' \leftarrow \mathsf{Dec}(c') \\ \text{If } m' = \perp \text{ then } m' := \mathbf{R_f}(c) \ ; \ \text{Output: } m' \end{array} \right\},$$

where the randomness is over the encoding function Enc.

▶ Remark. Here, and everywhere in this section, the indistinguishability depends on the setting (information theoretic or computational). However, we emphasize that even if we are in the computationally bounded scenario, where the adversary is PPT, we do not restrict the replacer to be a PPT algorithm. This assumption is justified because the replacer is required only to establish the meaningfulness of the definition without affecting the natural intuition. Intuitively the purpose of the replacer is to relax the traditional notion in a way such that the tampering function is allowed to distinguish the tampering experiments, albeit only by making the codeword invalid. Nonetheless in the computational setting all the other algorithms involved as well as the the tampering functions are required to be PPT.

Some intuitions. We first provide some intuition behind why the above definition is meaningful. For every adversary, there is guaranteed to exist another adversary which always tampers in the same way as the original adversary, except, when the original adversary were to output an invalid codeword. In that case, the new adversary may employ any other (PPT) strategy. However when the original adversary outputs an invalid codeword, (in many applications) it could be considered as aborting or failing in those cases. Hence, our new adversary could be seen as strictly more powerful than the original one. However as the definition guarantee, the new adversary actually obeys the standard non-malleable code guarantee. Thus, in many scenarios, we believe the above weaker notion may be sufficient. Indeed, as shown in [16], the corresponding weaker notion for non-malleable commitments

(called non-malleability w.r.t. replacement) turns out to be sufficient for several applications including for obtaining constant round multi-party computation.

We now give the syntactic definition of *block-wise encoding scheme*.

▶ **Definition 7** (Block-wise encoding scheme)**.** Let $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ be an (k, n)-encoding scheme. Then it is called an (ℓ, k, n)-*block-wise encoding scheme* if each string output by Enc is an ℓ-tuple: $(c_1, \ldots, c_\ell)$ where $|c_i| = n_i$, with $\sum_{i=1}^{\ell} n_i = n$. Also let $\nu_i = \sum_{j=1}^{i} n_i$.

Next we define a property of such block-wise encoding scheme called *reveal index*, that will be useful later on.

▶ **Definition 8** (Reveal Index)**.** Let $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ be an (ℓ, k, n)-block-wise encoding scheme. Then Code is said to have reveal index η if $\eta - 1 \in [\ell]$ is the largest index for which the following condition holds: For all pair of messages $m_0, m_1 \in \{0,1\}^k$ if $(c_1^{(0)}, \ldots, c_\ell^{(0)}) \leftarrow \mathsf{Enc}(m_0)$ and $(c_1^{(1)}, \ldots, c_\ell^{(1)}) \leftarrow \mathsf{Enc}(m_1)$ then $(c_1^{(1)}, \ldots, c_{\eta-1}^{(1)}) \approx (c_1^{(1)}, \ldots, c_{\eta-1}^{(1)})$.

▶ Remark. This definition formalizes the fact that, for any encoding scheme, there is an index η which reveals some information about the encoded message for the first time in the sequence and the sequence $(c_1, \ldots, c_{\eta-1})$ before that does not reveal anything about the encoded message. Obviously $\eta \leq \ell$ for any block-wise encoding scheme.

Finally, we present our main definition of a *block-wise non-malleable encoding scheme* which is essentially an explicit presentation of NMCwR for the class $\mathcal{F}^{\ell}_{\mathsf{block}}$.

▶ **Definition 9** (Block-wise non-malleable codes)**.** Let $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ be an (ℓ, k, n)-block-wise encoding scheme. Let $\mathbf{f} = (f_1, \ldots, f_\ell)$ be any tuple of functions specified as follows: $\forall i \in [\ell], f_i : \{0,1\}^{\nu_i} \to \{0,1\}^{n_i}$. Then Code is called an (ℓ, k, n)-*block-wise non-malleable code* (BNMC in short) if, for any such tuple $\mathbf{f}$, there exists a replacer $\mathbf{R}_{\mathbf{f}}$, such that, for any pair of messages $(m_0, m_1) \in \{0,1\}^k$, the following holds: $\mathsf{BLTamp}^{\mathbf{f}}_{m_0} \approx \mathsf{BLTamp}^{\mathbf{f}}_{m_1}$. where $\mathsf{BLTamp}^{\mathbf{f}}_{m}$ for any $m \in \{0,1\}^k$ is defined as:

$$\mathsf{BLTamp}^{\mathbf{f}}_{m} = \left\{ \begin{array}{c} \mathbf{c} = (c_1, \ldots, c_\ell) \leftarrow \mathsf{Enc}(m); \\ \forall i \in [\ell] : c'_i = f_i(c_1, \cdots, c_i);\ \text{Let } \mathbf{c}' = (c'_1, \ldots, c'_\ell); \\ \text{If } \mathbf{c}' = \mathbf{c} \text{ then set } m' := \mathsf{same}^\star; \text{Else decode } m' \leftarrow \mathsf{Dec}(c'_1, \ldots, c'_\ell); \\ \text{If } m' = \perp \text{ then } m' \leftarrow \mathbf{R}_{\mathbf{f}}(c_1, \ldots, c_\ell);\ \text{Output } m' \end{array} \right\}.$$

▶ Remark. Our notion of block-wise non-malleable codes is identical to the notion of *look-ahead non-malleable codes* defined in the concurrent and independent work of Aggarwal *et al.* [1]. We choose to use the term block-wise as it is more appropriate in our setting.

3 Our Construction

In this section, we provide our main construction of a BNMC based on *sub-exponentially hard one-way permutations*. We construct the encoding scheme in three steps:

(i) In Sec 3.1 we begin by constructing a weaker BNMC that we call *tag-based block-wise non-malleable encoding scheme* (TBNMC). In such a code, every codeword has a *tag* associated with it and the tampering function must change the tag of a codeword in order to successfully maul a codeword. In other words, we allow an adversary to create a related codeword only when the tag remains the same. The tag used here is an index of the block and hence is only of size $\log(\kappa)$.

(ii) Then in Sec. 3.2 we use a technique, commonly known as the DDN-XOR trick [10], to construct a tag-based BNMC with tags of length $poly(\kappa)$.

(iii) Finally in Sec. 3.3 we construct an BNMC which achieves Def. 9, by using the public key of a *one-time signature scheme* as the tag of the above code, and by signing the entire codeword using the corresponding signing key.

3.1 Tag-based non-malleability

In this section we diverge from our original definition and construct an encoding scheme which meets a weaker definition of non-malleability, called tag-based non-malleability.

We define the tag to be always the first block of any codeword. A tag-based BNMC (TBNMC for short) is defined exactly as the same way as BNMC with the only difference that whenever the tag of the tampered codeword is equal to the tag of the original codeword, then the tampering experiment outputs $\mathsf{same}^\star$ even if there is any other modification. Clearly this is strictly weaker than BNMC. Please see the full version [4] for a formal definition.

Now we construct an encoding scheme which satisfies this weaker definition based on sub-exponentially hard OWP. The proof uses complexity leveraging which essentially forces us to assume sub-exponential hardness as opposed to standard (super-poly) hardness.

We assume that sub-exponentially hard OWP exist that are considered to be hard to break even if the adversary is allowed to run in sub-exponential time, namely in $O(2^{\kappa_{\mathsf{s}}})$ such that $\kappa_{\mathsf{s}} = \kappa^{\epsilon}/2$ (recall that κ is the security parameter) for some constant $\epsilon \in (0, 1)$. The proof crucially relies on this as it uses one level of complexity leveraging. In particular, while reducing to such OWP, we assume that the adversary (the reduction in this case) is unable to break the one-way permutation (the hiding of a commitment scheme in this case) even when it is allowed to run in time $O(2^{\kappa_{\mathsf{s}}})$ (but in time $o(2^{\kappa})$).

We use a *non-interactive commitment*, Com, that is *perfectly binding*. We write $\mathsf{Com}_{\kappa_{\mathsf{s}}}$ and Com_{κ} to denote the commitment scheme has *computational hiding* with the security parameters κ_{s} and κ, respectively. In particular, Com_{κ} is a computationally hiding commitment scheme even against an adversary running in $O(2^{\kappa_{\mathsf{s}}})$ time. Suppose that such commitment scheme, on input some bit-string of length $k \in \mathbb{N}$, outputs commitments of length $\mathsf{p}(\kappa, k)$ where $\mathsf{p}(\cdot) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a fixed polynomial (determined by the specifications of the commitment scheme) in security parameter. We stress that such commitments can be constructed from *sub-exponentially hard one-way permutations*.

First we give a brief overview of the construction. Let $\mu \in \mathbb{N}$ be a parameter. We will now construct a TBNMC with ℓ blocks where $\ell = 2\mu + 2$. For now, assume ℓ to be a even number. Now for any tag $\mathsf{tg} \in [\mu]$ we construct the encoding scheme as follows: we put strings of 0 in all the blocks except the four "special" blocks: the first block is set to tg, the $(\mathsf{tg}+1)$-th block is set to the "bigger" commitment $\mathsf{Com}_{\kappa}(m)$, the $(\ell - \mathsf{tg})$-th block is set to the "smaller" commitment $\mathsf{Com}_{\kappa_{\mathsf{s}}}(m)$ and the ℓ-th (and final) block is set to the openings of the commitments. Now, for odd ℓ, one can just append one dummy block (string of 0's) right before the final block. So, without loss of generality we would assume ℓ to be even in this section. The detail construction is presented in Fig. 1. Note that here the blocks are of different length. However, it is easy to convert the code with equal block-length by padding additional zeros. We keep it without such padding for simplicity. Also, note that, from the computational hiding property of the commitment scheme, it follows that the construction has reveal index $\ell = 2\mu + 2$ for any PPT adversary.

The following theorem states that the construction is a TBNMC. The proof can be found in the full version [4].

▶ **Theorem 10.** *Let $\mu \in \mathbb{N}$ be some parameter. Assume that sub-exponentially hard one-way-permutations exists. Then, for any tag $\mathsf{tg} \in [\mu]$ and any $k \in \mathbb{N}$, the encoding scheme*

$\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ *described in Fig. 1 is a* $(\mathsf{tg}, \ell, k, n)$-TBNMC *against all* PPT *adversary such that* $n = O(k + \mu \cdot \mathsf{p})$ *and* $\ell = 2\mu + 2$.

3.2 Non-malleability amplification

In this section we extend our construction to an efficient construction which can support larger tags. This extension is similar to a well-known phenomenon, namely *non-malleability amplification* [20], in the non-malleable commitment literature using the DDN-XOR trick [10].

3.2.1 One-many non-malleability

Towards that, we first show that the construction given in Fig. 1 already satisfies a stronger notion, which we call *one-many* tag-based non-malleability (OMTBC). This definition (we refer to the full version [4] for a formal definition), informally states that an adversary that is able to tamper a single codeword of m, cannot even come up with a set of codewords such that one of them is related to m. In particular, each function f_i in the tuple $\mathbf{f} = (f_1, \ldots, f_\ell)$ has much larger range than the domain and produces many c_i's together with the knowledge of the first i blocks of the input codeword.[5] Our next theorem shows that our construction (Fig. 1) achieves this stronger definition.

▶ **Theorem 11.** *Let* $\mu, t \in \mathbb{N}$ *be some parameter. Assume that sub-exponentially hard one-way-permutations exists. Then, for any tag* $\mathsf{tg} \in [\mu]$ *and any* $k \in \mathbb{N}$ *the* $(\mathsf{tg}, \ell, k, n)$-TBC $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ *described in Fig. 1 is an* $(t, \mathsf{tg}, \ell, k, n)$*-one-many tag-based BNMC against all* PPT *adversary such that* $n = O(k + \mu \cdot \mathsf{p})$ *and* $\ell = 2\mu + 2$.

3.2.2 Using DDN-XOR trick

In this section we use the DDN-XOR trick to construct an "efficient" TBNMC with "large" tags. Let us start with some intuitions. The construction uses any OMTBC (called "inner code" in the following) with "small" tag in a black-box way. The basic idea is as follows: let the "big" tag TG be t-bit long. Then compute t shares of message m just using XOR's i.e. $(m_1, \ldots, m_t)$ which is nothing but a t-out-of-t secret sharing. Then encode each m_j with the inner code using $j \| \mathsf{TG}[j]$ (which is of $O(\log(t))$-size) as tag. Finally put the encodings in increasing order of j (from 1 to t). The first block of the final codeword is, by definition the tag TG. the second block would consist of the first t blocks of inner codes in order and so on. The key-intuitions why the construction works are as follows. In order to break the tag-based non-malleability of the final encoding (called "outer code" within this sub-section), the adversary must produce a valid codeword with different"big" tag $\widetilde{\mathsf{TG}} \neq \mathsf{TG}$. In that case, evidently, there must exist at least one index $j \in [t]$ where the "small" tags differ $\widetilde{\mathsf{tg}}_j \neq \mathsf{tg}_j$. Moreover notice that, the adversary can't copy tg_j to any other position than j as that would result in an invalid codeword. Therefore $\mathsf{tg}_j = j \| \mathsf{TG}[j]$ is different from *all* the "small tags" of the tampered inner codewords. Then we reduce to the one-many non-malleability of the inner code in first such position (say $j^\star$). In particular, if the adversary tampers with the $j^\star$-th inner code, then by one-many non-malleability of the "inner code" no tampering

[5] We note that Chattopadhyay et al. [5] introduced the notion of one-many non-malleable code which is in turn built on continuous non-malleable code [13](CNMC). It is important not to confuse this notion with CNMC where the adversary chooses each subsequent tampering function after observing the result of the previous tamperings.

Parameters: Let Com_{κ_s} takes a k-bit message as input and u_s-bit randomness to produce a v_s-bit commitment and Com_{κ} takes a message of the same length, but randomness of u-bit to produce a v-bit commitment[a]. Let $\mathsf{tg} \in [\mu]$ be the tag of the encoding scheme for some $\mu \in \mathbb{N}$. We define a $(\mathsf{tg}, \ell, k, n)$-block-wise encoding scheme where $\ell = 2\mu + 2$ and $n = k + u_s + u + \mu(v_s + v) + \lfloor \log \mu \rfloor + 1$ as follows:

Encoding $\mathsf{TEnc}(m)$**:** The encoder gets a message $m \in \{0,1\}^k$ as input and do as follows:

1. Initialize: Choose randomnesses $r_s \xleftarrow{\$} \{0,1\}^{u_s}$ and $r \xleftarrow{\$} \{0,1\}^u$ for commitment scheme. Set the first block $c_1 := \mathsf{tg}$.
2. Stage-1: For all $i \in \{2, \ldots, \mu + 1\}$, define the i-th block of codeword c_i as follows:

$$c_i := \begin{cases} 0^v & i \neq \mathsf{tg} + 1 \\ \mathsf{Com}_{\kappa}(m, r) & i = \mathsf{tg} + 1 \end{cases}$$

3. Stage-2: For all $i \in \{\mu + 2, \ldots, 2\mu + 1\}$, define the i-th block of codeword c_i as follows:

$$c_i := \begin{cases} 0^{v_s} & i \neq 2\mu + 2 - \mathsf{tg} \\ \mathsf{Com}_{\kappa_s}(m, r_s) & i = 2\mu + 2 - \mathsf{tg} \end{cases}$$

4. Final stage: Define the last block as the decommitments i.e. the message and the randomnesses in the order of commitments are sent: $c_{2\mu+1} := (m, r, r_s)$.

Decoding $\mathsf{TDec}(\mathbf{c})$**:** On receiving a codeword $\mathbf{c}$ parse it as $\mathbf{c} = (c_1, \ldots, c_{2\mu+2})$ such that $|c_1| = \lfloor \mu \rfloor + 1$, for $i \in \{2, \ldots, \mu + 1\}$, $|c_i| = v$, for $i \in \{\mu + 2, \ldots, 2\mu + 1\}$, $|c_i| = v_s$ and for $i = 2\mu + 2$, $|c_i| = k + u_s + u$. Then do as follows:

1. Correctness of Structure: First check if the structure is correct: that is if $c_1 \neq 0$ and there are exactly two indexes $i_1 \in \{2, \ldots, \mu + 1\}$, $i_2 \in \{\mu + 2, 2\mu + 1\}$ such that:
   - **a.** $c_{i_1} \neq 0^v$ and $c_{i_2} \neq 0^{v_s}$.
   - **b.** for all other indexes $i \in \{2, \ldots, \mu+1\} \backslash \{i_1\}$, $c_i = 0^v$ and $i \in \{\mu+2, \ldots, 2\mu+1\} \backslash \{i_2\}$, $c_i = 0^{v_s}$.
   - **c.** $i_1 + i_2 = 2\mu + 1$.

   if any of them fails, then the structure of the tampered codeword is incorrect and therefore output $\perp$, else go to the next step.
2. Consistency of commitment: Parse $c_{2\mu+2}$ as $(m, r, r_s) := c_{2\mu+2}$ such that $|m| = k$, $|r| = u$ and $|r_s| = u_s$. Then check the validity of the commitment-decommitment pair $(c_{i_1}, (m, r))$ and $(c_{i_2}, (m, r_s))$, if any of them are invalid output $\perp$, otherwise output the committed message m.

[a] We assume $|v_s|, |v| = poly(\kappa)$

Figure 1 The construction of $(\mathsf{tg}, \ell, k, n)$-TBNMC for tag size $\log \kappa$.

function would not be able to succeed in producing any valid inner codeword that encodes a value which is "related" to the $j^\star$-th original share. Clearly, this implies the entire tampered outer codeword would have no information about $j^\star$-th share which makes the encoded massage (if valid) completely unrelated to the original message by the property of secret sharing.

For any tag $\mathsf{TG} \in \{0,1\}^t$ we construct a $(\mathsf{TG}, \ell', k', n')$-TBNMC $\mathsf{LCode} = (\mathsf{LEnc}, \mathsf{LDec})$ from a $(t, \mathsf{tg}, \ell, k, n)$-OMTBC $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ for any $\mathsf{tg} \in \{0,1\}^\alpha$ such that $t = 2^{\alpha-1} - 1$, $\ell' = \ell + 1$, $k' = k$ and $n' = nt$ as follows.

- **Encode** $\mathsf{LEnc}(m)$:
  1. Secret-sharing: On receiving an input message $m \in \{0,1\}^{k'}$, first choose $(t-1)$ random k'-bit strings $(m_1, \ldots, m_{t-1})$ and then compute $m_t = m \oplus m_1 \oplus \cdots \oplus m_{t-1}$. Note that the tuple $(m_1, \ldots, m_t)$ represents a (t,t)-secret sharing of m.
  2. Encode using smaller tag: Then for each $j \in t$, let the j-th "smaller" tag be $\mathsf{tg}_j = \mathtt{BIT}(j) \| \mathsf{TG}[j]$. Then compute the encoding of m_j as: $(c_{1,j}, \ldots, c_{\ell,j}) \leftarrow \mathsf{TEnc}_{\mathsf{tg}_j}(m_j)$.
  3. Constructing blocks: Define the tag-block $c_0 := \mathsf{TG}$. For all $i \in [\ell]$ define the i-th block as $c_i := (c_{i,1}, \ldots, c_{i,t})$. Output the codeword $\mathbf{c} = (c_0, \ldots, c_\ell)$.
- **Decode** $\mathsf{LDec}(\mathbf{c})$:
  1. Parsing: On receiving a codeword $\mathbf{c}$, parse it as $(c_0, \ldots, c_\ell) := \mathbf{c}$ such that $|c_0| = t$ and for all $i \in [\ell]$ $|c_i| = tn_i$. Then, for all $i \in [\ell]$ parse c_i as $(c_{i,1}, \ldots, c_{i,t})$ such that for all $j \in [t]$, $|c_{i,j}| = n_i$.
  2. Checking Tag consistency: Check if the "bigger" tag is consistent with the "smaller" tag: $c_0 = c_{1,1}[\alpha] \| c_{1,2}[\alpha] \| \cdots \| c_{1,t}[\alpha]$. Also check if the positions of the smaller tags are correct: $\forall\ j \in [t]$, $c_{1,j}[1 \ldots (\alpha-1)] = \mathtt{BIT}(j)$. If any of these fail output $\perp$, otherwise go to the next step.
  3. Decoding with smaller tag: For each $j \in [t]$ decode each value $v_j \leftarrow \mathsf{TDec}_{\mathsf{tg}_j}(c_{1,j}, \ldots, c_{\ell,j})$. If any of them is $\perp$ then output $\perp$. Otherwise, parse each v_j as m_j and finally output $m = m_1 \oplus \cdots \oplus m_t$.

Formally we show the following theorem. The proof can be found in the full version [4].

▶ **Theorem 12.** *Let* $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ *be a* $(t, \mathsf{tg}, \ell, k, n)$-OMTBC *for any tag* $\mathsf{tg} \in \{0,1\}^\alpha$, $t = 2^{\alpha-1} - 1$ *and* $k \in \mathbb{N}$. *Then for any tag* $\mathsf{TG} \in \{0,1\}^t$ *the above construction* $\mathsf{LCode} = (\mathsf{LEnc}, \mathsf{LDec})$ *is a* $(\mathsf{TG}, \ell', k', n')$-TBNMC *for* $\ell' = \ell + 1$, $k' = k$ *and* $n' = nt$.

3.3 The full construction by removing tags

Finally we present a transformation to remove tags using one-time signature scheme and a tag-based code with "large tag" (will be referred to as "inner code" in this section). This is similar to a standard trick [10] used in the area of non-malleable commitment for the same purpose. The main idea is to sign the entire codeword and set the public-key as the tag. This forces the tampering function either to keep the tag same and forge the signature in order to tamper, otherwise change the tag by producing its own key-pairs and then tamper. But the "inner code" guarantees that whenever the tag is changed, the tampering would result in an "unrelated" codeword.

Let $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ be an $(\mathsf{tg}, \ell, k, n)$-TBNMC for any tag $\mathsf{tg} \in \{0,1\}^t$. Let $\mathsf{OTSig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-time signature scheme with public key $pk \in \{0,1\}^t$ which takes any $k_m = n - t$-bit message to produce a n_s-bit signature. Then we construct an $(\ell, k, n + n_\mathsf{s})$-BNMC $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ as follows:

- **Encode** $\mathsf{Enc}(m)$:
  1. GENERATE SIGNATURE KEYS: On input message $m \in \{0,1\}^k$ first run the key-generation algorithm of the signature scheme OTSig to generate a key pair: $(pk, sk) \leftarrow \mathsf{KGen}(1^\kappa)$.
  2. ENCODE WITH TAG: Run the tag-based encoding scheme with pk as the tag on the input message m to produce the codeword $(\tilde{c}_1, \ldots, \tilde{c}_\ell) \leftarrow \mathsf{TEnc}(m)$. Note that $\tilde{c}_1 = pk$.
  3. SIGN THE CODEWORD: Sign the codeword (except the tag) $(\tilde{c}_2, \ldots, \tilde{c}_\ell)$ to compute the signature $\sigma \leftarrow \mathsf{Sign}(sk, (\tilde{c}_2, \ldots, \tilde{c}_\ell))$.
  4. OUTPUT: Set for all $i \in [\ell - 1]$, $c_i = \tilde{c}_i$ and $c_\ell = \tilde{c}_\ell \| \sigma$. Output the codeword $\mathbf{c} = (c_1, \ldots, c_\ell)$
- **Decode** $\mathsf{Dec}(c_1, \ldots, c_\ell)$:
  1. PARSE: On input the codeword $(c_1, \ldots, c_\ell)$, set $\forall i \in [\ell - 1]$, $\tilde{c}_i := c_i$ and parse c_ℓ as $(\tilde{c}_\ell \| \sigma) := c_\ell$ such that $|\tilde{c}_\ell| = n_\ell$ and $|\sigma| = n_{\mathsf{s}}$.
  2. VERIFY SIGNATURE: Then verify the signature $d \leftarrow \mathsf{Verify}\,(\tilde{c}_1, (\tilde{c}_2, \ldots, \tilde{c}_\ell), \sigma)$. If $d = 0$ (i.e. verification fails) then output $\perp$. Otherwise go to the next step.
  3. DECODE WITH TAG: Decode the codeword as $\widetilde{m} \leftarrow \mathsf{TDec}(\tilde{c}_1, \ldots, \tilde{c}_\ell)$. Output $\widetilde{m}$.

Formally, we have the following theorem (see the full version [4] for a formal proof).

▶ **Theorem 13.** *Let* $\mathsf{TCode} = (\mathsf{TEnc}, \mathsf{TDec})$ *be a* $(\mathsf{tg}, \ell, k, n)$*-TBNMC for any tag* $\mathsf{tg} \in \{0,1\}^t$ *and* $\mathsf{OTSig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ *be a one-time signature scheme with public key* $pk \in \{0,1\}^t$ *which takes any* $k_m = n - t$*-bit message to produce a* n_{s}*-bit signature.Then the above construction* $\mathsf{Code} = (\mathsf{Enc}, \mathsf{Dec})$ *is a* (ℓ', k', n')*-BNMC for* $\ell' = \ell$, $k' = k$ *and* $n' = n + n_{\mathsf{s}}$.

Theorem 4 now follows by combining Theorem 11, Theorem 12 and Theorem 13 and instantiating with appropriate parameters.

References

1 Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 459–468. ACM, 2015.

2 Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 774–783. ACM, 2014.

3 Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *Advances in Cryptology – CRYPTO 2015 – 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, pages 538–557, 2015. doi:10.1007/978-3-662-47989-6_26.

4 Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. *IACR Cryptology ePrint Archive*, 2015:129, 2015. URL: http://eprint.iacr.org/2015/129.

5 Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. *To Appear in* STOC *(full version available at arXiv:1505.00107)*, 2016.

6 Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 306–315. IEEE, 2014.

7 Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *Theory of Cryptography*, pages 440–464. Springer, 2014.

8 Ivan Damgård, Sebastian Faust, Pratyay Mukherjee, and Daniele Venturi. Bounded tamper resilience: How to go beyond the algebraic barrier. In *ASIACRYPT (2)*, pages 140–160, 2013.

9 Ivan Damgard and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 426–437. ACM, 2003.

10 Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM review*, 45(4):727–784, 2003.

11 Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In *CRYPTO (2)*, pages 239–257, 2013.

12 Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452, 2010.

13 Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *Theory of Cryptography*, pages 465–488. Springer, 2014.

14 Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In *Advances in Cryptology – EUROCRYPT 2014*, pages 111–128. Springer, 2014.

15 Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *Theory of Cryptography*, pages 258–277. Springer, 2004.

16 Vipul Goyal. Constant round non-malleable protocols using one way functions. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 695–704. ACM, 2011.

17 Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. *IACR Cryptology ePrint Archive*, 2015:1178, To appear at STOC 2016. URL: `http://eprint.iacr.org/2015/1178`.

18 Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In *CRYPTO*, pages 373–390, 2011.

19 Leslie Lamport. Constructing digital signatures from a one-way function. In *Technical Report SRI-CSL-98*. SRI International Computer Science Laboratory, 1979.

20 Huijia Lin and Rafael Pass. Non-malleability amplification. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 189–198. ACM, 2009.

21 Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO*, pages 517–532, 2012.

22 Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In *Advances in Cryptology – CRYPTO 2008*, pages 57–74. Springer, 2008.

Provably Secure Virus Detection: Using The Observer Effect Against Malware*

Richard J. Lipton[1], Rafail Ostrovsky[†‡2], and Vassilis Zikas[§‡3]

1 **Department of Computer Science, Georgia Institute of Technology, Atlanta, USA**
rjl@cc.gatech.edu

2 **Department of Computer Science & Department of Mathematics, University of California, Los Angeles, USA**
rafail@cs.ucla.edu

3 **Department of Computer Science, Rensselaer Polytechnic Institute, Troy, USA**
vzikas@cs.rpi.edu

Abstract

Protecting software from malware injection is one of the biggest challenges of modern computer science. Despite intensive efforts by the scientific and engineering community, the number of successful attacks continues to increase.

This work sets first footsteps towards a provably secure investigation of malware detection. We provide a formal model and cryptographic security definitions of attestation for systems with dynamic memory, and suggest novel provably secure attestation schemes. The key idea underlying our schemes is to use the very insertion of the malware itself to allow for the systems to detect it. This is, in our opinion, close in spirit to the quantum Observer Effect. The attackers, no matter how clever, no matter when they insert their malware, change the state of the system they are attacking. This fundamental idea can be a game changer. And our system does not rely on heuristics; instead, our scheme enjoys the unique property that it is proved secure in a formal and precise mathematical sense and with minimal and realistic CPU modification achieves strong provable security guarantees. We envision such systems with a formal mathematical security treatment as a venue for new directions in software protection.

1998 ACM Subject Classification D.4.6 Security and Protection (K.6.5)

Keywords and phrases Cryptography, Software Attestation, Provable Security

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.32

1 Introduction

Protecting software from malware injection is a major goal of computer security. Nonetheless, the problem of basing solutions on strong theoretical foundations does not seem to have received sufficient attention in the malware protection literature. In this work we suggest a novel provably secure and practically efficient paradigm for software protection against

* A preliminary version of this work was presented at the Securing Computation Workshop, Simons Institute, UC Berkeley. US Patent Pending [22].

† This author was supported in part by the NSF award CNS-1136174.

‡ Work done in part while the author was visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467.

§ This author was supported in part by the Swiss NSF Ambizione grant PZ00P2_142549.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 32; pp. 32:1–32:14

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

arbitrary malicious code injection. We provide the first formal cryptographic model tailored to detect malware injection in modern computers, which we envision as the basis for fruitful research on provable secure detection and prevention of malware leading into the next generation of secure systems. Importantly, we provide first matching solutions that are accompanied by mathematical proofs that any memory oblivious injection will be caught with arbitrary high probability. Our solutions are practically efficient, demonstrating that provable security does not come at a too high price.

Our proposed system has the following desirable properties: 1) it requires minor to no changes to the CPU specification to provably defend against injection attacks that can not read the code before they inject their malware; 2) it only affects the performance of the program by a modest amount. The first property implies that it can be used on today's computers. The second property means that it can be practical, since performance is critical in most applications; this is obvious even in its simplified form presented here, where various optimizations have been avoided for the sake of clarity in the presentation. Finally, we allow attacks both spatial and temporal freedom: the attack can occur at any time during the execution and on any part of the memory. It can attack code as well as data.

1.1 Overview of our Contributions

We put forward and formally define the notion of a *virus detection scheme* (in short, VDS) which compiles any given program W (and its data) into a new secured program $\widetilde{W}$ that performs the same computation as W but allows us to detect any virus injected in the memory at any point of the execution path. The detection is done via a provably secure challenge-response mechanism between the machine executing the (compiled) software and a verifying external device. Importantly, we insist that the verification algorithm be very simple, and in particular that it be executed by a very lightweight device (Our constructions require that the verifier only does an encryption and compares strings for equality.) Thus, the role of the verifier can be, for example, played by the user with a smartphone, or by a compact and simple, intrusion-free hardware module that could even be part of the CPU.[1] Moreover, the verification uses public key techniques, where the verifier needs only the public key. This restricts the attack surface to the machine executing the secured code, as compromising the verifier's privacy gives an attacker no advantage in breaking our scheme.

We provide our formal cryptographic definitions of VDS security based on the well-studied Random Access Machine (RAM) model, slightly adapted to be closer to an abstract version of a modern computer following the *von Neumann architecture*. The suggested model corresponds to a simplistic *closed-system* abstraction of software execution: The code to be executed along with its associated data is loaded initially on the random access memory (RMEM,) which, through the execution, only communicates with the CPU.[2]

In more detail, a secure VDS is described as follows. The software to be executed is compiled, prior to being loaded onto the memory, to a secure version. Importantly, this compilation does not necessarily need the source code but for a wide class of software (non-self-modifying code) can be applied on the binary (machine-language) code of the program; thus, the compilation can be performed by the program vendor (this might allow for further

[1] Unlike software-based attestation techniques (cf. Section 1.2), the verification in our scheme can be done even over an insecure network, e.g. the Internet.

[2] In particular, the current theoretical model does not address how the data is exchanged with secondary storage devices, e.g., flash memory. It is part of ongoing research to extend this model to more complicated architectures.

efficiency optimizations) or by the user without requiring any reverse engineering. The compiler, in addition to the (binary code) of the program and its data, uses a randomly chosen *compilation key* K.

To check/verify that the software running on a system has not been attacked by a virus (or to detect such an attack in case it has occurred), the following *detection process* is executed. The user issues a challenge c that depends on the compilation key K and the system is required to produce a reply which passes a specified verification test. As already discussed, we arrange things in such a way that the verifier does not need to know the compilation key, and he can just know some public information on it. Formally, this is done by the use of public key cryptography, where the compilation key is the secret key, and the information held by the verifier, which we refer to as the *verification key*, is the corresponding public key. The security of the VDS requires that if the RAM has not been attacked then it can always reply to the challenge in an accepting manner (verification correctness); otherwise, any reply will be rejected with high probability (security).

We provide concrete instantiations of practical VDSs which, depending on the assumptions about the CPU they are executed on, achieve from a reasonably strong security (namely protection against continuous injection of moderate-sized virus) to security against arbitrary small viruses injected on locations that might depend on key-share locations. Informally, we prove the following result for our VDSs.

▶ **Theorem** (informal). *Under standard complexity assumptions our VDSs compile any non-self-modifying software into a secure version that detects any malware injection with very high probability.*

Our schemes are independent of the virus code, are platform-independent, and might require only small modifications to the common CPU architecture (in particular, they do not assume tamper-resilient hardware); and, with appropriate optimizations we can make it that they only affect the performance of the executed software by a practically acceptable amount. As a useful side effect, our scheme is even able to detect hardware errors, e.g., random bit-flips in the memory; as demonstrated in [5], such errors might have important consequences on the security of the executed software.

To construct our VDS we start by constructing a scheme satisfying a weak(er) notion of security, which (1) catches only injections of continuous and sufficiently long viruses, and (2) assumes that the response is computed by an external (trusted) device that is given access to the state of the attacked system. We give a formal definition of this weaker notion which depending on the application and the desired security level might already be satisfactory. We then proceed and gradually modify (strengthen) our scheme. The first modification ensures that the compiled program can compute the response by itself, i.e., without the help of an external trusted device. We note that the security of our weakest scheme is information theoretic (it is based on the perfect privacy of one-time pad encryption), whereas the proof of the more secure scheme requires a leakage-resilient encryption algorithm [9].

The second modification uses a simple *message authentication code* (in short, MAC) to remove the limitation to sufficiently long and continuous virus injection. A MAC authenticates a value using a random key, so that an attacker without access to the key cannot change the value without being detected. More concretely, to authenticate a word w our scheme relies on the standard MAC [16] that uses two keys K_1 and K_2, and compute a MAC tag as $t = w \cdot K_1 + K_2$, where w, K_1, and K_2 are interpreted as elements of an appropriate large arithmetic field. Observe that such arithmetic operations are implemented in hardware in existing CPUs. We assume, however, that the executing CPU has a slightly extended instruction-set which, informally, has each "load" instruction, i.e., read-from-memory instruction, check that the

MAC is correct before it loads the word on the CPU registers. We stress that this only needs the architecture to provide us with a very simple extra piece of microcode. By engineering the CPU's hardware in a smart way so that these operations are done on the circuit level, we expect to be able to reduce the effect of our compiler to a barely noticeable slowdown. Fortunately, the new architecture that Intel recently announced promises to embed such microcode in its next-generation microchips [24].

Overview of our VDS Construction. So how can we detect an intrusion that we have never seen before? How can we arrange that any injection of malware into our program by an attacker who does not know the compilation-key will be detected? The answer is that we will hide the compilation-key/secret in our program in a way that ensures that with very high probability, any injection by an adversary must destroy the secret. Once this secret is destroyed, the adversary may indeed be able to take over the program, but will be quickly detected. We note that current systems may fail to detect such an attack forever – for the type of injection we protect against, we are able to detect it in seconds.

A point: We can imagine situations where even such a swift detection of an attack is not sufficient to stop potential harm. An attacker could, in some situations, do immense damage even if he is in control of a program for only a fraction of a second. Furthermore, our approach is not targeted towards fixing buggy software or detecting malware-including software which might be (unknowingly) installed by the user. Rather, our method offers protection against "unauthorized" injection of malicious code, e.g., direct injection to the memory or radiation attacks. But we view the ability to provably detect any such attack, new, old, clever, or not, as a big improvement over the current state of the art.

So how can we do this? Let W be a program that we wish to protect from malware. We recompile W to $\widetilde{W}$. The idea is that W and $\widetilde{W}$ must compute the same thing, run in about the same time, and yet $\widetilde{W}$ must be able to detect itself against the insertion of malware. An obvious idea is to have $\widetilde{W}$ periodically check to see if its code or data have been maliciously changed. The trouble with this approach is that the attacker could easily inject his own code, take over the checker, and thereby disable the checking. In short: who checks the checker?

Here is how we proceed. The protected program $\widetilde{W}$ operates normally most of the time. Periodically it is challenged by another machine to prove that it has a secret key K. This key is known only to $\widetilde{W}$ and not to the attacker. If $\widetilde{W}$ has not been attacked, then it simply uses the key K to answer the challenge and thus proves that it is still operating properly.

The issue is what happens if $\widetilde{W}$ has been attacked and some code has been injected into it. We can arrange $\widetilde{W}$ so that no matter how the injection of code has happened the key K is lost. The attacker's very injection will have changed the state of $\widetilde{W}$ so that it is now in a state that no longer knows the key K. This is the analog of the quantum observer effect, often referred to as the Heisenberg Uncertainty Principle: the attacker has damaged the state of $\widetilde{W}$ so that the key has been destroyed.

Making the attack destroy the key is the central idea here. If $\widetilde{W}$ simply stores K somewhere in memory, the attacker will take over the program, look around for the key, and likely find the key;[3] this defeats the protection, since now it can answer the periodic challenges just like $\widetilde{W}$ could. To resolve the above issue we distribute the key K all through memory by using what is called *secret sharing* [30]. This is a standard method in cryptography that breaks a small key, like K, into many small pieces, called shares. These pieces are cleverly placed throughout all of $\widetilde{W}$'s memory. Our secret sharing allows K to be reconstructed only

[3] Note that we do not assume private memory, i.e., memory which the CPU cannot read.

if all the pieces are left untouched – if any are changed in any way, then it is impossible to reconstruct the key. Obviously, if there has been no attack, then in normal operation $\widetilde{W}$ can reconstruct the key from the pieces and answer the challenges. However, if the pieces are not all intact, then $\widetilde{W}$ will be unable to answer the next challenge.

This is the high-level idea of our method: hide the key via secret sharing, and rely on the attacker to destroy at least one share of the key. This destruction is irreversible and makes the attack fail the next challenge.

There is one additional point that we must mention. We must arrange that $\widetilde{W}$ can be attacked at anytime, including when the system has collected all the shares and reconstructed the key K on its CPU. This is a very dangerous time, since if $\widetilde{W}$ is attacked at this moment the fact that shares are destroyed does not matter. The attacker can simply use the reconstructed key K. We avoid such *time-of-check-time-of-use* (TOCTOU) attacks [25] by actually using two keys K_1 and K_2 in tandem. Both are stored via secret sharing as before, but now one must have both in order to answer the challenges. We arrange that $\widetilde{W}$ only reconstructs one key at a time and this means that an attacker must destroy either or both of the keys. This handles the above dangerous situation and makes our method provably secure.

Tying up the last loose end, we treat the case of very small viruses – i.e., ones that consist of a single bit or just a few bits – and viruses that know (or guess) the exact memory locations of key-shares in advance and therefore might not need to overwrite them. We armor our system to defend even against such tiny/informed viruses by employing an additional lightweight cryptographic primitive, namely a MAC. We use the MACs in a straightforward and efficient manner: we authenticate each word in W using the key-shares (also) as MAC keys, and require that for any word that is loaded from the random access memory, the CPU verifies the corresponding MAC before further processing the word. Thus, if the MAC does not check out we detect it (and immediately destroy the share), and if the adversary changes the MAC key, he loses one of the shares of one of the two secrets, and we detect that as well.

We enforce the above checks by assuming a modified CPU architecture (with only very small amount of additional computation, so the CPU power consumption is not affected). More specifically, to get the most out of the MAC functionality we need that when the CPU executes the program, it verifies authenticity of the words it loads from the memory. That is, the load operation of the CPU loads a block of several (in our case, five) words – this is already the case in most modern CPUs – including the program word, the MAC-tag and the corresponding keys, and check with every load instruction that the MAC verifies before further processing the word. Importantly, our MAC uses just one field addition and one field multiplication per authentication/verification; the circuits required to perform these operation are actually trivial compared to the modern CPU complexity and are part of many existing CPU specifications.

But to obtain security against any injection, we need one more trick: instead of using the actual key-shares in the generation/verification of the MAC tag, we use their hash-values. This ensures that the virus cannot manipulate the keys and forge a consistent MAC unless he overwrites a large part of them. We anticipate that given our goal's potential impact on security, such functionality might be included in the next generations of CPUs [24].

1.2 Related Literature

The literature on defense mechanisms against malware-injection attacks is vast. Due to its urgency and importance, the problem has been intensively researched. In what follows we provide a brief overview of the directions most related to our approach.

Closest related to our goals is the literature on verifying the authenticity/integrity of the internal state of computing devices to confirm that they have not been attacked by malware,

a mechanism usually referred to as *attestation*. A rough taxonomy divides this literature in three general categories, namely *hardware-based attestation* [2, 6, 11, 21], *software-based attestation* [20, 29, 28, 1, 17, 18], and *remote attestation* [6, 11, 21, 14, 4]. Due to the similarity in the goals of attestation with our system, we present a detailed overview of the corresponding literature in the full version of this work [23]. As a general note, we stress that similarly to other practical defense mechanisms, attestation schemes do not come with a formal proof of security or even a theoretical security model.[4] In contrast, our scheme is not only backed by a formal mathematical proof, but has several additional advantages with respect to existing attestation schemes, which are highlighted in the full version of this work [23]. We note, however, that much of the attestation literature allows the system to leak its entire state; although our schemes do not account for such leakage (in particular, the key shares should not leak), one can use sharing-refreshing techniques, e.g., [26, 3], to add protection against periodic leakage. The details of such a scheme are subject of future work.

On the most theoretical side of cybersecurity, cryptography provides solutions whose security is backed by rigorous mathematical proofs that typically reduce hardness of breaking the scheme to hardness of solving a mathematical problem, e.g., factoring. But with only a few exceptions, e.g., [7, 13, 10, 19, 15, 27], the cryptographic literature has not, to the best of our knowledge, targeted the problem of malicious code injection. And in contrast to the security-engineering research, cryptographic solutions that do target this problem are often inefficient and/or adopt a too-abstract model of computation which makes them inapplicable or impractical for today's systems.

1.3 Organization of the Paper

In Section 2 we describe the model of computation and the virus injection model, and in Section 3 we provide our security definitions for VDSs. Our VDS constructions are then described in Sections 4 and 5. Due to limited space, some of the formal definitions and proofs have been moved to [23].

2 The Model

In this section we provide an abstract specification of our model of computation. We use as our basis the well known Random Access Machine (RAM) model but slightly adapt it to be closer to an abstract version of a modern computer following the *von Neumann architecture*. In a nutshell, this modification consist of assuming that both the program and its data are written on the RAM's random access memory[5] which is polynomially bounded. A RAM $\mathcal{R}$ consist of two components: A *Random Access Memory (in short RMEM)* and a *Central Processing Unit (in short CPU)*. The RMEM and the CPU communicate in *fetch-and-execute cycles, aka CPU cycles*, where the number of CPU cycles is the default complexity measure of a RAM. We will refer to a CPU cycle as a *round* in the RAM execution.

The memory RMEM is modeled as a vector $\texttt{MEM}$ of $m = |\texttt{MEM}| = poly(k)$ *words*, where k is the, often implicit, security parameter. Each word is an L-bit string, where we assume that L is linear in k. (Wlog, in the following we will assume that $L = k$.) For $i \in \{0, 1, \ldots, m-1\}$ we denote by $\texttt{MEM}[i]$ the ith word, i.e., the contents of the i-th RMEM register. The CPU

[4] An exception here is [4] which uses an idea similar in spirit to ours – i.e., sharing a secret through the memory – for provably protecting against a class of heap overflow attacks (cf. [23] for a comparison).

[5] In the literature, a RAM with this modification is usually called a *Random Access Stored-Program* machine [12] (in short, RASP).

consists of a much smaller number of L-bit registers – in this work we assume that the total amount of storage of the CPU is linear in the security parameter k – and an instruction set $\mathcal{I}$ that defines which operations can be performed on the registers, and how data are loaded-to/output-from the CPU. The CPU registers include a read-only input-register and an output register which correspond to its interface with its environment (the user). The CPU registers are modeled as an array $\texttt{REG}$ of words, where we denote by $\texttt{REG}[i]$ the content of the ith CPU register. We denote a CPU by the pair $\mathcal{C} = (\texttt{REG}, \mathcal{I})$ of the vector $\texttt{REG}$ of registers and the instruction set $\mathcal{I}$. We denote a RAM with CPU $\mathcal{C}_k$ and RMEM $\texttt{MEM}_k$ as $\mathcal{R}_k = (\mathcal{C}_k, \texttt{MEM}_k)$.[6] The *state* of $\mathcal{R}_k$ at any point in the protocol execution is the vector $(\texttt{REG}_k, \texttt{MEM}_k)$ including the current contents of all its CPU and RMEM registers. Details of the model and formal definitions can be found in [23].

To allow for asymptotic security definitions – where the word size, the size of the CPU (i.e., number of its registers), and the size of the memory depend on the security parameter – we often consider a family of RAMs, $\mathcal{R} = \{\mathcal{R}_k\}_{k\in\mathbb{N}}$ with $\mathcal{R}_k = (\mathcal{C}_k = (\texttt{REG}_k, \mathcal{I}_k), \{\texttt{MEM}_k\})$. (The size of each word processed by $\mathcal{R}_k$ is $L = k$.) The RAM families considered in this work have the following property: The instruction set $\mathcal{I}_k$ is the same for all values of the security parameter k. In particular, we assume that all elements of a RAM family, have the same constant number $c = \mathcal{O}(1)$ of CPU registers, where each register of $\mathcal{R}_k$ is of size k, and there is some set of instructions $\mathcal{I}$ that includes operations defined over strings of arbitrary size such that $\mathcal{I}_k = \mathcal{I}$ for every $k \in \mathbb{N}$. (Of course for each value of k, $\mathcal{I}_k$ corresponds to the restriction of $\mathcal{I}$ on k-bit strings.)

Software Execution and Virus Injection. A program to be executed on a RAM $\mathcal{R} = (\mathcal{C}, \texttt{MEM})$ is described as a vector $W = (w_0, \ldots, w_{n-1}) \in (\{0,1\}^L)^n$ of words that might be instructions, addresses, or program data. To avoid confusion, we refer to such a vector including the (binary of a) software and its corresponding data as *code for* $\mathcal{R}$. By convention, whenever, for a RAM family $\mathcal{R}$ we say that W is code for $\mathcal{R}$, we mean that W is code for the element $\mathcal{R}_k \in \mathcal{R}$ with register size as long as the word size of W and instruction set that includes all the instructions used by W. The execution of a program proceeds as follows: The code W is loaded onto the memory $\texttt{MEM}$. Unless stated otherwise, we assume that W is loaded sequentially on the first $n = |W|$ locations of $\texttt{MEM}$, where all remainder locations are filled with ($\texttt{no\_op}$) instructions. The user might give input(s) to $\mathcal{R}$ by writing them on its input register. The RAM starts its execution by fetching the word of the RMEM which the program counter points to, i.e., $\texttt{MEM}[\texttt{pc}]$ (wlog, initially $\texttt{pc} = 0$). We assume that the RAM is reactive, i.e., any new input written on its input register, makes the RAM resume its computation even if it had halted with output (cf. [23]).

A CPU is *complete* (also referred to as *universal*) if given sufficient (but polynomial) random access memory it can perform any efficient deterministic computation (for a formal definition see [23]). We at times refer to a RAM (family) with a complete CPU as a complete RAM (family).

Modeling Virus Attacks. In our model, a virus attacks a RAM by injecting its code on selected locations of the memory RMEM. More formally, an ℓ-bit virus is modeled as a tuple $\mathbf{v} = (\vec{\alpha}, V) = ((\alpha_0, \ldots, \alpha_{\ell-1}), (b_0, \ldots, b_{\ell-1}))$, where each $\alpha_i \in \vec{\alpha}$ is a location in the memory and each b_i is a bit. The effect of injecting a virus $\mathbf{v}$ into a RAM $\mathcal{R} = (\mathcal{C}, \texttt{MEM})$, is to have, for each $\alpha_i \in \vec{\alpha}$, the α_i-th bit on the memory – i.e., the $(\alpha_i \mod L)$-th bit of the word written

[6] In slight abuse of notation we at times drop the security parameter whenever it is clear from the context.

on register $\texttt{MEM}[\lfloor \frac{\alpha_i}{L} \rfloor]$ – (over)written with b_i. We say that $\mathbf{v}$ is valid for $\mathcal{R}$ if the following properties hold: (1) $\alpha_i \neq \alpha_j$ for every $\alpha_i, \alpha_j \in \vec{\alpha}$, and (2) $\alpha_i \in \{0, \ldots, L(|\texttt{MEM}| - 1)\}$ for every $\alpha_i \in \vec{\alpha}$. Furthermore, we say that $\mathbf{v}$ is *non-empty* if $|\vec{\alpha}| > 0$. Note that in this work we do not consider viruses which inject themselves on the CPU registers.

3 Virus Detection Schemes

We next formalize our notion of provably secure virus detection. More concretely, we introduce the notion of a *virus detection scheme* (in short VDS) which demonstrates how to compile a given program (and its data) into a new program which allows us to detect any virus injection. The detection is done via a challenge-response mechanism.

▶ **Definition 1.** A *virus detection scheme (VDS)* $\mathcal{V}$ consists of five (potentially) randomized algorithms, i.e., $\mathcal{V} = (\texttt{Gen}, \texttt{Comp}, \texttt{Chal}, \texttt{Resp}, \texttt{Ver})$, defined as follows:[7]

- $\texttt{Gen}$ is a key-generation algorithm; it computes a pair (compilation-key, verification-key), i.e., $(K_c, K_v) \xleftarrow{\$} \texttt{Gen}$.[8]
- $\texttt{Comp}$ on input the description $\mathcal{R} = \{(\mathcal{C}_k, \texttt{MEM}_k)\}_{k \in \mathbb{N}}$ of a RAM family,[9] some code W for $\mathcal{R}_k$, and a compilation key K_c, $\texttt{Comp}$ outputs a new code $\widetilde{W}$ for $\mathcal{R}_k$ (which we will refer to as *secure code*) ; i.e., $\widetilde{W} \xleftarrow{\$} \texttt{Comp}(\mathcal{R}_k, W, K_c)$.
- $\texttt{Chal}$ on input a verification key K_v, and a string $z \in \text{INP}_{\texttt{Chal}} \subseteq \{0,1\}^{poly(k)}$, $\texttt{Chal}$ outputs $poly(k)$-bit string c called the *challenge*; i.e., $c \xleftarrow{\$} \texttt{Chal}(z, K_v)$.
- $\texttt{Resp}$ on input a string $c \in \text{OUT}_{\texttt{Chal}}$ and some code $\widetilde{W}$, $\texttt{Resp}$ outputs a $poly(k)$-bit string y called the *response*; i.e., $y \xleftarrow{\$} \texttt{Resp}(c, \widetilde{W})$.
- $\texttt{Ver}$ on input a verification key K_v, a message $z \in \text{INP}_{\texttt{Chal}}$, a challenge c and a response y, $\texttt{Ver}$ outputs a bit b; i.e., $b \xleftarrow{\$} \texttt{Ver}(K_v, z, c, y)$. We say that $\texttt{Ver}$ accepts iff $b = 1$.

A VDS should satisfy the following four security properties (see [23] for formal definitions).[10] The first property is *verification correctness* which, intuitively, guarantees that if the RAM has not been attacked, then the reply to the challenge is accepting. The second property is *compilation correctness*, which intuitively ensures that the compiled code $\widetilde{W}$ performs the same computation (on $\mathcal{R}$) as the original code W. In an application of a VDS, the verifier inputs the challenge at a point of his choice and checks that the reply verifies according to the predicate $\texttt{Ver}$. Thus, the third property of a VDS is *self-responsiveness*: the secured code $\widetilde{W}$ includes code that on some special input emulates algorithm $\texttt{Resp}$ on the RAM it executes. Finally, the fourth property requires *detection accuracy* which states that if some non-empty malware gets injected onto the RAM, then it is detected with high probability. This is one of the most challenging properties to ensure and is the heart of any VDS. We next specify this property by means of a security game between an adversary $\texttt{Adv}$ who aims to inject a virus on a RAM, and a challenger $\texttt{Ch}$ who aims to detect it.

---

[7] All five algorithms below take as an additional input the security parameter k, which is omitted for compactness.

[8] Note that if we instantiate the VDS with symmetric-key cryptography then $K_c = K_v$.

[9] This description of the family includes the word-size, the size and addresses of the CPU and RMEM registers and (an encoding) of the instruction set $\mathcal{I}$.

[10] Without loss of generality, whenever we refer to the secure (i.e., compiled by $\texttt{Comp}$) version $\widetilde{W}$ of some code W for a RAM family $\mathcal{R}$ we will implicitly assume that $\mathcal{R}$ has enough memory for writing $\widetilde{W}$ on it, i.e., if the word size of $\widetilde{W}$ is k then $|\texttt{MEM}_k| > |\widetilde{W}|$.

3.1 Detection Accuracy as a Security Game

At a high-level, the security game, denoted by $\mathcal{G}_{\mathtt{VDS}}^{\mathcal{R},\mathcal{V},W}$, proceeds as follows: The challenger `Ch` runs the key-generation algorithm to obtain a key-pair (K_c, K_v), and compiles some code W for RAM $\mathcal{R} = (\mathcal{C}, \mathtt{MEM})$ onto a new code $\widetilde{W}$ for $\mathcal{R}$ by invocation of algorithm `Comp`; `Ch` then emulates an execution of $\widetilde{W}$ on $\mathcal{R}$, i.e., emulates its CPU cycles and stores its entire state at any given point. The adversary is allowed to inject a virus of his choice on any location in the memory `MEM`. Eventually, the challenger executes the *(virus) detection procedure*: It computes a challenge c by invocation of algorithm $\mathtt{Chal}(z, K_v)$, and then feeds input $(\mathtt{check}, c)$ to the emulated RAM and lets it compute the response y. The adversary wins if the output b of the verification algorithm with this response y equals 1. To capture worst case attack scenarios, we allow the adversary to inject his virus at any point during the RAM emulation and make no assumption as to how many rounds the RAM executes after the virus has been injected and before it receives the challenge. Furthermore, we make no assumptions on how much information the adversary holds on the original code W or on the inputs/outputs of $\mathcal{R}$. The formal description of the security game $\mathcal{G}_{\mathtt{VDS}}^{\mathcal{R},\mathcal{V},W}$ can be found in [23].

▶ **Definition 2.** We say that a virus detection scheme $\mathcal{V}$ =(`Gen`,`Comp`,`Chal`,`Resp`,`Ver`) is secure for RAM family $\mathcal{R}$ if it satisfies the following properties:

1. $\mathcal{V}$ is verification correct, compilation correct, and self-responsive.
2. For sufficiently large k for any code W for $\mathcal{R}_k$ and any polynomial adversary `Adv` in the game $\mathcal{G}_{\mathtt{VDS}}^{\mathcal{R}_k,\mathcal{V},W}$ who injects a valid non-empty virus the following holds: $\Pr[b = 1]$ is negligible, where the probability is taken over the random coins of `Adv` and `Ch`.

In our constructions we restrict our statements to a certain class of code that satisfies some desirable properties making the compilation easier. We will then say that the corresponding VDS is *secure with respect to the given class of code*.

The Repeated Detection Game $\tau\mathcal{G}_{\mathtt{VDS}}^{\mathcal{R},\mathcal{V}_1,W}$. Definition 2 requires that the adversary is caught even when he injects its virus while the RAM is executing the `Resp` algorithm. We next describe a relaxed security game, which provides a useful guarantee for practical purposes. In this game the virus detection (challenge/response) procedure is executed multiple times periodically (on the same compiled code); the requirement is that if the adversary injects its virus to the RAM at round ρ, then he will be caught by the first invocation of the virus detection procedure which starts *after round* ρ. Note that all executions use *the same* compiled RAM program and therefore the same key K. The corresponding security game $\tau\mathcal{G}_{\mathtt{VDS}}^{\mathcal{R},\mathcal{V},W}$ which involves τ executions of the detections procedure is detailed in [23]. The security definition is similar to Definition 2 but requires that any virus that is injected in the RAM will be caught *in the first virus detection attempt performed after the injection* . Our VDSs will be proven secure in this repeated detection game. In [23] we describe a simple trick which transforms our VDSs to ones that are secure in the standard game using a secure hash function.

4 A Software-based VDS for Continuous Moderate-size Virus

In this section we provide a VDS which is secure assuming the virus is linear in the security parameter and that it is injected in continuous memory locations, i.e., if $\mathbf{v} = ((\alpha_0, \ldots, \alpha_{|V|-1}), V)$, then $\alpha_i = \alpha_{i-1} + 1$ for all $i \in [|V| - 1]$. This captures the entire class of tampering accounted for in [4]. In Section 5 we will show how to get rid of these restrictions. Our construction proceeds in two steps. First, we show how to construct a VDS $\mathcal{V}_1$ which

achieves a weaker notion of security that, roughly, does not have self-responsiveness. In a second step, we show how to transform $\mathcal{V}_1$ into a VDS $\mathcal{V}_2$ which is secure in the repeated detection game.

A VDS without self-responsiveness. We start by describing a VDS $\mathcal{V}_1 = (\mathtt{Gen}_1, \mathtt{Comp}_1, \mathtt{Chal}_1, \mathtt{Resp}_1, \mathtt{Ver}_1)$ which achieves security without self-responsiveness: The corresponding attack-game $\mathcal{G}^{\mathcal{R},\mathcal{V},W}_{\mathtt{VDS}-}$ is derived from the standard attack-game $\mathcal{G}^{\mathcal{R},\mathcal{V},W}_{\mathtt{VDS}}$ by modifying the detection procedure so that instead of emulating $\mathcal{R}$ on the compiled code $\widetilde{W}$ and input $(\mathtt{check}, c)$ to compute $y = \mathtt{Resp}(c, \widetilde{W})$, the challenger evaluates $y \xleftarrow{\$} \mathtt{Resp}(c, \widetilde{W})$ himself on the current contents $\widetilde{W}$ of the memory of the emulated RAM.

At a high level, the idea of our construction is as follows: The key generation algorithm $\mathtt{Gen}_1$ samples a k-bit key K for a symmetric-key cryptosystem, i.e., $K_c = K_v = K \xleftarrow{\$} \{0,1\}^k$. Given key K, the algorithm $\mathtt{Comp}_1$ computes an additive sharing $\langle K \rangle$ of K and fills the entire memory $\mathtt{MEM}$ by interleaving a different share of $\langle K \rangle$ between every two words in the original code. Concretely, $\mathtt{Comp}$ compiles some code W for a RAM $\mathcal{R}$ into some new code $\widetilde{W}$ for $\mathcal{R}$ constructed as follows: Between any two consecutive words w_i and w_{i+1} of W the compiler interleaves a uniformly chosen k-bit string $K^{i,i+1} = K_1^{i,i+1} || \ldots || K_{\frac{k}{L}}^{i,i+1}$, where each $K_j^{i,i+1} \in \{0,1\}^L$. In the last k bits of the compiled code (i.e., after the last word $w_{|W|-1}$) the string $K^{last} = K \oplus \bigoplus_{i=0}^{|W|-2} K^{i,i+1}$ is written.[11]

To ensure that the compiled code $\widetilde{W}$ executes the same computation as W we need that while being executed it "jumps over" the key-share locations. For this purpose, we do the following modification: After each word w_j of W we insert a $(\mathtt{jumpby}, n)$ instruction where n is the number of key-shares between this and the next W-word in the compiled code. Similarly, we modify any "jump" instructions of the original code W to point to the correct locations in $\widetilde{W}$. Note that this modification is not necessarily applicable to arbitrary code, e.g., for certain self-modifying code it might even be infeasible. However, it is possible for a complete class of code, which we refer to as *non-self-modifying structured code,* – roughly, this includes code that does not modify its instructions and might only jump by a fixed amount in each operation.

In the following we provide the description of our compiler. The first step to this direction is a process, called $\mathtt{Spread}$, which spreads such a code to allow for enough space between the words to fit the key-shares and adds the extra "jump" instructions to preserve the right program flow. The compiler $\mathtt{Comp}_1$ uses the process $\mathtt{Spread}$ to translate, as sketched above, some non-self-modifying structured code W for a RAM $\mathcal{R} = (\mathcal{C}, \mathtt{MEM})$ into (secured) code $\widetilde{W}$ for $\mathcal{R}$. The algorithm $\mathtt{Chal}_1$ chooses random string $x \in \{0,1\}^k$, and computes the challenge as an encryption of x with key K; i.e., $c \leftarrow \mathtt{Chal}(x, K) = \mathtt{Enc}_K(x)$. For achieving security without self-responsiveness, we can take $\mathtt{Enc}$ to be the one-time pad, i.e., $\mathtt{Enc}_K(x) = x \oplus K$. The corresponding algorithm $\mathtt{Resp}_1$ works as follows: On input the challenge $c = \mathtt{Enc}_K(r)$ and the compiled code $\widetilde{W}$, it reconstructs K by summing up all its shares as retrieved from $\widetilde{W}$, and outputs a decryption of the challenge under the reconstructed key, i.e., outputs $y = c \oplus K$. The corresponding verification algorithm $\mathtt{Ver}_1$ simply verifies that $y = x$.

The security of the above scheme follows from the fact that an injection of a sufficiently long continuous virus will have to overwrite a substantial number of bits of a key-share, which will make it infeasible to correctly decrypt the challenge and thus pass the VDS check. The formal security proof of the statement can be found in [23].

[11] Wlog, here we implicitly assume that $(|\mathtt{MEM}| - |W|) = 0 \mod k$ so that the keys fit exactly in the memory. The general case can also be easily treated.

▶ **Theorem 3.** *Assuming $\mathcal{R}$ is a complete RAM family the VDS $\mathcal{V}_1$ is secure for $\mathcal{R}$ without self-responsiveness with respect to the class of all non-self-modifying structured codes and the class of adversaries who injects a virus $\mathbf{v}$ with $|\mathbf{v}| \geq 2L + \ell$ where $\ell = \omega(\log k)$ on consecutive memory locations.*

Adding Self-Responsiveness. We next modify $\mathcal{V}_1$ so that it is secure (with self-responsiveness) in the repeated detection game. Due to space limitation, the detailed transformation has been moved to [23]; here we restrict to an overview of the basic technical ideas. The first step is to devise a code $W_{\texttt{Resp}}$ for computing a given VDS response algorithm `Resp` and then combine it with W via a threading mechanism: When executing, the combined code $\text{Emb}(W, W_{\texttt{Resp}})$ (periodically) checks if a verification request-input with challenge c is handed to $\mathcal{R}$; if so, it stores the CPU state on free memory locations – namely, locations at the end of the memory that are occupied by (`no_op`) – and changes the program counter to point to the location where $W_{\texttt{Resp}}$ is stored; once `Resp` has produced output, it restores the CPU to its prior state and continues the execution of W.

But the above modification applied on $\mathcal{V}_1$ does not yield a secure VDS, as it only detects attacks (virus injections) that occur *outside* the virus verification procedure, i.e., has the TOCTOU vulnerability discussed in the introduction. To solve this, instead of using a single compilation-key K of length k we use a $2k$-bit key K which is parsed as two k-bit keys $K_{\texttt{od}}$ and $K_{\texttt{ev}}$ via the following transformation: Let $K = x_1 || \dots || x_{\frac{2k}{L}}$, where each x_i is a word.[12] Then $K_{\texttt{od}}$ is a concatenation of the odd-indexed words, i.e., x_i's with $i = 1 \mod 2$, and $K_{\texttt{ev}}$ is a concatenation of the even indexed words. Now the challenge algorithm outputs a double encryption of z with keys $K_{\texttt{od}}$ and $K_{\texttt{ev}}$, i.e., $c = \texttt{Enc}_{K_{\texttt{od}}}(\texttt{Enc}_{K_{\texttt{ev}}}(z))$. In order to decrypt, the response algorithm does the following: First it reconstructs $K_{\texttt{od}}$ by XOR-ing the appropriate shares, and uses it to decrypt c, thus computing $\texttt{Enc}_{K_{\texttt{ev}}}(z)$. Subsequently, it erases $K_{\texttt{od}}$ from the CPU register (e.g., by filling the register where $K_{\texttt{od}}$ is stored with 0's) and *after the erasure completes* it starts reconstructing $K_{\texttt{ev}}$ and uses it (as above) to decrypt $\texttt{Enc}_{K_{\texttt{ev}}}(z)$ and output $y = z$.

The above modification ensures that in order to correctly answer the challenge, the virus needs both $K_{\texttt{od}}$ and $K_{\texttt{ev}}$. However, the keys are never simultaneously written in the CPU. Thus if the adversary injects the virus before $K_{\texttt{od}}$ is erased he will overwrite bits from a share of $K_{\texttt{ev}}$ (which at that point exists only in the memory RMEM); thus he will not be able to decrypt the challenge. Otherwise, if the adversary injects the virus after $K_{\texttt{od}}$ has been erased from the CPU, he will overwrite bits from a share of $K_{\texttt{od}}$; in this case he might successfully pass this detection attempt, but will fail the next detection attempt.

But we are still not done, because the above argument cannot work with any encryption scheme, e.g., it fails if we instantiate $\texttt{Enc}(\cdot)$ with one-time-pad encryption as in VDS $\mathcal{V}_1$. The reason is that once inside the system, the adversary might be able to use the key material it has not destroyed to answer the challenge with non-negligible probability. To avoid this, we make use of a public-key leakage-resilient encryption scheme, e.g., [8], which is secure as long as the adversary's probability of guessing the key is negligible even when one leaks a big part of the key.

The above tricks are the heart of our modified VDS, but a few more low level hacks are employed to ensure that the modified protocols compose well with the ones from the previous section. For example, we need to make sure that the part of the code $\text{Emb}(W, W_{\texttt{Resp}})$ that

[12] We provide the general solution for $k = Lq$ for some q; with the simplification that $L = k$, we get that $K = x_1 || x_2 = K_{\texttt{od}} || K_{\texttt{ev}}$.

performs the verification accesses the correct key-share locations. We refer to [23] for the details and the security proof of the resulting VDS $\mathcal{V}_2 = (\mathtt{Gen}_2, \mathtt{Comp}_2, \mathtt{Chal}_2, \mathtt{Resp}_2, \mathtt{Ver}_2)$.

On the performance of our VDS. It is easy to verify that the above VDS induces a moderate slowdown, i.e., a factor two, due to processing the newly inserted jump instructions plus whatever slowdown the periodically checking for inputs might impose, on the execution of the code on the RAM (while the code is not being verified). Both slowdowns can be reduced. E.g., the factor two is an overestimate as several instructions are already "jumps" plus reading data does not execute "jumps"; similarly, the check for the input can be executed periodically (or explicitly assumed as being part of any CPU-cycle). On the other hand, verification is quite heavy as it needs to traverse almost the entire memory but this is not our main consideration as (1) we believe that our scheme's provable security guarantee makes the waiting acceptable, and (2) unlike existing attestation methods the verification does not need to stop the normal execution of the program – e.g., in a RAM with parallel processors it could be done by a single designated processor.

In terms of memory usage, the secured code requires a linear (concretely, a factor four) amount of memory with respect to the original code. Although we are less concerned about memory needs – memory is an inexpensive expandable resource the capacities of which increase faster than computation speed – improving the memory usage is an interesting research direction.

5 Detecting Arbitrary Injection

The above VDSs are secure only against (sufficiently) long and consecutive viruses. In this section we describe a construction which is secure independently of the virus length even if the virus bits are not injected continuously, and in particular even when the virus targets non-key-share locations. To achieve this ultimate security guarantees we use a compiler similar to $\mathtt{Comp}_2$, but we include, for each code word and pair of key-shares, message-authentication-code tags (MACs) which we verify every time we load a code word to the CPU. Concretely, the difference with $\mathtt{Comp}_2$ is that $\mathtt{Comp}_3$ appends a MAC tag t^i to each word, keyed with the (concatenation of the) key-shares $K_{\mathtt{od}}^{i,i+1}||K_{\mathtt{ev}}^{i,i+1}$ that follow this word. We use a MAC with a special leakage-resilient property that ensures that the adversary cannot forge a tag even when he knows a large portion of the key. Thus, if the malware overwrites only a few bits of some of the key-shares it will be unable to guess an appropriate manipulation of the MAC key. And if it overwrites a large portion it will be unable to answer decryption challenges.

To use the power of the MACs we need to make sure that during the program execution, before loading any word to the CPU we first verify its MAC. To this direction, we assume that, by default, the CPU loads values from the memory RMEM to its registers via a special load instruction $(\mathtt{read\_auth}, i, j)$, which fetches five consecutive words (corresponding to $\widetilde{w}_i$ and its MAC key and tag), verifies the MAC with the corresponding keys, and only if the MAC verifies, it keeps the word on the CPU to process. If the MAC verification fails, then $(\mathtt{read\_auth}, i, j)$ deletes at least one of the key-shares from the memory, thus introducing an inconsistency that will be caught by the detection procedure. Furthermore, to ensure that the compiled program will not introduce inconsistencies, our compiler replaces every $(\mathtt{write}, j, i)$ instruction (which writes the contents of register j in RMEM location i) with microcode, denoted as `write_auth`, which, when writing a word in the memory it also updates the corresponding MAC tag. We stress that, unlike `read_auth` which we need the CPU to support as an atomic instruction, the instruction `write_auth` does *not* require any

change in the architecture or additional assumptions as it can be implemented in code itself. The details of the corresponding VDS $\mathcal{V}_3$ along with its security proof can be found in [23].

▶ **Theorem 4.** *Let $\mathcal{R}$ be a complete RAM. If the encryption scheme used in $\mathcal{V}_3$ is CPA secure even against an adversary who learns all but $\omega(\log k)$ bits of the secret key, then $\mathcal{V}_3$ is secure for $\mathcal{R}$ in the repeated-detection mode with respect to the class of all non-self-modifying structured codes and adversaries in the bit-by-bit (non-continuous) injection model.*

References

1. T. AbuHmed, N. Nyamaa, and D. Nyang. Software-based remote code attestation in wireless sensor network. In *GLOBECOM'09*, pages 4680–4687. IEEE Press, 2009.
2. W. A. Arbaugh, D. J. Farber, and J. M. Smith. A secure and reliable bootstrap architecture. In *SP'97*, pages 65–, Washington, DC, USA, 1997. IEEE Computer Society. URL: `http://dl.acm.org/citation.cfm?id=882493.884371`.
3. J. Baron, K. El Defrawy, J. Lampkins, and R. Ostrovsky. How to withstand mobile virus attacks, revisited. In M. M. Halldorsson and S. Dolev, editors, *PODC 2014*, pages 293–302. ACM Press, 2014.
4. A. Boldyreva, T. Kim, R. J. Lipton, and B. Warinschi. Provably-secure remote memory attestation for heap overflow protection. Cryptology ePrint Archive, Report 2015/729, 2015.
5. D. Boneh, R. A. DeMillo, and R. J. Lipton. On the importance of checking cryptographic protocols for faults. In *EUROCRYPT' 97*, volume 1233 of *LNCS*, pages 37–51. Spriger, 1997.
6. B. Chen and R. Morris. Certifying program execution with secure processors. In *HOTOS'03*, pages 23–23. USENIX Association, 2003.
7. D. Dachman-Soled, F.-H. Liu, E. Shi, and H.-S. Zhou. Locally decodable and updatable non-malleable codes and their applications. In *TCC 2015*, volume 9014 of *LNCS*, pages 427–450. Springer, 2015.
8. Y. Dodis, S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC 2010*, volume 5978 of *LNCS*, pages 361–381. Springer, 2010.
9. Y. Dodis, Y. T. Kalai, and S. Lovett. On cryptography with auxiliary input. In *STOC '09*, pages 621–630. ACM, 2009.
10. S. Dziembowski, K. Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS 2010*, pages 434–452, 2010.
11. K. El Defrawy, G. Tsudik, A. Francillon, and D. Perito. SMART: secure and minimal architecture for (establishing dynamic) root of trust. In *NDSS 2012*. The Internet Society, 2012.
12. C. C. Elgot and A. Robinson. Random-access stored-program machines, an approach to programming languages. *J. ACM*, 11(4):365–399, October 1964.
13. S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi. A tamper and leakage resilient von neumann architecture. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 579–603. Springer, 2015.
14. A. Francillon, Q. Nguyen, K. B. Rasmussen, and G. Tsudik. A minimalist approach to remote attestation. In *DATE'14*, pages 244:1–244:6. European Design and Automation Association, 2014.
15. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.

16 P. Gemmell and M. Naor. Codes for interactive authentication. In *CRYPTO'94*, volume 773 of *LNCS*, pages 355–367. Springer, 1994.

17 M. Jakobsson and K.-A. Johansson. Practical and secure Software-Based attestation. In *LightSec 2011*, 2011.

18 M. Jakobsson and G. Stewart. Mobile malware: Why the traditional AV paradigm is doomed, and how to use physics to detect undesirable routines. In *BlackHat 2013*, 2013.

19 A. Juels and B. S. Kaliski Jr. Pors: Proofs of retrievability for large files. In *CCS 2007*, pages 584–597. ACM, 2007.

20 R. Kennell and L. H. Jamieson. Establishing the genuinity of remote computer systems. In *SSYM'03*, pages 21–21. USENIX Association, 2003.

21 X. Kovah, C. Kallenberg, C. Weathers, A. Herzog, M. Albin, and J. Butterworth. New results for timing-based attestation. In *SP 2012*, pages 239–253. IEEE Computer Society, 2012.

22 R. J. Lipton, R. Ostrovsky, and V. Zikas. Provably secure virus detection. U.S. Patent (pending), Application No. 62/054,160, 2014.

23 R. J. Lipton, R. Ostrovsky, and V. Zikas. Provably secure virus detection: Using the observer effect against malware. Cryptology ePrint Archive, Report 2015/728, 2015.

24 F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. Innovative instructions and software model for isolated execution. In *HASP'13*, pages 10:1–10:1. ACM, 2013.

25 W. S. McPhee. Operating systems integrity in os/vs2. *IBM Systems Journal, 13 Issue 3*, pages 230–252, 1974.

26 R. Ostrovsky and M. Yung. How to withstand mobile virus attacks (extended abstract). In L. Logrippo, editor, *PODC '91*, pages 51–59. ACM, 1991.

27 A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *STOC 2014*, pages 475–484. ACM, 2014.

28 A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla. Scuba: Secure code update by attestation in sensor networks. In *WiSe'06*, pages 85–94. ACM, 2006.

29 A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems. In *SOSP'05*, pages 1–16. ACM, 2005.

30 A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

An Almost Cubic Lower Bound for Depth Three Arithmetic Circuits*

Neeraj Kayal[1], Chandan Saha[2], and Sébastien Tavenas[3]

1 **Microsoft Research India, Bangalore, India**
neeraka@microsoft.com
2 **Indian Institute of Science, Bangalore, India**
chandan@csa.iisc.ernet.in
3 **Microsoft Research India, Bangalore, India**
sebastien.tavenas@ens-lyon.org

Abstract

We show an almost cubic lower bound on the size of any depth three arithmetic circuit computing an explicit multilinear polynomial in n variables over any field. This improves upon the previously known quadratic lower bound by Shpilka and Wigderson [CCC, 1999].

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases arithmetic circuits, depth-3 circuits, shifted partials

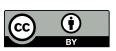
Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.33

1 Introduction

An arithmetic circuit is a directed acyclic graph with leaves (nodes with in-degree zero) labeled by formal variables and other nodes labeled by addition ($+$) or multiplication ($\times$) operations. Nodes with out-degree zero are the output nodes; for simplicity and without losing generality we will assume that there is only one output node in a circuit. Non-leaf nodes are also referred to as addition or multiplication *gates*. Such a circuit naturally represents a multivariate polynomial; we say this polynomial is *computed* at the output node of the circuit (or simply computed by the circuit). Two parameters that determine the complexity of a circuit are its *size* and *depth*, which are respectively the number of edges and the length of the longest path from any input node to the output node of the circuit. Computations involving arithmetic operations can be naturally modeled by arithmetic circuits and hence study of these objects forms a fundamental aspect of complexity theory.

Research on arithmetic circuits received a great impetus from the seminal paper by Valiant [42] who defined two non-uniform complexity classes that are algebraic analogues of classes P and NP. These algebraic complexity classes are known as VP and VNP in the literature. Class VP consists of families of polynomials $\{g_n\}_{n\geq 1}$ such that the number of variables and the degree of g_n are $n^{O(1)}$, and there is an arithmetic circuit of size $n^{O(1)}$ computing g_n. A family of polynomials $\{f_n\}_{n\geq 1}$ is in VNP if there is another family of polynomials $\{g_n(\mathbf{x},\mathbf{y})\}_{n\geq 1}$ in VP such that $f_n = \sum_{\mathbf{y}\in\{0,1\}^{|\mathbf{y}|}} g_n(\mathbf{x},\mathbf{y})$. Valiant defined a notion of completeness for the classes VNP and VP, and showed that the family of permanent polynomials is VNP-complete whereas the family of determinant polynomials is *almost* complete for VP. This gave rise

* A full version of the paper (which contains all the missing proofs) can be found at http://eccc.hpi-web.de/report/2016/006/



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 33; pp. 33:1–33:15

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

to the famous 'determinantal complexity of the permanent' problem, a suitable resolution of which would imply $\mathsf{VP} \neq \mathsf{VNP}$ or equivalently a super-polynomial size lower bound for arithmetic circuits. We refer the reader to the surveys [26, 39], the book [4] and the paper [27] for more on these and other related algebraic complexity classes, their inter-relationships and their associations with Boolean complexity classes. Throughout this article, whenever we use the term 'circuit(s)' we will mean 'arithmetic circuit(s)'.

Starting with Valiant's work there has been significant progress in proving lower bounds for several restricted models of arithmetic circuits. Multilinear [31, 30, 34], noncommutative [28, 25], monotone [13] and special low-depth circuits [29, 8, 36, 35, 32, 1, 15, 16, 21, 19] are examples of such interesting circuit classes. But still, our knowledge of general circuit lower bound is rather limited. The best known lower bound for general circuits is Baur and Strassen's $\Omega(n \log d)$ bound [40, 3] for circuits computing the simple polynomial $\sum_{i\in[n]} x_i^d$. A recent line of work on depth reduction, starting with [2, 43] and culminating with [20, 10, 41], has shown that a moderately strong lower bound for circuits of *depth three*[1] implies a super-polynomial lower bound for general circuits. Also, Raz [33] showed that a strong enough lower bound for a special kind of (namely, set-multilinear) depth three circuits implies a super-polynomial lower bound for general arithmetic formulas[2]. These depth reduction results have opened up the possibility of proving a super-polynomial lower bound for general circuits/formulas by first proving strong lower bounds for low-depth, in particular depth three, circuits. The hope is depth three circuits, which have an apparent simple structure, might be more amenable to lower bound proofs. But, unfortunately, even at depth three we do not know of any super-polynomial lower bound over fields of characteristic zero!

Depth three circuits. In this paper, whenever we mention a depth three circuit we will mean a $\Sigma\Pi\Sigma$ circuit that has an addition gate at the top, followed by a layer of multiplication gates and finally a bottom layer of sum gates. Such a circuit is a "sum of product of linear polynomials" representation of the computed polynomial. The fan-in of the top addition gate is called the top fan-in, and that of the bottom layer of addition gates the bottom fan-in of the circuit. Observe that bottom fan-in can be at most $n+1$ where n is the number of variables. The *multiplicative complexity* of a depth three circuit $\mathtt{C}$ is the sum of the fan-ins of the multiplication gates of the circuit, i.e. if $\mathtt{C} = \sum_{i=1}^{s} l_{i1} \cdots l_{id_i}$ where l_{ij}'s are linear polynomials then multiplicative complexity of $\mathtt{C}$ is $\sum_i^s d_i$. It is easy to see that multiplicative complexity is less than the size of a depth three circuit. Circuit $\mathtt{C}$ is *homogeneous* if l_{ij}'s are homogeneous linear polynomials (a.k.a. linear forms).

Previous works on depth three circuit lower bound. In [38, 36], Shpilka and Wigderson proved an $\Omega(n^2)$ lower bound on the multiplicative complexity of depth three circuits computing the elementary symmetric polynomial

$$\text{ESym}_n^d(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \sum_{S\subseteq[n],|S|=d} \prod_{i\in S} x_i$$

on n-variables and degree $d = \Theta(n)$. This bound is essentially optimal for fields of size more than n, as n-variate elementary symmetric polynomials can be computed by depth three

[1] over fields of characteristic zero
[2] a *formula* is a circuit whose underlying directed acyclic graph is a tree

circuits with multiplicative complexity $O(n^2)$ [3]. A similar tight quadratic lower bound but for the power symmetric polynomial $\sum_{i\in[n]} x_i^n$ was shown in [12]. Also, a near quadratic lower bound is known for the determinant polynomial [38, 36]. The situation is a lot better over small fields or under the restriction of homogeneity. An exponential lower bound was shown by [8] (and by [9]) for depth three circuits over any fixed finite field computing the determinant polynomial (even if the circuit and the determinant are treated in the algebra of functions over the finite field). It was shown in [29] that any homogeneous depth three circuit computing ESym_n^{2d} has size $\Omega((n/4d)^d)$. Recently, [16] showed a lower bound of $n^{\Omega(\sqrt{d})}$ for depth three circuits, with bottom fan-in bounded by n^ε for any fixed $\varepsilon < 1$, computing an explicit n-variate polynomial of degree d.

1.1 Our results

▶ **Theorem 1** (Depth-3 circuit lower bound). *There is a family of homogeneous multilinear polynomials $\{f_n\}_{n\geq 1}$ in* VNP*, where f_n is a $\Theta(n)$-variate polynomial of degree $\Theta(n)$ such that any depth-3 circuit computing f_n has multiplicative complexity (and hence size) $\Omega\left(\frac{n^3}{(\ln n)^2}\right)$.*

Theorem 1 can be seen as an improvement in the state of the art of the long-standing quadratic lower bound for depth three circuits [38, 36], although our target polynomial family is harder – it is in VNP and not known to be in VP. Also, from our analysis, we arrive at a near quadratic lower bound for the *symmetric model* defined in [37] thereby improving upon the linear bound therein (Theorem 2).

Let ESym_m^d be an elementary symmetric polynomial in m variables and of degree d. Borrowing terminologies from [37], a *symmetric circuit* has a bottom layer of plus gates computing linear polynomials, and a top gate that computes some elementary symmetric polynomial on the linear polynomials computed at the bottom level gates. Thus, a symmetric circuit with m bottom level gates outputs a polynomial of the form $\text{ESym}_m^d(l_1, \ldots, l_m)$ for some d, where $l_1, \ldots, l_m$ are linear polynomials computed by the m bottom level gates. The parameter m is defined as the *size* of the symmetric circuit. This model was shown to be complete or universal in [37] (i.e. every polynomial can be computed in this model), and linear lower bounds were shown on the size of the smallest symmetric circuit computing the determinant polynomial and the polynomial $\prod_{i=1}^{n/2} x_i + \prod_{i=n/2+1}^{n} x_i$. The following theorem improves this lower bound but once again the target polynomial family is likely harder than the ones studied in [37].

▶ **Theorem 2** (Symmetric circuit lower bound). *Let $\{f_n\}_{n\geq 1}$ be the polynomial family of Theorem 1. The size of the smallest symmetric circuit computing f_n is $\Omega\left(\frac{n^2}{(\ln n)^2}\right)$ over any infinite field.*

In an attempt to make progress in understanding lower bounds for circuit models where formal degree of the circuit is much higher than the number of variables (as might be the case for a depth three circuit), [17] posed the problem of proving lower bounds for homogeneous depth three circuits with formal degree much larger than the number of variables. The following theorem gives a solution to this problem.

▶ **Theorem 3** (Homogeneous depth three circuits with high degree). *For any positive integer $d = d(n) \geq n$, there exists an explicit family $\{f_{n,d}\}$ of n-variate polynomials of degree d*

[3] this follows from an interpolation trick attributed to Michael Ben-Or in [29]

such that any homogeneous depth three circuit computing $f_{n,d}$ must have size at least $2^{\Omega(n)}$. Moreover, one can even choose such a family $f_{n,d}$ so that it can in fact be computed by a $(nd)^{O(1)}$-sized algebraic branching program[4].

The above theorem can be viewed as a generalization of the lower bound by [29] for homogeneous depth three circuits. Since elementary symmetric polynomials in n-variables have degree at most n, the lower bound in [29] holds for homogeneous depth three circuits with degree less than the number of variables. To the best of our knowledge, a lower bound of $(nd)^{\omega(1)}$ for homogeneous depth three circuits with degree d much greater than the number of variables n was not known. Theorem 3 fills in this gap in our understanding as long as $d = 2^{o(n)}$. However, note that the lower bound in the above theorem is independent of d, ideally one should get $d^{\Omega(n)}$ instead of $2^{\Omega(n)}$.

1.2 Proof ideas

Like in many of the previous works, we use a measure $\mu : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$ to capture some 'weakness' of a circuit family as opposed to a 'hard' family of polynomials which leads to a lower bound for the circuit family. In both Theorem 1 and 3, the improvements are achieved by applying the *dimension of the shifted partials* measure, introduced in [14], and used subsequently (at times with certain crucial alterations) in many other recent lower bound results [11, 18, 7, 15, 22, 16, 24, 21, 19, 23]. The shifted partials measure is a generalization of the dimension of the partial derivatives measure used previously in [29, 36]. It is quite effective in proving lower bounds for the model of depth four ($\Sigma\Pi\Sigma\Pi$) circuits with formal degree close to the actual degree of the computed polynomial, and somewhat low bottom fan-in [11, 18]. In fact, all the recent lower bounds (for restricted depth 3 and 4 circuits) obtained using shifted partials 'reduce' to this case of depth four circuits one way or the other. We take a similar route here, but make the crucial observation that a simple "grouping" step in the analysis with shifted partials gives some *leeway to the formal degree* of the circuit and allows it to grow over the actual degree of the computed polynomial. This observation and a careful construction of the target family of polynomials to take advantage of this leeway are the primary sources of improvement of the depth three lower bound.

An immediate hurdle in proving lower bounds for depth three circuits is that the formal degree of the circuit can be much larger than the degree and number of variables of the computed polynomial. The existing proof techniques and measures have had limited success in handling high formal degree circuits [16, 23]. To get around this first hurdle, we begin by following the same approach as in [36] of pruning the circuit of high degree product gates by going modulo some linear polynomials picked from among the factors of such 'heavy' product gates. This step is exactly (borrowing terminologies from [36]) satisfying some affine linear constraints and restricting the circuit to an affine subspace. However, the degree threshold used to define 'heavy' product gates can now be chosen higher than that in [36] because of the 'leeway to formal degree' provided by shifted partials. In the pruned circuit, a simple "grouping" of linear polynomials in every product term of a depth three circuit turns out to be surprisingly effective in handling the remaining product gates. The grouping step transforms a depth three circuit to a depth four circuit with bottom fan-in more than 1, but at the same time brings down the number of factors in every product term. The tradeoff between the bottom fan-in and the number of factors per product term is then analyzed to obtain a suitable upper bound on the shifted partials dimension of a depth three circuit.

[4] The definition of an algebraic branching program can be found, for example, in [39].

Finally, in order to maximize the gain and obtain a near cubic bound we need an explicit multilinear polynomial with *degree linear in the number of variables*, and that has close to the maximum possible shifted partials dimension even when restricted to an affine subspace. The polynomial family $\{f_n\}_{n\geq 1}$ in Theorem 1 is a variant of the family of Nisan-Wigderson polynomials used in [18, 15]. A notable difference between the Nisan-Wigderson families used in earlier works and the one used here is that the degree of f_n is linearly related to its number of variables, unlike $d = n^{o(1)}$ in previous works. Although, a greedy construction of a Nisan-Wigderson family can make degree $\Theta(n)$, it is not clear if such a family is in VNP. To ensure both – a VNP family and linear degree – we construct a family by 'composing' two smaller families of Nisan-Wigderson polynomials, one is obtained by a greedy algorithm and the other explicitly defined in [18, 15]. A detailed description of the polynomial family is given in Section 6.

Few more details on the polynomial families. Polynomial f_n in Theorem 1 is homogeneous with three sets of variables $\mathbf{u}, \mathbf{y}, \mathbf{x}$ such that $|\mathbf{u}| = |\mathbf{y}| = |\mathbf{x}| = \frac{10n}{9}$. (To avoid a few ceil and floor notations in the analysis, we shall assume without any loss of generality that n is divisible by $1872 = 9 \cdot 13 \cdot 16$.) Let $\mathbf{u} = \{u_1, \ldots, u_{\frac{10n}{9}}\}, \mathbf{y} = \{y_1, \ldots, y_{\frac{10n}{9}}\}$ and $\mathbf{x} = \{x_1, \ldots, x_{\frac{10n}{9}}\}$. Every monomial of f_n is a product of a $\mathbf{u}$-monomial of degree $d_{\mathbf{u}} = n$, a $\mathbf{y}$-monomial of degree $d_{\mathbf{y}} = \lfloor \ln n \rfloor$, and an $\mathbf{x}$-monomial of degree $d_{\mathbf{x}} \in \left[\frac{2n}{13}, \frac{n}{3}\right]$. Thus the number of variables and the degree of f_n are both $\Theta(n)$. The $\mathbf{x}$ and the $\mathbf{y}$ variables are the primary variables; derivatives of f_n of order $\lfloor \ln n \rfloor$ with respect to the $\mathbf{y}$-variables give rise to $\mathbf{x}$-monomials with large 'pairwise distance' that help estimate the shifted partials dimension of the target polynomial. The $\mathbf{u}$-variables are auxiliary variables which ensure that the measure remains high for the target polynomial even when restricted to an affine subspace.

The polynomial family $\{f_{n,d}\}$ used in Theorem 3 is a simple variant of the multi-r-ic iterated matrix multiplication polynomial family used in [19].

1.3 Organization

Sections 3 to 6 are devoted to the proofs of Theorem 1 and 2. We prove Theorem 3 in Section 7.

2 Preliminaries

2.1 Basic notations

For any $m \in \mathbb{N}$, the set of natural numbers, the set $\{1, \ldots, m\}$ is denoted by $[m]$. We use upper-case letters (like A or S) to denote sets of numbers, calligraphic upper-case letters (like $\mathcal{B}, \mathcal{D}$ or $\mathcal{L}$) to denote sets of polynomials, and bold lower-case letters (like $\mathbf{x}$ or $\mathbf{y}$) to denote sets of variables. When the base ring of polynomials is clear from the context, the ideal generated by a set of polynomials of the ring, say $\mathcal{L}$, is denoted by $\langle \mathcal{L} \rangle$. A circuit is denoted using typewriter font, as in `C` or `D`. For a set of numbers $S \subseteq [m]$, $\bar{S}$ denote the complement of S. Sometimes, we use the notation $\mathsf{poly}(n)$ to mean $n^{O(1)}$.

2.2 The measure

Although, the results in this paper can be derived using the shifted partials measure as it is in [14], we choose to work with a variant of this measure for better clarity in the analysis. This variant is similar in outlook to the *shifted skewed partials* measure used recently in [19],

although for our application there is no difference (or skew) between the number of $\mathbf{x}$ and $\mathbf{y}$ variables. Such a skew between $|\mathbf{y}|$ and $|\mathbf{x}|$ was important for the results in [19].

Let $A \subset \left[\frac{10n}{9}\right]$ of size $|A| = n$. Let $\mathbf{x}_A = \{x_i : i \in A\}$ and $g(\mathbf{y}, \mathbf{x}_A) \in \mathbb{F}[\mathbf{y}, \mathbf{x}_A]$. For $k, \ell \in \mathbb{N}$, define the measure $\mathrm{SP}_{k,\ell,A} : \mathbb{F}[\mathbf{y}, \mathbf{x}_A] \to \mathbb{N}$ as follows.

$$\mathrm{SP}_{k,\ell,A}(g) \stackrel{\text{def}}{=} \dim(\mathbf{x}_A^{\leq \ell} \cdot \sigma_{\mathbf{y}}(\boldsymbol{\partial}_{\mathbf{y}}^{=k} g)),$$

where $\boldsymbol{\partial}_{\mathbf{y}}^{=k} g$ is the set of all k-th order partial derivatives of g with respect to the $\mathbf{y}$-variables, and $\sigma_{\mathbf{y}} : \mathbb{F}[\mathbf{y}, \mathbf{x}_A] \to \mathbb{F}[\mathbf{x}_A]$ is a map that sets all the $\mathbf{y}$-variables to zero. Naturally, $\sigma_{\mathbf{y}}$ is a homomorphism from $\mathbb{F}[\mathbf{y}, \mathbf{x}_A]$ to $\mathbb{F}[\mathbf{x}_A]$, and $\sigma_{\mathbf{y}}(\mathcal{D})$ is defined by $\{\sigma_{\mathbf{y}}(h) : h \in \mathcal{D}\}$ for any set of polynomials $\mathcal{D} \subseteq \mathbb{F}[\mathbf{y}, \mathbf{x}_A]$. $\mathbf{x}_A^{\leq \ell}$ is the set of all monomials in the $\mathbf{x}_A$-variables of degree ℓ or less. For two sets of polynomials $\mathcal{B}$ and $\mathcal{D}$, $\mathcal{B}.\mathcal{D} \stackrel{\text{def}}{=} \{h_1.h_2 : h_1 \in \mathcal{B} \text{ and } h_2 \in \mathcal{D}\}$, and the dimension of a set of polynomials $\mathcal{D}$ (denoted by $\dim(\mathcal{D})$) is the dimension of the vector space spanned by the polynomials in $\mathcal{D}$ over the field $\mathbb{F}$.

It is worth noting that the above measure (as in [19]) can be thought of as a hybrid of the *rank of the partial derivatives matrix* measure of [28] and the shifted partials measure of [14]. The former measure has been refined and used in several other subsequent work, most notably in [31, 35], and is also identified with the *evaluation dimension* measure in [6] over fields of characteristic zero. The following proposition is easy to verify.

▶ **Proposition 4** (Sub-additivity). *For any $k, \ell \in \mathbb{N}$, $\mathbf{x}_A \subseteq \mathbf{x}$ and $g_1, g_2 \in \mathbb{F}[\mathbf{y}, \mathbf{x}_A]$,*

$$\mathrm{SP}_{k,\ell,A}(g_1 + g_2) \leq \mathrm{SP}_{k,\ell,A}(g_1) + \mathrm{SP}_{k,\ell,A}(g_2).$$

3 Lower bounding the measure for the target polynomial family

We will show that the measure SP (from Section 2.2) is considerably large when applied suitably to the polynomial family $\{f_n\}_{n \geq 1}$. The precise statement is given in the theorem below.

Polynomials restricted to an affine subspace. Let $S \subseteq \left[\frac{10n}{9}\right]$ be a set of size $\frac{n}{9}$ and

$$\mathcal{L}_S = \{x_i - h_i\}_{i \in S} \tag{1}$$

be a set of $|\mathcal{L}_S| = |S| = \frac{n}{9}$ linear polynomials in $\mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}]$ such that $h_i \in \mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}_{\bar{S}}]$ for every $i \in S$, where $\bar{S} = \left[\frac{10n}{9}\right] \setminus S$.

Denote the ideal of $\mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}]$ generated by the linear polynomials of $\mathcal{L}_S$ by $\langle \mathcal{L}_S \rangle$. For any polynomial $f \in \mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}]$, let

$$f_{\langle \mathcal{L}_S \rangle} \stackrel{\text{def}}{=} f \mod \langle \mathcal{L}_S \rangle$$

be the image of the polynomial f in the ring $\mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}]/\langle \mathcal{L}_S \rangle$. Since $\mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}]/\langle \mathcal{L}_S \rangle$ is isomorphic to $\mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}_{\bar{S}}]$, $f_{\langle \mathcal{L}_S \rangle}$ can be represented by a polynomial in the ring $\mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}_{\bar{S}}]$; this polynomial is obtained from f by replacing x_i by h_i for all $i \in S$. So, we will treat $f_{\langle \mathcal{L}_S \rangle}$ as an element of $\mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}_{\bar{S}}]$.

Finally, let $f_{\langle \mathcal{L}_S \rangle, \mathbf{u}_S = 0}$ be the polynomial obtained from $f_{\langle \mathcal{L}_S \rangle} \in \mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}_{\bar{S}}]$ by setting the $\mathbf{u}$-variables to 0/1-values as follows: $u_i = 0$ if $i \in S$, else $u_i = 1$. We will describe the family $\{f_n\}_{n \geq 1}$ and prove the following theorem in Section 6.

▶ **Theorem 5.** *Let n be the parameter that defines the polynomial family $\{f_n\}_{n\geq 1}$. Let $k = \lfloor \ln n \rfloor$, q be the smallest prime greater or equal to $\lceil \frac{n}{1000\cdot \ln n} \rceil$, $\ell = \left\lfloor \frac{n^2}{32\cdot k \cdot \ln q} \right\rfloor$. Then for every set $S \subseteq \left[\frac{10n}{9}\right]$ of size $|S| = \frac{n}{9}$, and every set of linear polynomials $\mathcal{L}_S$ as in Equation (1), and $f = f_n$,*

$$\mathrm{SP}_{k,\ell,\bar{S}}(f_{\langle \mathcal{L}_S\rangle, \mathbf{u}_S=0}) \geq \frac{1}{2}\cdot q^k \cdot \binom{n+\ell}{n}.$$

Next, we show an upperbound of the measure for a depth-3 circuit and prove Theorem 1.

4 Upper bounding the measure for a depth three circuit

Pruning 'heavy' product gates from a depth three circuit. Let $\mathtt{C} = \sum_{i=1}^{s} T_i$ be a depth three circuit computing $f = f_n$, where T_i is a product term[5] of $\mathtt{C}$. Let c_0 be a constant to be fixed later in the analysis. Then either of the following two cases is obviously true.

Case 1: The number of product terms of $\mathtt{C}$, with $\mathbf{x}$-degree greater or equal to $\left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$, is greater than $\frac{n}{9}$.

Case 2: The number of product terms of $\mathtt{C}$, with $\mathbf{x}$-degree greater or equal to $\left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$, is less than or equal to $\frac{n}{9}$.

If Case 1 is true then the multiplicative complexity of $\mathtt{C}$ is at least $\left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor \cdot \frac{n}{9} = \Omega(\frac{n^3}{(\ln n)^2})$ as $d_{\mathbf{x}} \in \left[\frac{2n}{13}, \frac{n}{3}\right]$ and we have nothing to prove in this case. If Case 2 is true then we can find a 'few' linear polynomials such that modulo these the circuit is free of 'heavy' product terms. This is stated formally in the lemma below and the corollary thereafter, and is directly inspired by a similar argument in [38, 36]. However, the threshold chosen to define 'heavy' product gates in [38, 36] is linear in n, whereas the one here has an extra $\frac{d_{\mathbf{x}}}{(\ln n)^2}$ factor that finally accounts for the improvement in the lower bound. As mentioned in Section 1, this is the leeway to the formal degree of the circuit provided by the analysis with shifted partials.

▶ **Lemma 6.** *Suppose the number of product terms of $\mathtt{C}$, with $\mathbf{x}$-degree greater or equal to $\left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$, is bounded by $\frac{n}{9}$. Then, there is a set $S \subseteq \left[\frac{10n}{9}\right]$ of size $\frac{n}{9}$ and a set of linear polynomials,*

$$\mathcal{L}_S = \{x_i - h_i\}_{i\in S}, \text{ where } h_i \text{ is a linear polynomial in } \mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}_{\bar{S}}] \text{ for every } i \in S,$$

such that $f_{\langle \mathcal{L}_S\rangle} \in \mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}_{\bar{S}}]$ is computed by a depth three circuit, say $\mathtt{C}_{\langle \mathcal{L}_S\rangle}$, satisfying the following:

1. *top fan-in of $\mathtt{C}_{\langle \mathcal{L}_S\rangle}$ is upper bounded by the top fan-in of $\mathtt{C}$,*
2. *every product term of $\mathtt{C}_{\langle \mathcal{L}_S\rangle}$ has $\mathbf{x}$-degree upper bounded by $\left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$.*

The proof of the lemma is relatively straightforward and and can be found in the full version paper.

▶ **Corollary 7.** *Polynomial $f_{\langle \mathcal{L}_S\rangle, \mathbf{u}_S=0} \in \mathbb{F}[\mathbf{y}, \mathbf{x}_{\bar{S}}]$ is computed by a depth three circuit, say $\mathtt{C}_{\langle \mathcal{L}_S\rangle, \mathbf{u}_S=0}$, with top fan-in bounded by the top fan-in of $\mathtt{C}$ and every product term of $\mathtt{C}_{\langle \mathcal{L}_S\rangle, \mathbf{u}_S=0}$ has $\mathbf{x}$-degree bounded by $\left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$.*

[5] a product term corresponds to a multiplication gate of $\mathtt{C}$

Let us denote the circuit $\mathtt{C}_{\langle\mathcal{L}_S\rangle,\mathbf{u}_S=0}$ by $\mathtt{D}$. Let $\mathtt{D} = \sum_{i=1}^{s} P_i$, where a term P_i is a product of linear polynomials in $\mathbb{F}[\mathbf{y}, \mathbf{x}_{\bar{S}}]$. Note that the pruned circuit $\mathtt{D}$ has only $\mathbf{y}$ and $\mathbf{x}_{\bar{S}}$ variables. Now, as the degrees of the intermediate gates are bounded, we can upperbound the shifted partial measure for $\mathtt{D}$ by using a grouping argument (as it was done in [19]). We get then the next lemma.

▶ **Lemma 8.** *Let $k, \ell \in \mathbb{N}$ be as in Theorem 5. Then*

$$\mathrm{SP}_{k,\ell,\bar{S}}(\mathtt{D}) \leq s \cdot \binom{\lceil 32c_0 d_{\mathbf{x}} \rceil}{k} \cdot \binom{n+\ell+kt}{n}, \text{ where } t = \left\lceil \frac{n}{32 \cdot (\ln n)^2} \right\rceil.$$

5 Putting together: Proof of Theorem 1

Let $\mathtt{C}$ be a depth three circuit computing f_n. Then, as explained in Section 4, we have two cases to handle. In Case 1, the multiplicative complexity of $\mathtt{C}$ is already $\Omega(\frac{n^3}{(\ln n)^2})$ and we have nothing to prove. Whereas, in Case 2, the circuit can be pruned of heavy product gates so that the polynomial $f_{\langle\mathcal{L}_S\rangle,\mathbf{u}_S=0} \in \mathbb{F}[\mathbf{y}, \mathbf{x}_{\bar{S}}]$ is computed by a depth three circuit, say $\mathtt{D}$, whose top fan-in is upper bounded by the top fan-in of $\mathtt{C}$ (by Corollary 7). Moreover, every product term of $\mathtt{D}$ has $\mathbf{x}$-degree bounded by $\left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$ so that Lemma 8 is applicable now. The computations of the next lemma can be found in the full version paper.

▶ **Lemma 9.** *In Case 2, the top fan-in of $\mathtt{D}$ (hence also the top fan-in of $\mathtt{C}$) is $\omega(n^3)$.*

Proof. By Theorem 5 and Lemma 8, the top fan-in s of $\mathtt{D}$ can be lower bounded as follows:

$$s \geq \frac{1}{2} \cdot \frac{q^k \cdot \binom{n+\ell}{n}}{\binom{\lceil 32c_0 d_{\mathbf{x}} \rceil}{k} \cdot \binom{n+\ell+kt}{n}},$$

which will imply after some computations that $s = \omega(n^3)$. ◀

Thus, in Case 2, the top fan-in of $\mathtt{D}$ (and hence $\mathtt{C}$) must be $\omega(n^3)$ and therefore putting Case 1 and 2 together, the multiplicative complexity of $\mathtt{C}$ is $\min\{\Omega(\frac{n^3}{(\ln n)^2}), \omega(n^3)\} = \Omega(\frac{n^3}{(\ln n)^2})$ for sufficiently large n.

5.1 Proof of Theorem 2

The proof follows from Lemma 9. Suppose $f = f_n$ is computed by a symmetric circuit where $l_1, \ldots, l_m$ are the bottom level linear polynomials. Naturally, $f = \mathrm{ESym}_m^d(l_1, \ldots, l_m)$ for some d, and hence (by Ben-Or's interpolation trick over any field of size more than m) f is also computed by a depth three circuit $\mathtt{C}$ with top fan-in $m+1$ and degree of every product term bounded by m. If $m \geq \left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$ then we have nothing to prove. Suppose $m < \left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$. Then the condition of Case 2 (in Section 4) is satisfied as *every* product term of $\mathtt{C}$ has $\mathbf{x}$-degree (in fact, total degree) bounded by $m < \left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor$. But then, Lemma 9 tells us that $\mathtt{C}$ has top fan-in $\omega(n^3)$ which contradicts with the fact that the top fan-in is $m+1 = O(n^2)$. So, it must be that $m \geq \left\lfloor \frac{c_0 n d_{\mathbf{x}}}{(\ln n)^2} \right\rfloor = \Omega\left(\frac{n^2}{(\ln n)^2}\right)$.

6 The polynomial family and proof of Theorem 5

6.1 Construction of the Nisan-Wigderson polynomial family

Let $\mathbf{z} = \{z_1, \ldots, z_n\}$ be a set of n formal variables. For any two multilinear monomials m_1 and m_2 in the $\mathbf{z}$-variables of degree $d_{\mathbf{z}}$ each, let $|m_1 \cap m_2|$ be the number of variables common

between m_1 and m_2. Define *distance* between the monomials m_1, m_2 as,

$$\Delta(m_1, m_2) \stackrel{\text{def}}{=} d_{\mathbf{z}} - |m_1 \cap m_2|.$$

As in the statement of Theorem 5, let q be the smallest prime greater or equal to $\left\lceil \frac{n}{1000 \ln n} \right\rceil$ and $k = \lfloor \ln n \rfloor$. The following lemma plays a central role in the construction of the polynomial family $\{f_n\}_{n \geq 1}$.

▶ **Lemma 10.** *There is a family $\{g_n(\mathbf{z})\}_{n \geq 1}$ in* VNP *such that $g_n(\mathbf{z})$ is a homogeneous multilinear polynomial of degree $d_{\mathbf{z}} \in \left[\frac{2n}{13}, \frac{n}{3}\right]$ in n $\mathbf{z}$-variables, and $\Delta(m_1, m_2) \geq \frac{n}{16}$ for any pair of distinct monomials m_1 and m_2 of g_n. Further, g_n is a sum of q^k distinct monomials.*

In the previous lemma, we need to find a family whose monomials are pairwise distant, such that the degree is linear in the number of variables and the family is in VNP. The Nisan-Wigderson polynomial family in [18] is in VNP but its degree is not linear. On the other hand, one can greedily get a family such that the degree is linear but which is not known to be in VNP. We show in the full version paper that by 'composing' these two families one can get both the desired properties.

The family $\{f_n\}_{n \geq 1}$. Let $(m_1, \ldots, m_{q^k})$ be an ordered sequence of monomials of the polynomial $g_n(\mathbf{z})$ from the above lemma under lexicographic monomial ordering $z_1 \succ \ldots \succ z_n$. Let $\mathbf{w} = \{w_1, \ldots, w_n\}$ be n formal variables different from $\mathbf{z}$. The number of multilinear monomials in $\mathbf{w}$-variables of degree k is $\binom{n}{k} \geq (\frac{n}{k})^k = \left(\frac{n}{\lfloor \ln n \rfloor}\right)^k \geq q^k$. Under lexicographic monomial ordering $w_1 \succ \ldots \succ w_n$, let $(\beta_1, \ldots, \beta_{q^k})$ be the ordered sequence of the first q^k monomials among all multilinear monomials in the $\mathbf{w}$-variables of degree k. Define the polynomial $F_n(\mathbf{w}, \mathbf{z})$ as,

$$F_n(\mathbf{w}, \mathbf{z}) \stackrel{\text{def}}{=} \sum_{j=1}^{q^k} \beta_j m_j. \tag{2}$$

Now let $\mathbf{u} = \{u_1, \ldots, u_{\frac{10n}{9}}\}$, $\mathbf{y} = \{y_1, \ldots, y_{\frac{10n}{9}}\}$ and $\mathbf{x} = \{x_1, \ldots, x_{\frac{10n}{9}}\}$ be the sets of variables on which $f_n(\mathbf{u}, \mathbf{y}, \mathbf{x})$ is defined as follows.

$$f_n(\mathbf{u}, \mathbf{y}, \mathbf{x}) \stackrel{\text{def}}{=} \sum_{\substack{A \subseteq \left[\frac{10n}{9}\right] \\ |A| = n}} \prod_{i \in A} u_i \cdot F_n(\mathbf{y}_A, \mathbf{x}_A). \tag{3}$$

We assume the lexicographic order $x_1 \succ \ldots \succ x_{\frac{10n}{9}}$ and $y_1 \succ \ldots \succ y_{\frac{10n}{9}}$. The polynomial $F_n(\mathbf{y}_A, \mathbf{x}_A)$ is obtained by substituting the $\mathbf{y}_A$-variables $\{y_i : i \in A\}$ in place of the $\mathbf{w}$-variables and $\mathbf{x}_A$-variables $\{x_i : i \in A\}$ in place of the $\mathbf{z}$-variables such that the underlying lexicographic orders, $z_1 \succ \ldots \succ z_n$ and $w_1 \succ \ldots \succ w_n$, are obeyed. Note that $d_{\mathbf{y}} = \deg_{\mathbf{y}} f_n = k$, $d_{\mathbf{u}} = \deg_{\mathbf{u}} f_n = n$ and $d_{\mathbf{x}} = \deg_{\mathbf{x}} f_n = \deg_{\mathbf{z}} g_n = d_{\mathbf{z}} \in \left[\frac{2n}{13}, \frac{n}{3}\right]$. Further, the polynomial family $\{f_n\}_{n \geq 1}$ is in VNP: It would be clear from the proof of Lemma 10 that the computational problem of finding the 'index' of a given monomial in g_n can be solved in $\mathsf{poly}(n)$ time, which in turn implies the coefficient of a given monomial in f_n can be found in $\mathsf{poly}(n)$ time. The *index* of a monomial m in g_n is the position of m in the lexicographically ordered list of q^k monomials of g_n.

6.2 Proof of Theorem 5: The measure on the polynomial family

In this section, we show that the relevant measure is high for the family of polynomials (defined in Equation 3) even when restricted to an affine subspace. As in Section 3 (Equation

1), let $S \subseteq \left[\frac{10n}{9}\right]$ be a set of size $\frac{n}{9}$. Let $\mathcal{L}_S = \{x_i - h_i\}_{i \in S}$ be any set of $\frac{n}{9}$ linear polynomials in $\mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}]$ such that $h_i \in \mathbb{F}[\mathbf{u}, \mathbf{y}, \mathbf{x}_{\bar{S}}]$ for every $i \in S$, where $\bar{S} = \left[\frac{10n}{9}\right] \setminus S$. Let $f = f_n(\mathbf{u}, \mathbf{y}, \mathbf{x})$ (as defined in Equation 3).

▶ **Observation 11.** $f_{\langle \mathcal{L}_S \rangle, \mathbf{u}_S = 0} = F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}}) \in \mathbb{F}[\mathbf{y}, \mathbf{x}_{\bar{S}}]$.

Proof. The polynomial $f_{\langle \mathcal{L}_S \rangle, \mathbf{u}_S = 0}$ is obtained from f by substituting every x_i by h_i for every $i \in S$, and then setting $u_j = 0$ for every $j \in S$ and $u_j = 1$ otherwise. Since the only $\mathbf{x}$-variables occurring in $F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}})$ are from $\mathbf{x}_{\bar{S}}$, it remains untouched by the above substitutions. Finally, the setting of the $\mathbf{u}$-variables retains only $F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}})$ from the sum in Equation 3. ◀

So, we need to show that

$$\mathrm{SP}_{k,\ell,\bar{S}}(F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}})) \geq \frac{1}{2} \cdot q^k \cdot \binom{n+\ell}{n}.$$

This part of the argument bears close resemblance to and is inspired by similar arguments in [7, 5]. We begin with the following observation.

▶ **Observation 12.** *The set $\boldsymbol{\partial}_{\mathbf{y}}^{=k} F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}})$ consists of exactly the monomials of $g_n(\mathbf{x}_{\bar{S}})$. Hence, $\sigma_{\mathbf{y}}(\boldsymbol{\partial}_{\mathbf{y}}^{=k} F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}}))$ also consists of exactly the monomials of $g_n(\mathbf{x}_{\bar{S}})$.*

Proof. Follows easily from the definition of the polynomial F_n in Equation 2. ◀

Reusing notation, let the monomials of $g_n(\mathbf{x}_{\bar{S}})$ be $\{m_1, \ldots, m_{q^k}\}$ – these are monomials in $\mathbf{x}_{\bar{S}}$-variables. By Lemma 10, $\Delta(m_i, m_j) \geq \frac{n}{16}$ for every $i \neq j$ and $\frac{2n}{13} \leq \deg(m_i) \leq \frac{n}{3}$ for every $i \in [q^k]$. Let

$$B_i \stackrel{\text{def}}{=} \mathbf{x}_{\bar{S}}^{\leq \ell} \cdot m_i, \qquad \text{for } i \in [q^k].$$

Then,

$$\begin{aligned} \dim(\mathbf{x}_{\bar{S}}^{\leq \ell} \cdot \sigma_{\mathbf{y}}(\boldsymbol{\partial}_{\mathbf{y}}^{=k} F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}}))) &= |B_1 \cup \ldots \cup B_{q^k}| \\ \Rightarrow \mathrm{SP}_{k,\ell,\bar{S}}(F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}})) &\geq \sum_{i=1}^{q^k} |B_i| - \frac{1}{2} \cdot \sum_{\substack{i,j \\ i \neq j}} |B_i \cap B_j| \\ &= q^k \cdot \binom{n+\ell}{n} - \frac{1}{2} \cdot \sum_{\substack{i,j \\ i \neq j}} |B_i \cap B_j|, \end{aligned} \tag{4}$$

as $|\bar{S}| = n$ and $|B_i| = \binom{n+\ell}{n}$.

▶ **Proposition 13.** *For every $i, j \in [q^k]$ and $i \neq j$, $|B_i \cap B_j| \leq \binom{n+\ell-n/16}{n}$.*

Proof. If a monomial m belongs to both B_i and B_j then $m = s_1 \cdot m_i = s_2 \cdot m_j$ where $\deg(s_1), \deg(s_2) \leq \ell$. Since $\Delta(m_i, m_j) \geq n/16$,

$$m = s' \cdot \frac{m_j}{\gcd(m_i, m_j)} \cdot m_i, \qquad \text{where } \deg(s') \leq \ell - \frac{n}{16}.$$

Hence, the number of such monomials m is bounded by $\binom{n+\ell-n/16}{n}$. ◀

Therefore, by Equation (4), after some computations.

▶ **Claim 14.** $\mathrm{SP}_{k,\ell,\bar{S}}(F_n(\mathbf{y}_{\bar{S}}, \mathbf{x}_{\bar{S}})) \geq \frac{1}{2} \cdot q^k \cdot \binom{n+\ell}{n}$.

This completes the proof of Theorem 5.

7 Homogeneous depth three circuits with large degree

We prove Theorem 3 in this section. The measure remains the same as before, but the notation is simplified a little bit (as we do not need to include a subset of variables in the definition of the measure). For any $g \in \mathbb{F}[\mathbf{y}, \mathbf{x}]$, define the measure $\mathrm{SP}_{k,\ell} : \mathbb{F}[\mathbf{y}, \mathbf{x}] \to \mathbb{N}$ as

$$\mathrm{SP}_{k,\ell}(g) \stackrel{\text{def}}{=} \dim(\mathbf{x}^{\leq \ell} \cdot \sigma_{\mathbf{y}}(\boldsymbol{\partial}_{\mathbf{y}}^{=k} g)).$$

Like before, the measure is sub-additive, i.e. for $g_1, g_2 \in \mathbb{F}[\mathbf{y}, \mathbf{x}]$ and $k, \ell \in \mathbb{N}$,

$$\mathrm{SP}_{k,\ell}(g_1 + g_2) \leq \mathrm{SP}_{k,\ell}(g_1) + \mathrm{SP}_{k,\ell}(g_2).$$

Moreover, the measure is invariant under multiplication by any fixed polynomial from $\mathbb{F}[\mathbf{x}]$ (the proof of the following lemma is very simple and appears in the full version paper):

▶ **Lemma 15.** *For any $g \in \mathbb{F}[\mathbf{y}, \mathbf{x}]$, $h \in \mathbb{F}[\mathbf{x}]$ and $k, \ell \in \mathbb{N}$, $\mathrm{SP}_{k,\ell}(h \cdot g) = \mathrm{SP}_{k,\ell}(g)$.*

The outline of the proof of Theorem 3 also remains the same: we show a suitable upper bound on the measure for the circuit, and a lower bound for the target family of polynomials. The target family of polynomials is basically a *multi-r-ic* variant of the iterated matrix multiplication polynomial defined and analysed in [19] – we will recall some parts of the analysis from there to lower bound the measure for the family of polynomials. Furthermore, this polynomial can be computed by an algebraic branching program of size polynomial in the number of variables and degree of the polynomial.

7.1 Upper bound for the circuit

Let $\mathtt{C}$ be any homogeneous depth three circuit computing a polynomial in n variables $\mathbf{y} \uplus \mathbf{x}$ and of degree d. More precisely, by identifying the circuit with the polynomial it computes, $\mathtt{C} = T_1 + T_2 + \ldots + T_s$, where the T_i's are products of d homogeneous linear polynomials i.e. $T_i = l_{i1} \cdot l_{i2} \cdot \ldots \cdot l_{id}$, where every l_{ij} is a linear form. Let us consider any one product term, say T. By grouping t linear forms together and multiplying the linear forms within each group, we obtain $T = Q_1 \cdot \ldots \cdot Q_{\lceil \frac{d}{t} \rceil}$, where $\deg(Q_j) \leq t$ for every $j \in \left[\lceil \frac{d}{t} \rceil\right]$. By sub-additivity of the measure and following a similar argument as in the proof of Lemma 8, we get the following lemma.

▶ **Lemma 16.** *For any $k, \ell \in \mathbb{N}$ and $t \leq d$,*

$$\mathrm{SP}_{k,\ell}(\mathtt{C}) \leq s \cdot \binom{\lceil \frac{d}{t} \rceil}{k} \cdot \binom{|\mathbf{x}| + \ell + kt}{|\mathbf{x}|}. \tag{5}$$

7.2 Lower bound for the polynomial family

The polynomial family. We define a polynomial on n variables $\mathbf{y} \uplus \mathbf{x}$ and of degree d, where d is any integer greater or equal to n.

For $w, k, r, \alpha \in \mathbb{N}$, consider the following polynomial.

$$F_{w,k,r,\alpha}(\mathbf{y}, \mathbf{x}) \stackrel{\text{def}}{=} g_1(\mathbf{y}_1, \mathbf{x}_1) \cdot g_2(\mathbf{y}_2, \mathbf{x}_2) \cdot \ldots \cdot g_k(\mathbf{y}_k, \mathbf{x}_k),$$

where the g_i's are polynomials over the indicated (disjoint) subsets of variables $\mathbf{y} = \mathbf{y}_1 \uplus \ldots \uplus \mathbf{y}_k$ and $\mathbf{x} = \mathbf{x}_1 \uplus \ldots \uplus \mathbf{x}_k$, and defined as,

$$g_i(\mathbf{y}_i, \mathbf{x}_i) \stackrel{\text{def}}{=} \sum_{a,b \in [w]} y_{i,a,b} \cdot \prod_{c \in [\alpha]} x_{i,c,a}^r \cdot x_{i,c+\alpha,b}^r.$$

The number of $\mathbf{y}$-variables is $|\mathbf{y}| = kw^2$ and the number of $\mathbf{x}$-variables is $|\mathbf{x}| = 2k\alpha w$. The total number of variables in $F_{w,k,r,\alpha}$ is $(w^2 + 2\alpha w) \cdot k$, and it has degree $\tilde{d} = (2\alpha r + 1) \cdot k$. Our target polynomial is almost $F_{w,k,r,\alpha}$, except that we multiply it with a suitable power of a variable just to match its degree with the given degree parameter d which is any number more than the number of variables.

Let $n = (w^2 + 2\alpha w) \cdot k$ and $d \geq n$ be a given degree parameter. In the analysis, we eventually fix α and w to integer constants so that $n = \Theta(k)$. Set $r = \lceil \frac{d}{3\alpha k} \rceil$ and x be any arbitrarily fixed variable in $\mathbf{x}$. Our polynomial family $\{f_{n,d}\}$ is defined by

$$f_{n,d} \stackrel{\text{def}}{=} x^{d-\tilde{d}} \cdot F_{w,k,r,\alpha}. \tag{6}$$

This polynomial is well defined, i.e. $d \geq \tilde{d}$, as soon as $w \geq 3$. Observe that $f_{n,d}$ has the same set of n variables as $F_{w,k,r,\alpha}$ and has degree d. Let us record the values for k and r for the analysis later $k = \frac{n}{w^2+2\alpha w}$ and $r = \lceil \frac{d}{3\alpha k} \rceil$. Also, note that $f_{n,d}$ can be computed by a $\mathsf{poly}(n, d)$-size ABP.

The measure on the polynomial family. The following lemma was essentially proved in [19] (see Section 7.5 in there) with slightly different notations.

▶ **Lemma 17.** *Let $0 < \delta \leq 1/5$ be a constant and $w \geq 3$.*

1. *Then*

$$\mathrm{SP}_{k,\ell}(F_{w,k,r,\alpha}) \geq M \cdot \binom{|\mathbf{x}| + \ell}{|\mathbf{x}|} - \frac{M^2}{2} \cdot \binom{|\mathbf{x}| + \ell - \lceil \delta k \rceil \cdot \alpha r}{|\mathbf{x}|} \text{ with } M = \left(\left\lfloor \frac{w^{2-\delta}}{2} \right\rfloor \right)^k.$$

2. *Moreover, if $\ell \geq |\mathbf{x}|$ and $2 \cdot |\mathbf{x}| \cdot \alpha r \geq \ell\beta \cdot \ln w$ where $\beta \geq 4(2-\delta)/\delta$ is a constant then we can also conclude that $\mathrm{SP}_{k,\ell}(F_{w,k,r,\alpha})$ is lower bounded by $M \cdot \binom{|\mathbf{x}|+\ell}{|\mathbf{x}|}/2$.*

For the choice of parameters below, $\frac{w^{2-\delta}}{2}$ is an integer. Hence, $M = \left(\frac{w^{2-\delta}}{2}\right)^k$.

▶ **Corollary 18.** *If the conditions of Lemma 17 are satisfied then it follows from Lemma 15*

$$\mathrm{SP}_{k,\ell}(f_{n,d}) \geq \frac{M}{2} \cdot \binom{|\mathbf{x}| + \ell}{|\mathbf{x}|}.$$

7.3 Putting together: Proof of Theorem 3

Let us choose $t = \lfloor 2\varepsilon\alpha r \rfloor$, and $\ell = \left\lfloor \frac{|\mathbf{x}| \cdot t}{\varepsilon\beta \cdot \ln w} \right\rfloor$ with the following parameters $\alpha = 18$, $\delta = \frac{1}{5}$, $\beta = 36$, $\varepsilon = \frac{1}{200}$, and $w = 2^{10}$.

We can notice that $t > 0$ and $\lceil d/t \rceil \leq (2k)/\varepsilon$. Furthermore, the conditions $\ell \geq |\mathbf{x}|$, $2 \cdot |\mathbf{x}| \cdot \alpha r \geq \ell\beta \cdot \ln w$, and $\beta \geq 4(2-\delta)/\delta$ are satisfied. Hence, if $\mathtt{C}$ is a homogeneous depth three circuit computing $f_{n,d}$, then by Lemma 16 and Corollary 18,

$$s \cdot \binom{\lceil \frac{d}{t} \rceil}{k} \cdot \binom{|\mathbf{x}| + \ell + kt}{|\mathbf{x}|} \geq \mathrm{SP}_{k,\ell}(f_{n,d}) \geq \frac{M}{2} \cdot \binom{|\mathbf{x}| + \ell}{|\mathbf{x}|}.$$

It implies, after some computations that $s \geq 2^{\Omega(n)}$.

References

1. Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. Jacobian hits circuits: hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits. In *STOC*, pages 599–614, 2012.
2. Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 67–75, 2008.
3. W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22(3):317–330, 1983.
4. Peter Bürgisser. *Completeness and Reduction in Algebraic Complexity Theory*. Springer, 2000.
5. Suryajith Chillara and Partha Mukhopadhyay. Depth-4 lower bounds, determinantal complexity: A unified approach. In *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, pages 239–250, 2014. doi:10.4230/LIPIcs.STACS.2014.239.
6. Michael A. Forbes and Amir Shpilka. Quasipolynomial-Time Identity Testing of Noncommutative and Read-Once Oblivious Algebraic Branching Programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013.
7. Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for depth 4 formulas computing iterated matrix multiplication. In *STOC*, pages 128–135, 2014.
8. Dima Grigoriev and Marek Karpinski. An exponential lower bound for depth 3 arithmetic circuits. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 577–582, 1998.
9. Dima Grigoriev and Alexander A. Razborov. Exponential complexity lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. In *39th Annual Symposium on Foundations of Computer Science, FOCS'98, November 8-11, 1998, Palo Alto, California, USA*, pages 269–278, 1998.
10. Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth three. In *Foundations of Computer Science (FOCS)*, pages 578–587, 2013.
11. Ankit Gupta, Neeraj Kayal, Pritish Kamath, and Ramprasad Saptharishi. Approaching the chasm at depth four. In *Conference on Computational Complexity (CCC)*, pages 65–73, 2013.
12. Maurice J. Jansen and Kenneth W. Regan. "Resistant" Polynomials and Stronger Lower Bounds for Depth-Three Arithmetical Formulas. In *Computing and Combinatorics, 13th Annual International Conference, COCOON 2007, Banff, Canada, July 16-19, 2007, Proceedings*, pages 470–481, 2007.
13. Mark Jerrum and Marc Snir. Some exact complexity results for straight-line computations over semirings. *J. ACM*, 29(3):874–897, 1982.
14. Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:81, 2012.
15. Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. An Exponential Lower Bound for Homogeneous Depth Four Arithmetic Formulas. In *Foundations of Computer Science (FOCS)*, pages 61–70, 2014.
16. Neeraj Kayal and Chandan Saha. Lower Bounds for Depth Three Arithmetic Circuits with small bottom fanin. In *Conference on Computational Complexity*, pages 158–208, 2015.

17 Neeraj Kayal and Chandan Saha. Multi-k-ic depth three circuit lower bound. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30, pages 527–539, 2015.

18 Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *STOC*, pages 146–153, 2014.

19 Neeraj Kayal, Chandan Saha, and Sébastien Tavenas. On the size of homogeneous and of depth four formulas with low individual degree. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:181, 2015.

20 Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448:56–65, 2012.

21 Mrinal Kumar and Ramprasad Saptharishi. An exponential lower bound for homogeneous depth-5 circuits over finite fields. *CoRR*, abs/1507.00177, 2015. URL: `http://arxiv.org/abs/1507.00177`.

22 Mrinal Kumar and Shubhangi Saraf. On the power of homogeneous depth 4 arithmetic circuits. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 364–373, 2014.

23 Mrinal Kumar and Shubhangi Saraf. Arithmetic circuits with locally low algebraic rank. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:194, 2015.

24 Mrinal Kumar and Shubhangi Saraf. Sums of products of polynomials in few variables : lower bounds and polynomial identity testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:71, 2015.

25 Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for non-commutative skew circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:22, 2015.

26 Meena Mahajan. Algebraic complexity classes. *CoRR*, abs/1307.3863, 2013. URL: `http://arxiv.org/abs/1307.3863`.

27 Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *J. Complex.*, 24(1):16–38, 2008.

28 Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *STOC*, pages 410–418, 1991.

29 Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1996.

30 Ran Raz. Separation of multilinear circuit and formula size. *Theory of Computing*, 2(1):121–135, 2006.

31 Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2), 2009.

32 Ran Raz. Elusive functions and lower bounds for arithmetic circuits. *Theory of Computing*, 6(1):135–177, 2010.

33 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 659–666, 2010.

34 Ran Raz and Amir Yehudayoff. Balancing syntactically multilinear arithmetic circuits. *Computational Complexity*, 17(4):515–535, 2008.

35 Ran Raz and Amir Yehudayoff. Lower Bounds and Separations for Constant Depth Multilinear Circuits. *Computational Complexity*, 18(2):171–207, 2009.

36 A. Shpilka and A. Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001.

37 Amir Shpilka. Affine projections of symmetric polynomials. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 160–171, 2001.

38 Amir Shpilka and Avi Wigderson. Depth-3 arithmetic formulae over fields of characteristic zero. In *IEEE Conference on Computational Complexity*, pages 87–, 1999. Available at `http://eccc.hpi-web.de/report/1999/023/`.

39 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

40 V. Strassen. Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten. *Numerische Mathematik*, 20:238–251, 1973.

41 Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. In *MFCS*, pages 813–824, 2013.

42 L. G. Valiant. Completeness Classes in Algebra. In *STOC'79: Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 249–261, New York, NY, USA, 1979. ACM Press.

43 L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):641–644, 1983.

Boundaries of VP and VNP

Joshua A. Grochow[*1], **Ketan D. Mulmuley**[†2], **and Youming Qiao**[‡3]

1 **Santa Fe Institute, Santa Fe, USA**
jgrochow@santafe.edu
2 **University of Chicago, Chicago, USA**
mulmuley@uchicago.edu
3 **University of Technology Sydney, Sydney, Australia**
Youming.Qiao@uts.edu.au

Abstract

One fundamental question in the context of the geometric complexity theory approach to the VP vs. VNP conjecture is whether VP = $\overline{\text{VP}}$, where VP is the class of families of polynomials that can be computed by arithmetic circuits of polynomial degree and size, and $\overline{\text{VP}}$ is the class of families of polynomials that can be approximated infinitesimally closely by arithmetic circuits of polynomial degree and size. The goal of this article is to study the conjecture in (Mulmuley, FOCS 2012) that $\overline{\text{VP}}$ is not contained in VP.

Towards that end, we introduce three degenerations of VP (i.e., sets of points in $\overline{\text{VP}}$), namely the stable degeneration Stable-VP, the Newton degeneration Newton-VP, and the p-definable one-parameter degeneration VP*. We also introduce analogous degenerations of VNP. We show that Stable-VP $\subseteq$ Newton-VP $\subseteq$ VP* $\subseteq$ VNP, and Stable-VNP = Newton-VNP = VNP* = VNP. The three notions of degenerations and the proof of this result shed light on the problem of separating $\overline{\text{VP}}$ from VP.

Although we do not yet construct explicit candidates for the polynomial families in $\overline{\text{VP}} \setminus$VP, we prove results which tell us where not to look for such families. Specifically, we demonstrate that the families in Newton-VP $\setminus$VP based on semi-invariants of quivers would have to be non-generic by showing that, for many finite quivers (including some wild ones), Newton degeneration of any generic semi-invariant can be computed by a circuit of polynomial size. We also show that the Newton degenerations of perfect matching Pfaffians, monotone arithmetic circuits over the reals, and Schur polynomials have polynomial-size circuits.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases geometric complexity theory, arithmetic circuit, border complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.34

1 Introduction

One fundamental question in the context of the geometric complexity theory (GCT) approach (cf. [22, 23], [5], and [21]) to the VP vs. VNP conjecture in Valiant [27] is whether VP = $\overline{\text{VP}}$, where VP is the class of families of polynomials that computed by arithmetic circuits of polynomial degree and size, VNP is the class of p-definable families of polynomials, and $\overline{\text{VP}}$ is the class of families of polynomials that can be approximated infinitesimally closely by arithmetic circuits of polynomial degree and size. We assume in what follows that the

* During this work, J. A. G. was supported by an SFI Omidyar Fellowship.
† K. D. M. was supported by the NSF grant CCF-1017760.
‡ Y. Q. was supported by the Australian Research Council DECRA DE150100720.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 34; pp. 34:1–34:14

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

circuits are over an algebraically closed field $\mathbb{F}$. We call $\overline{\text{VP}}$ the *closure* of VP, and $\overline{\text{VP}} \setminus \text{VP}$ the *boundary* of VP. So the question is whether this boundary is non-empty. At present, it is not even known if $\overline{\text{VP}}$ is contained in VNP.

The VP vs. $\overline{\text{VP}}$ question is important for two reasons. First, all known algebraic lower bounds for the exact computation of the permanent also hold for its infinitesimally close approximation. For example, the known quadratic lower bound for the permanent [20] also holds for its infinitesimally closely approximation [18], and so also the known lower bounds in the algebraic depth-three circuit models [14]; cf. App. B in [11] for a survey of the known lower bounds which emphasizes this point. These lower bounds hold because some algebraic, polynomial property that is satisfied by the coefficients of the polynomials computed by the circuits in the restricted class under consideration is not satisfied by the coefficients of the permanent. Since a polynomial property is a closed condition,[1] the same property is also satisfied by the coefficients of the polynomials that can be approximated infinitesimally closely[2] by circuits in the restricted class under consideration. This is why the same lower bound also holds for infinitesimally close approximation. We expect the same phenomenon to hold in the unrestricted algebraic circuit model as well. Hence, it is natural to expect that any realistic proof of the VP $\neq$ VNP conjecture will also show that VNP $\not\subseteq \overline{\text{VP}}$, as conjectured in [22] (note that if VNP $\not\subseteq \overline{\text{VP}}$ then there exists a polynomial property showing this lower bound). This is, in fact, the underlying thesis of geometric complexity theory that is implicit in [21]. But, if $\overline{\text{VP}} \neq \text{VP}$, as conjectured in [21], this would mean that any realistic approach to the VP vs. VNP conjecture would even have to separate the permanent from the families in $\overline{\text{VP}} \setminus \text{VP}$ with high circuit complexity.[3]

Second, it is shown in [21] that, assuming a stronger form of the VNP $\not\subseteq \overline{\text{VP}}$ conjecture, the problem NNL (short for Noether's Normalization Lemma) of computing Noether normalization of explicit varieties can be brought down from EXPSPACE, where it is currently, to P, ignoring a quasi-prefix. The existing EXPSPACE vs. P gap,[4] called the geometric complexity theory (GCT) chasm [21], in the complexity of NNL may be viewed as the common cause and measure of the difficulty of the fundamental problems in geometry (NNL) and complexity theory (Hardness). If $\overline{\text{VP}} = \text{VP}$, then it follows [21] that NNL is in PSPACE. Thus the conjectural inequality between $\overline{\text{VP}}$ and VP is the main difficulty that needs to be overcome to bring NNL from EXPSPACE to PSPACE unconditionally, and is the main reason why the standard techniques in complexity theory may not be expected to work in the context of the VP $\neq$ VNP conjecture.

The goal of this article is to study the conjecture in [21] that $\overline{\text{VP}}$ is not contained in VP.

1.1 Degenerations of VP and VNP

Towards that end, we introduce three notions of degenerations of VP and VNP; "degeneration" is the standard term in algebraic geometry for a limit point or infinitesimal approximation. These are subclasses of $\overline{\text{VP}}$ and $\overline{\text{VNP}}$, respectively; cf. Sec. 3 for formal definitions.

---

[1] It is defined by the vanishing of a continuous function, namely, a (meta) polynomial.

[2] This means the polynomials are the limits of the polynomials computed by the circuits in the restricted class under consideration.

[3] Although some lower bounds techniques in the restricted models do distinguish between different polynomials with high circuit complexity (e.g., [25]), we need a better understanding of the families in $\overline{\text{VP}} \setminus \text{VP}$ in order to know which techniques in this spirit could even potentially be useful in the setting of the VNP versus $\overline{\text{VP}}$ problem.

[4] Or, the EXPH vs. P gap, assuming the Generalized Riemann Hypothesis.

The first notion is that of a stable degeneration. Recall [24] that a polynomial f in $\mathbb{F}[x_1, \ldots, x_n]$ is called *stable* with respect to the natural action of $G = \mathrm{SL}(n, \mathbb{F})$ on $\mathbb{F}[x_1, \ldots, x_n]$ if the G-orbit of f is closed (in the Zariski topology). We say that a polynomial f is a *stable degeneration* of $g \in \mathbb{F}[x_1, \ldots, x_n]$ if f lies in a closed G-orbit (which is unique [24]) in the closure of the G-orbit of g. The degeneration is called stable since f in this case is stable. For any class of polynomials $\mathcal{C}$, the class Stable-$\mathcal{C}$ is defined to be the class of families of polynomials that are either in $\mathcal{C}$ or are stable degenerations thereof.

The second notion is that of a Newton degeneration. We say that a polynomial f is a *Newton degeneration* of g if it is obtained from g by keeping only those terms whose associated monomial-exponents lie in some specified face of the Newton polytope of g. For any class of polynomial families $\mathcal{C}$, the class Newton-$\mathcal{C}$ is defined to be the class of families of polynomials that are Newton degenerations of the polynomials in $\mathcal{C}$, or are linear projections of such Newton degenerations.[5]

The third notion, motivated by the notion of p-definability in Valiant [27], is that of a p-definable one-parameter degeneration. We say that a family $\{f_n\}$ of polynomials is a *p-definable one-parameter degeneration* of a family $\{g_n\}$ of polynomials, if $f_n = \lim_{t \to 0} g_n(t)$, where $g_n(t)$ is obtained from g_n by transforming its variables linearly such that (1) the entries of the linear transformation matrix are Laurent polynomials in t of possibly exponential degree (in n), and (2) there exists a small circuit C_n of size polynomial in n such that any coefficient of the Laurent polynomial in any entry of the transformation matrix can be obtained by evaluating C_n at the indices of that entry and the index of the coefficient.[6] Thus a p-definable one-parameter degeneration is a one-parameter degeneration of exponential degree that can be encoded by a small circuit. For any class $\mathcal{C}$, the class $\mathcal{C}^*$ is then defined to be the class of families of polynomials that are p-definable one-parameter degenerations of the families in $\mathcal{C}$.

$\overline{\mathrm{VP}}$ and $\overline{\mathrm{VNP}}$ are closed under these three types of degenerations (cf. Propositions 6, 8, 11). Since we want to compare $\overline{\mathrm{VP}}$ with VP, and $\overline{\mathrm{VNP}}$ with VNP, we ask how VP and VNP behave under these three degenerations. This is addressed in the following result.

▶ **Theorem 1.**
(a) *Stable-VNP = Newton-VNP = VNP* = VNP, and*
(b) *Stable-VP ⊆ Newton-VP ⊆ VP* ⊆ VNP.*

The statement of this result tells us nothing as to whether any of the inclusions in the sequence Stable-VP ⊆ Newton-VP ⊆ VP* ⊆ $\overline{\mathrm{VP}}$ can be expected to be strict or not. But its proof, as discussed below, does shed light on this subject.

Theorem 1 is proved by combining the Hilbert–Mumford–Kempf criterion for stability [15] with the ideas and results in Valiant [27]. The Hilbert–Mumford–Kempf criterion [15] shows that, for any polynomial f_n in the unique closed G-orbit in the G-orbit-closure of any $g_n \in \mathbb{F}[x_1, \ldots, x_n]$, with $G = \mathrm{SL}_n(\mathbb{F})$, there exists a one-parameter subgroup of G that drives g_n to f_n. Furthermore, by Kempf [15], such a subgroup can be chosen in a canonical manner. As a byproduct of the proof of Theorem 1, we get a complexity-theoretic form of this criterion (cf. Theorem 18), which shows that such a one-parameter group can be chosen

[5] Taking a Newton degeneration and a linear projection need not commute, so the set of Newton degenerations alone will not in general be closed under linear projections. For example, any polynomial f is a linear projection of a sufficiently large determinant, but the Newton degenerations of the determinant only consist of polynomials of the form $\det(X')$ where X' is matrix consisting only of variables and 0s.

[6] It is assumed here that the indices are encoded as the lists of 0-1 variables.

so that the resulting one-parameter degeneration of any $\{g_n\} \in \text{VP}$ to $\{f_n\} \in \text{Stable-VP}$ is p-definable. Thus the inclusion of Stable-VP in VNP ultimately depends on the existence of a very special type of one parameter degeneration of $\{g_n\}$ to $\{f_n\}$, as provided by the Hilbert–Mumford–Kempf criterion, which can be encoded by a small circuit. However, no such degeneration scheme, which can be encoded by a small circuit, is known if f_n is allowed to be any polynomial in the $\text{GL}(n, \mathbb{F})$-orbit-closure of g_n.

If such a scheme exists for every f_n in the $\text{GL}(n, \mathbb{F})$-orbit-closure of g_n, then it would follow that $\overline{\text{VP}} \subseteq \text{VP}^*$, and in conjunction with Theorem 1, that $\overline{\text{VP}} \subseteq \text{VNP}$. This is one plausible approach to show that $\overline{\text{VP}} \subseteq \text{VNP}$, if this is true. If, on the other hand, no such special scheme akin to the Hilbert–Mumford–Kempf criterion for stability exists for every f_n in the $\text{GL}(n, \mathbb{F})$-orbit-closure of g_n, as the extensive research in geometric invariant theory [24] in the last century since the work of Hilbert [12] suggests, then this may be taken as an indication that $\overline{\text{VP}}$ is not contained in VP^*, and hence, also not in VP.

The complexity-theoretic form of the Hilbert-Mumford criterion mentioned above (Theorem 18) also provides an exponential (in n) upper bound on the degree of the canonical Kempf-one-parameter subgroup that drives g_n to f_n, with $\{g_n\} \in \text{VP}$ and $\{f_n\} \in \text{Stable-VP}$. This canonical Kempf-one-parameter subgroup is known to be the fastest way to approach a closed orbit [16]. If one could prove a polynomial upper bound on this degree, then it would follow that Stable-VP = VP (cf. Lemma 17). On the other hand, if a worst-case superpolynomial lower bound on this degree can be proved, then it would be an indication that Stable-VP, and hence $\overline{\text{VP}}$, are different from VP. In other words, this suggests a possible route to formally separate VP and $\overline{\text{VP}}$.

An analogue of Theorem 1 also holds for VP_{ws}, the class of families of polynomials that can be computed by symbolic determinants of polynomial size.

Next we ask if one can construct an *explicit* family in Newton-VP_{ws} that can reasonably be conjectured to be not in VP_{ws} or even VP. With this mind, we first construct an explicit family $\{f_n\}$ of polynomials that can be approximated infinitesimally closely by symbolic determinants of size $\leq n$, but conjecturally cannot be computed exactly by symbolic determinants of $\Omega(n^{2+\delta})$ size, for a small enough positive constant $\delta < 1$; cf. Section 5. This construction follows a suggestion made in [22, Section 4.2]. The family $\{f_n\}$ is a Newton degeneration of the family of perfect matching Pfaffians of graphs. However, this family $\{f_n\}$ turns out to be in VP_{ws}. So we need to extend this idea much further to construct an explicit family in Newton-VP_{ws} that can be conjectured to be not in VP.

To see how, note that perfect matching Pfaffians are derived from a semi-invariant of the symmetric quiver with two vertices and one arrow. This suggests that to upgrade the conjectural $\Omega(n^{2+\delta})$ lower bound to obtain a candidate for a superpolynomial lower bound a possible route is to replace perfect matching Pfaffians by appropriate representation-theoretic invariants. This leads to the second line of investigation, which we now discuss.

1.2 On Newton degeneration of generic semi-invariants

Our next result suggests that these invariants should be non-generic by showing that, for many finite quivers, including some wild ones, Newton degeneration of any generic semi-invariant can be computed by a symbolic determinant of polynomial size.

A quiver $Q = (Q_0, Q_1)$ [6, 8] is a directed graph (allowing multiple edges) with the set of vertices Q_0 and the set of arrows Q_1. A linear representation V of a quiver associates to each vertex $x \in Q_0$ a vector space V^x, and to each arrow $\alpha \in Q_1$ a linear map V^α from $V^{s\alpha}$ to $V^{t\alpha}$, where $s\alpha$ denotes the start (tail) of α and $t\alpha$ its target (head). The dimension vector of V is the tuple of non-negative integers that associates $\dim(V^x)$ to

each vertex $x \in Q_0$. Given a dimension vector $\beta \in \mathbb{N}^{|Q_0|}$, let $\mathrm{Rep}(Q, \beta)$ denote the space of all representations of Q with the dimension vector β. We have the natural action of $\mathrm{SL}(\beta) := \prod_{x \in Q_0} \mathrm{SL}(\beta(x), \mathbb{F})$ on $\mathrm{Rep}(Q, \beta)$ by change of basis. Let $\mathrm{SI}(Q, \beta) = \mathrm{Rep}(Q, \beta)^{\mathrm{SL}(\beta)}$ denote the ring of semi-invariants. The *generic* semi-invariants in this ring (see [6]) will be recalled in Section 6.

We will be specifically interested in the following well-known types of quivers, cf. [7]. The m-Kronecker quiver is the quiver with two vertices, and m arrows between the two vertices with the same direction. It is wild if $m \geq 3$; wildness is a universality property in representation theory, analogous to NP-completeness (see, e.g., [1]). The k-subspace quiver is the quiver with $k+1$ vertices $\{x_1, \ldots, x_k, y\}$ and k arrows $(x_1, y), \ldots, (x_k, y)$. It is wild if $k \geq 5$. The A-D-E Dynkin quivers are the only quivers of finite representation type – they have only finitely many indecomposable representations.

The following result tells us where *not* to look for explicit candidate families in $\overline{\mathrm{VP}} \setminus \mathrm{VP}$.

▶ **Theorem 2.** *Let Q be an m-Kronecker quiver, or a k-subspace quiver, or an A-D-E Dynkin quiver. Then any Newton degeneration of a generic semi-invariant of Q with dimension vector β and degree d can be computed by a weakly skew circuit (or equivalently a symbolic determinant) of* $\mathsf{poly}(|\beta|, d)$ *size, where $|\beta| = \sum_{x \in Q_0} \beta(x)$.*

The proof strategy for Theorem 2 is as follows. Define the coefficient complexity $\mathrm{coeff}(E)$ of a set E of integral linear equalities in $\mathbb{R}^m$ as the sum of the absolute values of the coefficients of the equalities. Define the coefficient complexity of a face of a polytope in $\mathbb{R}^m$ as the minimum of $\mathrm{coeff}(E)$, where E ranges over all integral linear equality sets that define the face, in conjunction with the description of the polytope; cf. Section 6.1.

Theorem 2 is proved by showing that the coefficient complexity of every face of the Newton polytope of a generic semi-invariant of any quiver as above is polynomial in $|\beta|$ and d, though the number of vertices on a face can be exponential.

In view of this result and its proof, to construct an explicit family in $\text{Newton-VP}_{\mathrm{ws}} \setminus \mathrm{VP}_{\mathrm{ws}}$, we should look for appropriate *non-generic* invariants of representations of finitely generated algebras whose Newton polytopes have faces with *superpolynomial coefficient complexity* and *superpolynomial number of vertices*.

Of course, we do not have to confine ourselves to Newton-VP in the search of an explicit candidate family in $\overline{\mathrm{VP}} \setminus \mathrm{VP}$. We may search within VP^*, or even outside VP^*.

Organization. The rest of this article is organized as follows. In Section 2 we cover the preliminaries. In Section 3, we formally define the three degenerations of VP and VNP. In Section 4, we prove Theorem 1. In Section 5 we construct an explicit family $\{f_n\}$ that can be approximated infinitesimally closely by symbolic determinants of size $\leq n$, but conjecturally cannot be computed exactly by symbolic determinants of $\Omega(n^{2+\delta})$ size, for a small enough positive constant $\delta < 1$. In Section 6, we prove Theorem 2 for generalized Kronecker quivers. Due to page constraints, some proofs are deferred to the full version. In particular, there we give additional examples of representation-theoretic symbolic determinants whose Newton degenerations have small circuits. All these examples suggest that explicit families in $\text{Newton-VP}_{\mathrm{ws}} \setminus \mathrm{VP}_{\mathrm{ws}}$ have to be rather delicate.

2 Preliminaries

For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$. We denote by $\mathbf{x} = (x_1, \ldots, x_n)$ a tuple of variables; $\mathbf{x}$ may also denote $\{x_1, \ldots, x_n\}$. Let $\mathbf{e} = (e_1, \ldots, e_n)$ be a tuple of nonnegative integers. We usually

use $\mathbf{e}$ as the exponent vector of a monomial in $\mathbb{F}[x_1, \ldots, x_n]$. Thus, $\mathbf{x}^{\mathbf{e}}$ denotes the monomial with the exponent vector $\mathbf{e}$. Let $|\mathbf{e}| := \sum_{i=1}^{n} e_i$.

For a field $\mathbb{F}$, $\mathrm{char}(\mathbb{F})$ denotes the characteristic of $\mathbb{F}$. Throughout this paper, we assume that $\mathbb{F}$ is algebraically closed. S_n denotes the symmetric group consisting of permutations of n objects.

We say that a polynomial $g = g(x_1, \ldots, x_n)$ is a *linear projection* of $f = f(y_1, \ldots, y_m)$ if g can be obtained from f by letting y_j's be some (possibly non-homogeneous) linear combinations of x_i's with coefficients in the base field $\mathbb{F}$.

A family of polynomials $\{f_n\}_{n\in\mathbb{N}}$ is p-bounded if f_n is a polynomial in $\mathsf{poly}(n)$ variables of $\mathsf{poly}(n)$ degree. The class VP [27] consists of p-bounded polynomial families $\{f_n\}_{n\in\mathbb{N}}$ over $\mathbb{F}$ such that f_n can be computed by an arithmetic circuit over $\mathbb{F}$ of $\mathsf{poly}(n)$ size.

Convention: We call a class $\mathcal{C}$ of families of polynomials *standard* if it contains only p-bounded families, and is closed under linear projections.

By a symbolic determinant of size m over the variables $x_1, \ldots, x_n$, we mean the determinant of an $m \times m$ matrix, whose each entry is a possibly non-homogeneous linear function of $x_1, \ldots, x_n$ with coefficients in the base field $\mathbb{F}$. The class $\mathrm{VP}_{\mathrm{ws}}$ is the class of families of polynomials that can be computed by weakly skew circuits of polynomial size, or equivalently, by symbolic determinants of polynomial size [19].

The class VNP is the class of p-definable families of polynomials [27], that is, those families (f_n) such that f_n has $\mathsf{poly}(n)$ variables and $\mathsf{poly}(n)$ degree, and there exists a family $(g_n(x, y)) \in \mathrm{VP}$ such that $f_n(x) = \sum_{e\in\{0,1\}^{\mathsf{poly}(n)}} g_n(x, e)$.

The class $\overline{\mathrm{VP}}$ is defined as follows [22, 5]. Over $\mathbb{F} = \mathbb{C}$, we say that a polynomial family $\{f_n\}_{n\in\mathbb{N}}$ is in $\overline{\mathrm{VP}}$, if there exists a family of sequences of polynomials $\{f_n^{(i)}\}_{n\in\mathbb{N}}$ in VP, $i = 1, 2, \ldots$, s.t. for every n, the sequence of polynomials $f_n^{(i)}$, $i = 1, 2, \ldots$, goes infinitesimally close to f_n, in the usual complex topology. Here, polynomials are viewed as points in the linear space of polynomials. There is a more general definition that works over arbitrary algebraically closed fields – including in positive characteristic – using the Zariski topology. For a direct treatment, see, e.g. [4, App. 20.6]. The operational version of this definition we use is as follows: $\{f_n(x_1, \ldots, x_m)\} \in \overline{\mathrm{VP}}$ if there exist polynomials $f_{n,t}(x_1, \ldots, x_m) \in \mathrm{VP}_{\mathbb{C}((t))}$ – $f_{n,t}$ is a polynomial in the x_i whose coefficients are Laurent series in t – such that $f_n(x)$ is the coefficient of the term in $f_{n,t}(x)$ of lowest degree in t.

The classes $\overline{\mathrm{VP}_{\mathrm{ws}}}$, $\overline{\mathrm{VNP}}$, and $\overline{\mathcal{C}}$, for any standard class $\mathcal{C}$, are defined similarly.

By the determinantal complexity $\mathrm{dc}(f)$ of a polynomial $f(x_1, \ldots, x_n)$, we mean the smallest integer m s.t. f can be expressed as a symbolic determinant of size m over $x_1, \ldots, x_n$. By the approximative determinantal complexity $\overline{\mathrm{dc}}(f)$, we mean the smallest integer m s.t. f can be approximated infinitesimally closely by symbolic determinants of size m.

Thus the $\mathrm{VP}_{\mathrm{ws}} \neq \mathrm{VNP}$ conjecture in Valiant [27] is equivalent to saying that $\mathrm{dc}(\mathrm{perm}_n)$ is not $\mathsf{poly}(n)$, where perm_n denotes the permanent of an $n \times n$ variable matrix. The $\mathrm{VNP} \not\subseteq \overline{\mathrm{VP}_{\mathrm{ws}}}$ conjecture in [22] is equivalent to saying that $\overline{\mathrm{dc}}(\mathrm{perm}_n)$ is not $\mathsf{poly}(n)$.

A priori, it is not at all obvious that dc and $\overline{\mathrm{dc}}$ are different complexity measures. The following two examples should make this clear.

▶ **Example 3** (Example 9 in [17]). Let $f = x_1^3 + x_2^2 x_3 + x_2 x_4^2$. Then $\mathrm{dc}(f) \geq 5$, but $\overline{\mathrm{dc}}(f) = 3$.

▶ **Example 4** (Proposition 3.5.1 in [18]). Let n be odd. Given an $n \times n$ complex matrix M, let M_{ss} and M_s denote its skew-symmetric and symmetric parts. Since n is odd, $\det(M_{ss})$=0. Hence, for a variable t, $\det(M_{ss} + tM_s) = tf(M) + O(t^2)$, for some polynomial function

$f(M)$. Clearly, $\overline{\mathrm{dc}}(f) = n$, since $\det(M_{ss} + tM_s)/t$ goes infinitisimally close to $f(M)$ when t goes to 0. But $\mathrm{dc}(f) > n$.

The $\mathrm{VP}_{\mathrm{ws}} \neq \overline{\mathrm{VP}_{\mathrm{ws}}}$ conjecture in [21] is equivalent to saying that there exists a polynomial family $\{f_n\}$ such that $\overline{\mathrm{dc}}(f_n) = \mathsf{poly}(n)$, but $\mathrm{dc}(f_n)$ is not $\mathsf{poly}(n)$. Instead of this conjecture, we will focus on the $\mathrm{VP} \neq \overline{\mathrm{VP}}$ conjecture in [21], since the considerations for the former conjecture are entirely similar.

A (convex – we will only consider convex ones here) *polytope* is the convex hull in $\mathbb{R}^n$ of a finite set of points. A *face* of a polytope P is the intersection of P with linear halfspace $H = \{v \in \mathbb{R}^n | \ell(v) \geq c\}$ for some linear function ℓ and constant c such that H contains no points of the (topological) interior of P. Equivalently, a polytope is the intersection of finitely many half-spaces, a half-space $H_{\ell,c} = \{v | \ell(v) \geq c\}$ is tight for P if $P \subseteq H_{\ell,c}$ and $P \nsubseteq H_{\ell,c'}$ for any $c' > c$, and a face of P is the intersection of P with a half-space of the form $H_{-\ell,-c}$ where $H_{\ell,c}$ is tight for P.

3 Degenerations of VP and VNP

To understand the relationship between VP, VNP, and their closures $\overline{\mathrm{VP}}$ and $\overline{\mathrm{VNP}}$, we now introduce three degenerations of VP and VNP. The considerations for $\mathrm{VP}_{\mathrm{ws}}$ and $\overline{\mathrm{VP}_{\mathrm{ws}}}$ are entirely similar.

3.1 Stable degeneration

First we define stable degenerations of VP and VNP.

Consider the natural action of $G = \mathrm{SL}(n, \mathbb{F})$ on $\mathbb{F}[\mathbf{x}] = \mathbb{F}[x_1, \ldots, x_n]$ that maps $f(\mathbf{x})$ to $f(\sigma^{-1}\mathbf{x})$ for any $\sigma \in G$. Following Mumford et al. [24], call $f = f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ *stable* (with respect to the G-action) if the G-orbit of f is Zariski-closed. It is known [24] that the closure of the G-orbit of any $g \in \mathbb{F}[\mathbf{x}]$ contains a unique closed G-orbit. We say that f is a *stable degeneration* of g if f lies in the unique closed G-orbit in the G-orbit-closure of g. (If the G-orbit of g is already closed then this just means that f lies in the G-orbit of g.)

We now define the class Stable-$\mathcal{C}$, the stable degeneration of any standard class $\mathcal{C}$, as follows. We say that $\{f_n\}_{n\in\mathbb{N}}$ is in Stable-$\mathcal{C}$ if (1) $\{f_n\} \in \mathcal{C}$, or (2) there exists $\{g_n\}_{n\in\mathbb{N}}$ in $\mathcal{C}$ such that each f_n is a stable degeneration of g_n with respect to the action of $G = \mathrm{SL}(m_n, \mathbb{F})$, where $m_n = \mathsf{poly}(n)$ denotes the number of variables in f_n and g_n.

▶ **Proposition 5.** *For any class $\mathcal{C}$ of p-bounded families of polynomials, Stable-$\mathcal{C} \subseteq \overline{\mathcal{C}}$. In particular, Stable-VP $\subseteq \overline{VP}$ and Stable-VNP $\subseteq \overline{VNP}$.*

▶ **Proposition 6.** *Stable-$\overline{\mathcal{C}} = \overline{\mathcal{C}}$, in particular Stable-$\overline{VP} = \overline{VP}$, and Stable-$\overline{VNP} = \overline{VNP}$.*

This is a direct consequence of the definitions.

3.2 Newton degeneration

Next we define Newton degenerations of VP and VNP.

Given a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, suppose $f = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{x}^{\mathbf{e}}$. We collect the exponent vectors of f and form the convex hull of these exponent vectors in $\mathbb{R}^n$. The resulting polytope is called the *Newton polytope* of f, denoted $\mathrm{NPT}(f)$. Given an arbitrary face Q of $\mathrm{NPT}(f)$, the Newton degeneration of f to Q, denoted $f|_Q$, is the polynomial $\sum_{\mathbf{e}\in Q} \alpha_{\mathbf{e}} \mathbf{x}^{\mathbf{e}}$.

We now define the class Newton-$\mathcal{C}$, the Newton degeneration of any class $\mathcal{C}$, as follows: $\{f_n\}_{n\in\mathbb{N}}$ is in Newton-$\mathcal{C}$, if there exists $\{g_n\}_{n\in\mathbb{N}}$ in $\mathcal{C}$ such that each f_n is the Newton

degeneration of g_n to some face of $\mathrm{NPT}(g_n)$, or a linear projection of such a Newton degeneration.

▶ **Theorem 7.** *Let $\mathcal{C}$ be any standard class (cf. Section 2). Then Newton-$\mathcal{C} \subseteq \overline{\mathcal{C}}$. In particular, Newton-VP $\subseteq \overline{VP}$ and Newton-VNP $\subseteq \overline{VNP}$.*

Proof. Let $\{f_n\}_{n\in\mathbb{N}}$ be in Newton-$\mathcal{C}$, and suppose $f_n \in \mathbb{F}[x_1, \ldots, x_{m(n)}]$.Then there exists $\{g_n\}_{n\in\mathbb{N}} \in \mathcal{C}$, such that $g_n \in \mathbb{F}[x_1, \ldots, x_m]$, $m = m(n)$, and $f_n = g_n|_Q$, where Q is a face of $\mathrm{NPT}(g_n)$. Suppose the supporting hyperplane of Q is defined by $\langle \mathbf{a}, \mathbf{x}\rangle = b$, where $\mathbf{a} = (a_1, \ldots, a_m)$. If necessary, by replacing $(\mathbf{a}, b)$ with $(-\mathbf{a}, -b)$, we make sure that for an arbitrary exponent vector $\mathbf{e}$ in g_n, $\langle \mathbf{a}, \mathbf{e}\rangle \geq b$. That is, among all exponent vectors, exponent vectors on Q achieve the minimum value b in the direction $\mathbf{a}$.

Now introduce a new variable t, and replace x_i with $t^{a_i}x_i$ to obtain a polynomial $g'_n(x_1, \ldots, x_m, t) = g_n(t^{a_1}x_1, \ldots, t^{a_m}x_m) \in \mathbb{F}[x_1, \ldots, x_m, t]$. By the definition of f_n, $g'_n = t^b \cdot f_n +$ higher order terms in t. Therefore, $\{f_n\} \in \overline{\mathcal{C}}$. ◀

Noting that if $\mathcal{C}$ is closed under linear projections, then so is $\overline{\mathcal{C}}$, we have:

▶ **Corollary 8.** *For any standard class $\mathcal{C}$, Newton-$\overline{\mathcal{C}} = \overline{\mathcal{C}}$. In particular, Newton-$\overline{VP} = \overline{VP}$ and Newton-$\overline{VNP} = \overline{VNP}$.*

3.3 P-definable one-parameter degeneration

Finally, we define p-definable one-parameter degenerations of VP and VNP. We say a family $\{f_n(x_1, \ldots, x_{m_n})\}$, $m_n = \mathsf{poly}(n)$, is a *one-parameter degeneration* of $\{g_n(y_1, \ldots, y_{l_n})\}$, for $l_n = \mathsf{poly}(n)$, of *exponential degree*, if, for some positive integral function $K(n) = O(2^{\mathsf{poly}(n)})$, there exist $c_n(i, j, k) \in \mathbb{F}$, $1 \leq i \leq l_n$, $0 \leq j \leq m_n$, $-K(n) \leq k \leq K(n)$, such that $f_n = \lim_{t\to 0} g_n(t)$, where $g_n(t)$ is obtained from g_n by substitutions of the form

$$y_i = a_0^i + \sum_{j=1}^{m_n} a_j^i x_j, \quad 1 \leq i \leq l_n, \text{ where } a_j^i = \sum_{k=-K(n)}^{K(n)} c_n(i, j, k)t^k, \quad 1 \leq i \leq l_n,\ 0 \leq j \leq m_n.$$

Note that by [3], $\overline{\mathrm{VP}}$ consists exactly of those one-parameter degenerations of VP of exponential degree.

We say that the family $\{f_n(x_1, \ldots, x_{m_n})\}$, $m_n = \mathsf{poly}(n)$, is a *one-parameter degeneration* of $\{g_n(y_1, \ldots, y_{l_n})\}$, $l_n = \mathsf{poly}(n)$, of *polynomial degree* if $K(n)$ above is $O(\mathsf{poly}(n))$ (instead of $O(2^{\mathsf{poly}(n)})$).

We say that a family $\{f_n(x_1, \ldots, x_{m_n})\}$, $m_n = \mathsf{poly}(n)$, is a *p-definable one-parameter degeneration* of $\{g_n(y_1, \ldots, y_{l_n})\}$, $l_n = \mathsf{poly}(n)$, if, for some $K(n) = O(2^{\mathsf{poly}(n)})$, there exists a $\mathsf{poly}(n)$-size circuit family $\{C_n\}$ over $\mathbb{F}$ such that $f_n = \lim_{t\to 0} g_n(t)$, where $g_n(t)$ is obtained from g_n by substitutions of the form

$$y_i = a_0^i + \sum_{j=1}^{m_n} a_j^i x_j, \quad 1 \leq i \leq l_n, \text{ where } a_j^i = \sum_{k=-K(n)}^{K(n)} C_n(i, j, k)t^k, \quad 1 \leq i \leq l_n,\ 0 \leq j \leq m_n.$$

Here it is assumed that the circuit C_n takes as input $\lceil \log_2 l_n \rceil + \lceil \log_2 m_n \rceil + \lceil \log_2(K(n)+1) \rceil$ many 0-1 variables, which are intended to encode three integers (i, j, k) satisfying $1 \leq i \leq l = l_n$, $0 \leq j \leq m = m_n$, and $|k| \leq K(n)$, treating 0 and 1 as elements of $\mathbb{F}$.

Thus a p-definable one-parameter degeneration is a one-parameter degeneration of exponential degree that can be specified by a circuit of polynomial size.

For any class $\mathcal{C}$ we now define $\mathcal{C}^*$, called the p-definable one-parameter degeneration of $\mathcal{C}$, as follows. We say that $\{f_n\} \in \mathcal{C}^*$ if there exists $\{g_n\} \in \mathcal{C}$ such that $\{f_n\}$ is a p-definable one-parameter degeneration of $\{g_n\}$.

▶ **Lemma 9.** *For any standard class $\mathcal{C}$ (cf. Section 2), Newton-$\mathcal{C} \subseteq \mathcal{C}^*$. In particular, Newton-VP $\subseteq$ VP* and Newton-VNP $\subseteq$ VNP*.*

This follows from the proof of Theorem 7, noting that we may always take the coefficients of a face to have size at most $2^{\mathsf{poly}(n)}$. The following are easy consequences of the definitions:

▶ **Proposition 10.** *VP* $\subseteq \overline{\text{VP}}$, and VNP* $\subseteq \overline{\text{VNP}}$.*

▶ **Proposition 11.** *$\overline{\text{VP}}^*$ $= \overline{\text{VP}}$, and $\overline{\text{VNP}}^*$ $= \overline{\text{VNP}}$.*

4 Stable-VNP = Newton-VNP = VNP* = VNP

We now prove Theorem 1, by a circular sequence of inclusions.

Proof of Theorem 1. Since VNP $\subseteq$ Stable-VNP by definition, Theorem 1(a) follows from the facts that Stable-VNP $\subseteq$ Newton-VNP (cf. Theorem 12 below), Newton-VNP $\subseteq$ VNP* (Lemma 9), and VNP* $\subseteq$ VNP (cf. Theorem 15 below).

Theorem 1(b) follows from the facts that Stable-VP $\subseteq$ Newton-VP (cf. Theorem 12 below), Newton-VP $\subseteq$ VP* (Lemma 9), and VP* $\subseteq$ VNP (cf. Corollary 16 below). ◀

▶ **Theorem 12.** *For any class $\mathcal{C}$ of families of p-bounded polynomials, Stable-$\mathcal{C} \subseteq$ Newton-$\mathcal{C}$. In particular, Stable-VP $\subseteq$ Newton-VP and Stable-VNP $\subseteq$ Newton-VNP.*

Proof. Suppose $\{f_n\} \in$ Stable-$\mathcal{C}$. If $\{f_n\} \in \mathcal{C}$ then there is nothing to show. Otherwise, there exists $\{g_n\}_{n\in\mathbb{N}}$ in $\mathcal{C}$ s.t. each f_n is a stable degeneration of g_n with respect to the action of $G = SL_{m_n}(\mathbb{F})$, where m_n denotes the number of variables in f_n and g_n.

It suffices to show that $f = f_n(x_1, \ldots, x_m)$, $m = m_n$, is a Newton degeneration of $g = g_n(x_1, \ldots, x_m)$. Let $\mathbf{x} = (x_1, \ldots, x_m)$.

By the Hilbert–Mumford–Kempf criterion for stability [15], there exists a one-parameter subgroup $\lambda(t) \subseteq G$ such that $\lim_{t\to 0} \lambda(t).g = f$. Let T be the canonical maximal torus in G such that the monomials in x_i's are eigenvectors for the action of T. After a linear change of coordinates (which is allowed since Newton-$\mathcal{C}$ is closed under linear transformations by definition), we can assume that $\lambda(t)$ is contained in T. Thus $\lambda(t) = \text{diag}(t^{k_1}, \ldots, t^{k_m})$ (the diagonal matrix with t^{k_j}'s on the diagonal), $k_j \in \mathbb{Z}$, such that $\sum k_j = 1$.

It follows that f is the Newton degeneration of g to the face of NPT(g) where the linear function $\sum_j k_j x_j$ achieves the minimum value (which has to be zero). ◀

The following result is subsumed by Theorem 15; we include its proof here both for expository clarity (it is somewhat simpler but still gives the flavor) and brevity.

▶ **Theorem 13.** *Newton-VNP $\subseteq$ VNP.*

Proof. Suppose $\{f_n\} \in$ Newton-VNP. If $\{f_n\} \in$ VNP, then there is nothing to show. Otherwise, there exists $\{g_n\}_{n\in\mathbb{N}}$ in VNP such that each f_n is the Newton degeneration of g_n to some face of NPT(g_n), or a linear projection of such a Newton degeneration. Since VNP is closed under linear projections, we can assume, without loss of generality, that f_n is the Newton degeneration of g_n to some face of NPT(g_n).

By Valiant [27], we can assume that $g = g_n(x_1, \ldots, x_m)$, $m = m_n = \mathsf{poly}(n)$, is a projection of $\mathrm{perm}(X)$,[7] where X is a $k \times k$ variable matrix, with $k = \mathsf{poly}(n)$. This means $g = \mathrm{perm}(X')$, where each entry of X' is some variable x_i or a constant from the base field F. Since $f = f_n$ is a Newton degeneration of g, it follows that there is some substitution, as in the proof of Theorem 7, $x_j \to x_j t^{k_j}$, $k_j \in \mathbb{Z}$, such that $f = \lim_{t\to 0} \mathrm{perm}(X'(t))$, where $X'(t)$ denotes the matrix obtained from X' after this substitution.

It is easy to ensure that $|k_j| \leq O(2^{\mathsf{poly}(n)})$. Then, given any permutation $\sigma \in \mathrm{S}_k$, whether the corresponding monomial $\prod_i X'_{i\sigma(i)}$ contributes to f can be decided in $\mathsf{poly}(n)$ time. It follows that the coefficient of a monomial can be computed by an algebraic circuit summed over polynomially many Boolean inputs (convert the implicit $\mathsf{poly}(n)$-time Turing machine into a Boolean circuit, then convert it into an algebraic circuit (as in [27, Remark 1]) that incorporates the constants appearing in the projection). Hence $\{f_n\} \in \mathrm{VNP}$. ◀

Since $\mathrm{VP} \subseteq \mathrm{VNP}$, the preceding result implies:

▶ **Corollary 14.** *Newton-VP $\subseteq$ VNP.*

The following result can proved similarly to Theorem 13; see the full version for its proof.

▶ **Theorem 15.** *$\mathrm{VNP}^* \subseteq$ VNP.*

▶ **Corollary 16.** *$\mathrm{VP}^* \subseteq$ VNP.*

In contrast, using the interpolation technique of Strassen [26] and Bini [2] we have:

▶ **Lemma 17** (cf. also [3], [5, §9.4], [10, Prop. 3.5.4])**.** *If $\{f_n\}$ is a one-parameter degeneration of $\{g_n\} \in$ VP of polynomial degree, then $\{f_n\} \in$ VP.*

A complexity-theoretic form of the Hilbert–Mumford–Kempf criterion. As a byproduct of the proof of Theorem 1, we get the following complexity-theoretic form of the Hilbert–Mumford–Kempf criterion [15] for stability with respect to the action of $G = \mathrm{SL}(m, \mathbb{F})$ on $\mathbb{F}[x_1, \ldots, x_m]$. Given a one-parameter subgroup $\lambda(t) \subseteq G$, we can express it as $A \cdot \mathrm{diag}(t^{k_1}, \ldots, t^{k_m}) \cdot A^{-1}$, for some $A \in G$ and $k_j \in \mathbb{Z}$, $1 \leq j \leq m$. We call $\sum_i |k_i|$ the total degree of $\lambda(t)$. The following theorem is implicit in the proofs of Theorems 12 and 13.

▶ **Theorem 18.** *Suppose $f = f(x_1, \ldots, x_m)$ belongs to the unique closed G-orbit in the G-orbit-closure of $g = g(x_1, \ldots, x_m) \in \mathbb{F}[x_1, \ldots, x_m]$. Then there exists a one-parameter subgroup $\lambda(t) \subseteq G$ such that (1) $\lim_{t\to 0} \lambda(t) \cdot g = f$, and (2) the total degree of λ is $O(exp(m, \langle \deg(g) \rangle))$, where $\langle \deg(g) \rangle$ denotes the bitlength of the degree of g.*

It follows that if $\{f_n\}$ is a stable degeneration of $\{g_n\} \in$ VP, then $\{f_n\}$ is a p-definable one-parameter degeneration of $\{g_n\}$.

See the full version for an analogous result for reductive algebraic groups. We formally propose a question that has ramifications on the Stable-VP vs. VP question (cf. Section 1).

▶ **Question 19.** For some positive constant a, does there exist a stable degeneration $\{f_n\}$ of some $\{g_n\} \in$ VP, with an $\Omega(2^{n^a})$ lower bound on the degree of the canonical Kempf-one-parameter subgroup [15] λ_n driving $\{g_n\}$ to $\{f_n\}$?

[7] To get the proof to work in characteristic 2 as well, simply use the Hamilton cycle polynomial $HC(X) = \sum_{\text{k-cycles } \sigma \in S_k} \prod_{i\in[k]} x_{i,\sigma(i)}$ instead, which is VNP-complete in any characteristic [27].

5 Newton degeneration of perfect matching Pfaffians

In this section, we construct an explicit family $\{f_n\}$ of polynomials such that f_n can be approximated infinitesimally closely by symbolic determinants of size n, but conjecturally requires size $\Omega(n^{2+\delta})$ to be computed by a symbolic determinant, for a small enough positive constant δ. However, the family $\{f_n\}$ turns out to be in $\mathrm{VP}_{\mathrm{ws}}$.

Suppose we have a simple undirected graph $G = (V, E)$ where $V = [n]$. Let $\{x_e \mid e \in E\}$ be a set of variables. The Tutte matrix of G is the $n \times n$ skew-symmetric matrix T_G such that, if $(i, j) = e \in E$, with $i < j$, then $T_G(i, j) = x_e$ and $T_G(j, i) = -x_e$; otherwise $T_G(i, j) = 0$. For a skew-symmetric matrix T, the determinant of T is a perfect square, and the square root of $\det(T)$ is called the Pfaffian of T, denoted $\mathrm{pf}(T)$. We call $\mathrm{pf}(T_G)$ the *perfect matching Pfaffian* of the graph G, and $\mathrm{pf}(T_G) = \sum_P \mathrm{sgn}(P) \prod_{e \in P} x_e$, where the sum is over all perfect matchings P of G, and $\mathrm{sgn}(P)$ takes ± 1 in a suitable manner. It is well-known that $\mathrm{pf}(T_G) \in \mathrm{VP}_{\mathrm{ws}}$.

Note that $\mathrm{NPT}(\mathrm{pf}(T_G))$ is the perfect matching polytope of G, which has the following description by Edmonds. For any $S \subseteq V$, we use $e \sim S$ to denote that e lies at the border of S. When $S = \{i\}$, we may write $e \sim i$ instead of $e \sim \{i\}$.

▶ **Theorem 20** (Edmonds, [9]). *The perfect matching polytope of a graph G is characterized by the following constraints:*

$$
\begin{aligned}
&(a) \quad \forall e \in E, x_e \geq 0; \\
&(b) \quad \forall i \in V, \sum_{e \in E, e \sim i} x_e = 1; \\
&(c) \quad \forall C \subseteq V, |C| > 1 \text{ is odd}, \sum_{e \in E, e \sim C} x_e \geq 1.
\end{aligned} \tag{1}
$$

We shall refer to constraints of type (c) in Equation (1) as "odd-size constraints."

▶ **Theorem 21** (Kaltofen and Koiran, [13, Corollary 1]). *Given $f, g, h \in \mathbb{F}[\mathbf{x}]$, suppose $h = f/g$, and f and g are in VP_{ws}. Then $h \in VP_{ws}$.*

▶ **Theorem 22.** *For any graph G and any face Q of $\mathrm{NPT}(\mathrm{pf}(T_G))$, $\mathrm{pf}(T_G)|_Q \in VP_{ws}$.*

Proof. Thanks to Edmonds' description, any face of $\mathrm{NPT}(\mathrm{pf}(T_G))$ is obtained by setting some of the inequalities in Equation (1) to equalities. As setting $x_e = 0$ amounts to consider some graph G' with e deleted from G, the bottleneck is to deal with the odd-size constraints.

Suppose the face Q is obtained via setting the odd-size constraints corresponding to $C_1, \ldots, C_s$ to equalities, where $C_i \subseteq V$. Note that $s = \mathsf{poly}(n)$, because the dimension of $\mathrm{NPT}(\mathrm{pf}(T_G))$ is polynomially bounded, thus any face can be obtained by setting polynomially many constraints to equalities. Let y be a new variable. For any edge $e \in E$, let the number of $i \in [s]$ s.t. e lies at the border of C_i be k_e. Then transform x_e to $x_e y^{k_e}$. Let the skew-symmetric matrix after the transformation be $\widetilde{T_G}$. Since each perfect matching touches the border of every C_i at least once, y^s divides $\mathrm{pf}(\widetilde{T_G})$, so $f := \frac{\mathrm{pf}(\widetilde{T_G})}{y^s}$ is a polynomial. Furthermore, the y-free terms in f corresponds to those perfect matchings that touch each border exactly once. Thus, setting y to zero in f gives $\mathrm{pf}(T_G)|_Q$.

f is in $\mathrm{VP}_{\mathrm{ws}}$, because $\mathrm{pf}(\widetilde{T_G})$ and y^s are in $\mathrm{VP}_{\mathrm{ws}}$, and use Theorem 21. ◀

Construction of an explicit family. Now we turn to the construction of an explicit family $\{f_n\}$ mentioned in the beginning of this section. We assume that the base field $\mathbb{F} = \mathbb{C}$.

First, we give a randomized procedure for constructing f_n:

1. Fix a small enough constant $a > 0$, and let l be the nearest odd integer to n^a. Fix odd-size disjoint subsets $C_1, \ldots, C_k \subseteq [n]$, $k = \lfloor n^{1-a} \rfloor$, of size l. For example, we can let $C_1 = \{1, \ldots, l\}$, $C_2 = \{l+1, \ldots, 2l+1\}$, etc.
2. Choose a random regular non-bipartite graph G_n on n nodes with degree (say) $\sqrt{n}$.
3. Let Q be the face of $\mathrm{NPT}(\mathrm{pf}(T_G))$ obtained by setting the odd-size constraints corresponding $C_1, \ldots, C_k$ to equalities.
4. Let $f_n = \det(T_G)|_Q$.

Then, f_n can be approximated infinitesimally closely by symbolic determinants of size n; cf. the proof of Theorem 7. By Theorem 22, f_n can be expressed as a symbolic determinant of $\mathsf{poly}(n)$ size. But:

▶ **Conjecture 23.** If $a > 0$ is small enough, then, with a high probability, f_n cannot be expressed as a symbolic determinant of size $\leq n^{2+\delta}$, for a small enough positive constant δ.

This is because, with high probability, the coefficient complexity of Q is $\Omega(n^{1-a+1/2})$, and hence interpolation, which lies at the heart of the algorithm in Theorem 22, can be expected to incur $\Omega(n^{1+\delta})$ blow-up in the determinantal size, for a small enough constant $\delta > 0$. Specifically, if we unwind Strassen's proof of division gate elimination, the number of terms in the interpolation is the degree of the polynomial times the degree of the denomenator. The former number gets increased by a multiplicative factor of the sum of absolute values of variable coefficients in the equations, and the latter number is the sum of absolute values of constant terms. It follows that the coefficient complexity determines the blow-up factor.

To get an explicit family $\{f_n\}$, we let G_n be a pseudo-random graph, instead of a random graph. Some suggestions can be found in the full version.

6 Newton degenerations of generic semi-invariants of quivers

In this section we prove Theorem 2 for the generalized Kronecker quivers. Due to page constraints, proofs for k-subspace quivers and A-D-E Dynkin quivers are in the full version. We assume familiarity with the basic notions of the representation theory of quivers; cf. [6, 8].

6.1 Newton degeneration to faces with small coefficient complexity

We begin by observing that the technique used to prove Theorem 22 can be generalized further. In the proof of Theorem 22, due to Edmonds' description of the perfect matching polytope, every face has a "small" description, by a set of linear equalities whose coefficients are polynomially bounded in magnitude.

For a face Q of a polytope P, we say that a set of linear equalities E characterizes Q with respect to P, if the description of P together with that of E characterizes Q. For E, let $\mathrm{coeff}(E)$ be the sum of the absolute values of the coefficients of the linear equalities in E. We define the *coefficient complexity* of Q as the minimum of $\mathrm{coeff}(E)$ over the linear equality sets E that characterize Q with respect to P. Adapting the proof of Theorem 22 we easily get the following; see the full version for a proof.

▶ **Theorem 24.** *Suppose $f \in \mathbb{F}[x_1, \ldots, x_n]$ can be computed by a (weakly skew) arithmetic circuit of size s. Let Q be a face of $\mathrm{NPT}(f)$ whose coefficient complexity is $\mathsf{poly}(n)$. Then $f|_Q$ can be computed by a (weakly skew) arithmetic circuit of size $\mathsf{poly}(s, n)$.*

▶ Remark. If Q has $\mathsf{poly}(n)$ coefficient complexity, then it can be shown that $f|_Q$ is a one-parameter degeneration of f of $\mathsf{poly}(n)$ degree. Hence, Thm. 24 also follows from Lem. 17.

6.2 Generic semi-invariants of generalized Kronecker quivers

We now discuss Theorem 2 for the m-Kronecker quiver; the proof is deferred to the full version. The m-Kronecker quiver is the graph with two vertices s and t, with m arrows pointing from s to t. When $m \geq 3$, this quiver is wild.

Any tuple of m $n \times n$ matrices is a linear representation of the m-Kronecker quiver of dimension vector (n, n). Let $\mathbb{F}[x_{i,j}^{(k)}]$ denote the ring of polynomials in the variables $x_{i,j}^{(k)}$, where $i, j \in [n]$, and $k \in [m]$. For $k \in [m]$, let $X_k = (x_{i,j}^{(k)})$ denote the variable $n \times n$ matrix, whose (i, j)-th entry is $x_{i,j}^{(k)}$. Let $R(n, m)$ consist of those polynomials in $\mathbb{F}[x_{i,j}^{(k)}]$ that are invariant under the action of every $(A, C) \in \mathrm{SL}(n, \mathbb{F}) \times \mathrm{SL}(n, \mathbb{F})$, which sends $(X_1, \ldots, X_m)$ to $(AX_1C^{-1}, \ldots, AX_mC^{-1})$. $R(n, m)$ is the ring of semi-invariants for the m-Kronecker quiver for dimension vector (n, n) or "*matrix semi-invariants*" due to their similarity with the well-known matrix invariants. The following is proved using Theorem 24:

▶ **Theorem 25.** *The Newton degeneration of a generic semi-invariant of the m-Kronecker quiver with dimension vector (n, n) and degree dn to an arbitrary face can be computed by a weakly skew arithmetic circuit of size* $\mathsf{poly}(d, n)$.

Acknowledgment. We thank the anonymous reviewers for careful reading and suggestions that greatly help to improve the writing of the paper.

References

1. Genrich R. Belitskii and Vladimir V. Sergeichuk. Complexity of matrix problems. *Linear Algebra Appl.*, 361:203–222, 2003. Ninth Conference of the International Linear Algebra Society (Haifa, 2001). doi:10.1016/S0024-3795(02)00391-9.
2. D. Bini. Relations between exact and approximate bilinear algorithms. Applications. *Calcolo*, 17(1):87–97, 1980. doi:10.1007/BF02575865.
3. P. Bürgisser. The complexity of factors of multivariate polynomials. *Found. Comput. Math.*, pages 369–396, 2004.
4. P. Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic Complexity Theory*. A series of comprehensive studies in mathematics. Springer, 1997.
5. P. Bürgisser, J. Landsberg, L. Manivel, and J. Weyman. An overview of mathematical issues arising in the geometric complexity theory approach to $VP \neq VNP$. *SIAM Journal on Computing*, 40(4):1179–1209, 2011.
6. H. Derksen and J. Weyman. Semi-invariants of quivers and saturation for Littlewood-Richardson coefficients. *Journal of the American Mathematical Society*, 13(3):467–479, 2000.
7. H. Derksen and J. Weyman. Quiver representations. *Notices of the American Mathematical Society*, 52(2):200–206, 2005.
8. M. Domokos and A. Zubkov. Semi-invariants of quivers as determinants. *Transformation groups*, 6(1):9–24, 2001.
9. J. Edmonds. Maximum matching and a polyhedron with 0, l-vertices. *J. Res. Nat. Bur. Standards B*, 69(1965):125–130, 1965.
10. J. A. Grochow. *Symmetry and equivalence relations in classical and geometric complexity theory*. PhD thesis, University of Chicago, Chicago, IL, 2012.
11. J. A. Grochow. Unifying known lower bounds via geometric complexity theory. *Computational Complexity*, 24:393–475, 2015. Special issue on IEEE CCC 2014. doi:10.1007/s00037-015-0103-x.
12. D. Hilbert. Über die vollen Invariantensysteme. *Math. Ann.*, 42:313–370, 1893.

13 E. Kaltofen and P. Koiran. Expressing a fraction of two determinants as a determinant. In *Symbolic and Algebraic Computation, International Symposium, ISSAC 2008, Linz/Hagenberg, Austria, July 20-23, 2008, Proceedings*, pages 141–146, 2008. doi:10.1145/1390768.1390790.

14 N. Kayal and R. Saptharishi. A selection of lower bounds for arithmetic circuits. *Progress in Computer Science and Applied Logic*, 26, 2014.

15 G. Kempf. Instability in invariant theory. *Annals of Mathematics*, 108(2):299–316, 1978.

16 F. Kirwan. *Cohomology of Quotients in Symplectic and Algebraic Geometry*. Mathematical Notes 31. Princeton University Press, 1984.

17 J. M. Landsberg. An introduction to geometric complexity theory. *Newsletter Eur. Math. Soc.*, pages 10–18, 2016. Preprint available as arXiv:1509.02503 [math.AG].

18 Joseph M. Landsberg, Laurent Manivel, and Nicolas Ressayre. Hypersurfaces with degenerate duals and the geometric complexity theory program. *Comment. Math. Helv.*, 88(2):469–484, 2013. Preprint available as arXiv:1004.4802 [math.AG]. doi:10.4171/CMH/292.

19 G. Malod and N. Portier. Characterizing Valiant's algebraic complexity classes. *J. Complex.*, 24(1):16–38, 2008.

20 T. Mignon and N. Ressayre. A quadratic bound for the determinant and permanent problem. *International Mathematics Research Notices*, pages 4241–4253, 2004.

21 K. Mulmuley. Geometric complexity theory V: Equivalence between black-box derandomization of polynomial identity testing and derandomization of Noether's Normalization Lemma. In *Revised version under preparation. Preliminary version: FOCS*, 2012.

22 K. Mulmuley and M. Sohoni. Geometric complexity theory I: an approach to the P vs. NP and related problems. *SIAM J. Comput.*, 31(2):496–526, 2001.

23 K. Mulmuley and M. Sohoni. Geometric complexity theory II: towards explicit obstructions for embeddings among class varieties. *SIAM J. Comput.*, 38(3):1175–1206, 2008.

24 D. Mumford, J. Fogarty, and F. Kirwan. *Geometric invariant theory*. Springer-Verlag, 1994.

25 Ran Raz and Amir Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. *Comput. Complexity*, 18(2):171–207, 2009.

26 V. Strassen. Vermeidung von divisionen. *Journal für die reine und angewandte Mathematik*, 264:184–202, 1973.

27 L. G. Valiant. Completeness classes in algebra. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, STOC'79, pages 249–261, New York, NY, USA, 1979. ACM. doi:10.1145/800135.804419.

$AC^0 \circ MOD_2$ Lower Bounds for the Boolean Inner Product*

Mahdi Cheraghchi[†1], Elena Grigorescu[2], Brendan Juba[‡3], Karl Wimmer[§4], and Ning Xie[¶5]

1 **Department of Computing, Imperial College London, London, UK**
m.cheraghchi@imperial.ac.uk

2 **Department of Computer Science, Purdue University, IN, USA**
elena-g@purdue.edu

3 **Department of Computer Science and Engineering, Washington University, St. Louis, MO, USA**
bjuba@wustl.edu

4 **Department of Mathematics and Computer Science, Duquesne University, Pittsburgh, PA, USA**
wimmerk@duq.edu

5 **School of Computing and Information Sciences, Florida International University, Miami, FL, USA**
nxie@cs.fiu.edu

Abstract

$AC^0 \circ MOD_2$ circuits are AC^0 circuits augmented with a layer of parity gates just above the input layer. We study $AC^0 \circ MOD_2$ circuit lower bounds for computing the Boolean Inner Product functions. Recent works by Servedio and Viola (ECCC TR12-144) and Akavia et al. (ITCS 2014) have highlighted this problem as a frontier problem in circuit complexity that arose both as a first step towards solving natural special cases of the matrix rigidity problem and as a candidate for constructing pseudorandom generators of minimal complexity. We give the first superlinear lower bound for the Boolean Inner Product function against $AC^0 \circ MOD_2$ of depth four or greater. Specifically, we prove a superlinear lower bound for circuits of arbitrary constant depth, and an $\tilde{\Omega}(n^2)$ lower bound for the special case of depth-4 $AC^0 \circ MOD_2$. Our proof of the depth-4 lower bound employs a new "moment-matching" inequality for bounded, nonnegative integer-valued random variables that may be of independent interest: we prove an optimal bound on the maximum difference between two discrete distributions' values at 0, given that their first d moments match.

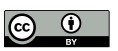
1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases Boolean analysis, circuit complexity, lower bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.35

* A draft of the full version of this paper appears as ECCC Technical Report TR15-030, available online at http://eccc.hpi-web.de/report/2015/030/.

† This work was done in part while Mahdi Cheraghchi was with the Simons Institute for the Theory of Computing at the University of California, Berkeley, supported by a Qualcomm fellowship.
‡ Brendan Juba is supported by an AFOSR Young Investigator Award.
§ Karl Wimmer's research is supported by NSF award CCF-1117079.
¶ Ning Xie's research is supported by NSF award CCF-1423034.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 35; pp. 35:1–35:14

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

1 Introduction

We study lower bounds for computing the inner product function by AC^0 circuits with parity gates on the level just above the input gates ($AC^0 \circ MOD_2$). As we will review, this problem has emerged as a common, particularly simple special case of several major open problems in Computational Complexity, about which we know surprisingly little. We therefore view progress on this special case as a benchmark for new techniques in circuit complexity for these larger questions.

A core program in Computational Complexity is to understand the power of restricted circuit families. One facet of such understanding is to identify functions that these circuits cannot compute. In practice, it turns out that once we can prove such lower bounds, then we become surprisingly facile with the class, gaining the ability to learn the functions computed by such circuits [14] (and this is necessary in some form [10, 11, 29]), the ability to generate inputs that are pseudorandom for the class [16, 18] (again necessary in some form [7, 24, 27]), and more. As a consequence, "understanding" the class is often identified with proving such lower bounds. It is therefore interesting when this intuition *fails* to hold.

Shaltiel and Viola [25] noticed such a gap: although we can prove that, e.g., the MOD_3 function has constant hardness for $AC^0[2]$ circuits [21, 26] (where $AC^0[2]$ is AC^0 equipped with parity gates), we still *do not* have pseudorandom generators for $AC^0[2]$. The trouble is that known constructions of pseudorandom generators require strongly hard on average functions [18], and proofs of hardness amplification require the class in question to compute the majority function, which $AC^0[2]$ cannot even approximate [21]. Shaltiel and Viola therefore highlight the problem of establishing such strong average case hardness against $AC^0[2]$ circuits as a challenge in circuit complexity. Servedio and Viola [23] pointed out that such strong hardness is not even known for $AC^0 \circ MOD_2$, and suggest the problem as a natural special case. In particular, they conjecture that, for this special case, the Inner Product function (IP), defined below, is an example of such a function (although it is trivially computable by $AC^0[2]$).

▶ **Definition 1.** $\mathrm{IP}(x, y)\colon \{0,1\}^{2n} \to \{0,1\}$ is the function $\sum_{i=1}^{n} x_i y_i \pmod 2$.

Thus, showing that IP cannot be computed by small $AC^0 \circ MOD_2$ circuits is a natural step towards a better understanding of $AC^0[2]$.

On the other hand, a better understanding of the class $AC^0 \circ MOD_2$ turns out to be of interest to practical cryptography as well. Along similar lines, Akavia et al. [1], in the course of proposing a candidate weak pseudorandom function of minimal complexity (computable in $AC^0 \circ MOD_2$ in this case), made a strong conjecture; namely that every $AC^0 \circ MOD_2$ circuit has a quasipolynomially heavy Fourier coefficient. Since IP only has small Fourier coefficients, this conjecture also entails the same consequence considered by Servedio and Viola, and simply showing that IP cannot be computed by small $AC^0 \circ MOD_2$ circuits is again a special case of this problem.

Finally, Servedio and Viola [23] note that a special case of Valiant's matrix rigidity problem [28] is to exhibit a function that has low correlation with all sparse polynomials. $AC^0 \circ MOD_2$ circuits are in turn well-approximated by such sparse polynomials, so giving explicit functions that are not correlated with any $AC^0 \circ MOD_2$ functions is again a natural special case; and IP is again the natural candidate for such a function.

Proving lower bounds for $AC^0 \circ MOD_2$ circuits computing IP is challenging since the usual techniques from the literature do not immediately apply. Specifically, although Razborov's technique [21] establishes strong lower bounds against $AC^0[2]$, we note that IP *does* have small $AC^0[2]$ circuits. There is thus no hope in using Razborov's technique directly to prove

lower bounds for IP. And of course, techniques based on random restrictions are helpless against the input layer parity gates.

Servedio and Viola noted that it follows from Jackson's work [8, Fact 8] that depth-3 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuits (i.e., a DNF of parities) cannot approximate IP. Also, Jukna [9] has shown that such circuits computing IP must have exponential size (a bound recently optimized by Cohen and Shinkar [4]). And yet, as Servedio and Viola noted, nothing is known about depth-4 circuits, let alone $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuits of arbitrary depth.

1.1 Our results

In this work, we give the first nontrivial (superlinear) lower bound for IP against (arbitrary depth) $\mathrm{AC}^0 \circ \mathrm{MOD}_2$. In fact, our result is slightly stronger and applies to the broader class of *bent* functions (i.e., functions whose Fourier coefficients are all equal in magnitude, IP being a special case).

▶ **Theorem 2.** *If C is an $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit of depth k and size S that computes the* IP *function on n variables, then $S = \Omega(n^{1+4^{-k}})$.*

The proof of this theorem follows by an adaptation of the results of Chaudhuri and Radhakrishnan [3] who showed a similar bound for AC^0 circuits; a similar adaptation for $\mathrm{AC}^0[2]$ circuits was previously given by Kopparty and Srinivasan [13].

Our main theorem is an $\tilde{\Omega}(n^2)$ $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ lower bound for IP:

▶ **Theorem 3.** *Any depth-4 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit computing the* IP *function on n variables must have size $s = \Omega(n^2/\log^6 n)$.*

An intuitive interpretation of the above results is the following. IP is a means to "generate" all possible parities on n bits. $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuits are merely AC^0 circuits that are given access to an arbitrary but fixed set of parity functions, bounded in number by the size of the circuit. Our results address the question of how much these few parities can aid the computation of most remaining parities.

1.2 Our technique: a moment-matching bound

At the heart of this second lower bound is a lemma that may be of independent interest:

▶ **Lemma 4** (Moment-matching bound)**.** *Let X and Y be random variables taking values in $\{0, 1, 2, \ldots, s\}$. Suppose that the first d moments of X and Y are equal. Then,* $\Pr(Y = 0) \leq \Pr(X = 0) + e^{-\Omega(d/\sqrt{s})}$.

Several other "moment-matching" bounds appear in the literature, and here we briefly discuss the relationship of our work to these bounds. First, the classical *"truncated moments" problem* concerns the conditions for the existence of a probability distribution on a given set with a given sequence of moments [2, 5]. But, as noted by Rashkodnikova et al. [20], the solutions generated by these techniques do not necessarily lie on integers, and so the conditions refer to a different class of random variables. Klivans and Meka [12] likewise considered bounds on the difference in probability of general events that may be induced by distributions with d matching moments. Their bounds apply to much more general properties (than simply the event $X = 0$) and much more general distributions; as such, in spite of some

similarities in the techniques employed in their work[1], they do not obtain bounds in a useful form for our purposes. Rashkodnikova et al. [20] in turn consider nonnegative, bounded, and integer-valued random variables as we do, but they consider a different property; namely, given that the first d moments are *proportional* (not necessarily identical), they maximize their ratio.

Interestingly, it turns out that the moment-matching bound we obtain has a close technical relationship to the *approximate inclusion-exclusion* bounds obtained by Linial and Nisan [15].[2] Indeed, the technique we use to prove Lemma 4 is essentially the same as the core technique underlying Linial and Nisan's work, and in fact, we can show that our moment-matching lemma is essentially equivalent to Linial and Nisan's approximate inclusion-exclusion bounds (details of this equivalence appear in the full version of the paper). In view of the naturalness of the statement of our moment-matching bound, we believe that this lemma may be of interest, even if one is familiar with the approximate inclusion-exclusion bounds.

1.3 Overview of the depth-4 lower bound

Our argument consists of two main steps: (1) We show that any depth-4 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit (without loss of generality, with an AND top gate) of size $s \leq n^2$ computing the Inner Product function must have a one-sided approximation by a DNF of parities in which the terms are all small: It is correct when it outputs 0, and the circuit outputs zero on at least a $1/n^2$ fraction of inputs. (2) We then show that such one-sided approximators for the Inner Product function can only output 0 with small probability, which can be made smaller than $1/n^2$ for some $s = O(n^2/\mathrm{poly}\log n)$.

The first part is relatively straightforward. We let a candidate circuit for the inner product function of size $s \leq n^2$ be given. We first obtain a one-sided approximation to our circuit by invoking the Discriminator Lemma of Hajnal et al. [6] to obtain a depth-3 circuit (eliminating the top AND layer) that is correct whenever it reports 0, and reports 0 on a large ($\geq 1/n^2$) fraction of the inputs. We then reduce the fan-in of the second (from bottom) layer of AND gates by trimming the AND gates with large fan-in at a slight cost in the approximation error (asymptotically smaller than $1/n^2$).

Towards the second part of our argument, we consider the *degree* of an arbitrary parity in the $\{\pm 1\}$-representation in terms of the original variables as well as the bottom layer parities. That is, the degree of a parity χ is now defined as the minimum number of variables and/or bottom layer parities that need to be added together (over $\mathbb{F}_2$) to obtain χ: e.g., a single parity gate (new variable) has degree 1, and a parity of k new variables (parity gates or old variables) has degree $\leq k$. Given the size of the circuit s, we obtain that w.h.p. over the setting of the input y variables, the inner product function $\mathrm{IP}(x, y)$ is a parity in the x variables that remains of high degree (at least $\Omega(n/\log s)$) over these new variables.

We show that, for a $1 - o(1)$ fraction of fixings of the y variables, the probability that our circuit outputs 0 when $\mathrm{IP}(x, y) = 0$ is small as follows. We apply the above-mentioned moment-matching bound (Lemma 12) to the random variable $N(x)$ (over a random x) that counts the number of the AND gates in the depth-3 approximator obtained by the Discriminator Lemma that output 1. We can then show that the first $m = \tilde{\Omega}(n)$ moments of $(N(x) \mid \mathrm{IP}(x, y) = 0)$ and $(N(x) \mid \mathrm{IP}(x, y) = 1)$ are identical and $\Pr_x(N(x) = 0 \mid \mathrm{IP}(x, y) = 1) = 0$ since $N(x) = 0$

[1] Indeed, although like us, Klivans and Meka related this problem to the existence of some polynomials via LP duality, for Klivans and Meka, constructing these (sandwiching) polynomials was the *problem*, not the *solution*.

[2] We are indebted to Johan Håstad for pointing out to us the similarity in the underlying technique.

precisely when the OR gate at the output of the depth-3 one-sided approximator outputs 0, in which case the circuit is correct. Using this information in a linear-programming based proof, we show that $\Pr_x(N(x) = 0 \mid \mathrm{IP}(x, y) = 0) \lesssim e^{-\tilde{\Omega}(m/\sqrt{s})}$. For our m, if $s \leq n^2/\operatorname{poly}\log n$, the upper bound becomes smaller than $1/n^2$, completing the second part and finishing the proof.

To see that the low-degree moments match, we note that $N(x)$ is represented by a low-degree polynomial: In the $\{0,1\}$-representation, it is simply the summation of monomials of degree $O(\log n)$ corresponding to the second-level AND gates (recall that the degree remains the same in the $\{\pm 1\}$-representation). In the $\{\pm 1\}$-representation, however, it is then clear that the parity in x that we obtain from our setting of the y variables in $\mathrm{IP}(x, y)$ is (w.h.p. over y) uncorrelated with $N(x)$. In other words, $\mathbf{E}_x(N(x) \mid \mathrm{IP}(x, y) = 0) = \mathbf{E}_x(N(x) \mid \mathrm{IP}(x, y) = 1)$. This argument can be seen to hold for larger moments as well.

We prove the moment-matching bound by writing a linear program for the probability distribution satisfying the given moment constraints over $\{0, \ldots, s\}$ that maximizes the probability of obtaining 0. We bound the value of this LP by giving an explicit dual-feasible solution; It turns out that the dual can be rewritten as maximizing the lower bound on the values a bounded degree polynomial attains at the integer points in $[0, s]$, given that it takes value 0 at the origin and is also upper bounded by 1 at these integer points. This is quite similar to the conditions for approximators for the OR function sought by Nisan and Szegedy [17], and our solution follows theirs, using Chebyshev polynomials to construct the desired (essentially optimal, cf. Paturi [19]) polynomial.

Alternative proof. There is also an alternative way of completing the proof of our lower bound that uses approximate inclusion-exclusion directly. We believe that the main proof we describe here, using moment-matching, is more natural. This alternative proof appears in the full version of the paper.

1.4 Preliminaries and notation

All logarithms in this paper are to the base 2. Let $n \geq 1$ be a natural number. We use $[n]$ to denote the set $\{1, \ldots, n\}$. We use $\mathbb{F}_2$ for the field with 2 elements $\{0, 1\}$, where addition and multiplication are performed modulo 2. We view elements in $\mathbb{F}_2^n$ as n-bit binary strings – that is elements of $\{0,1\}^n$ – alternatively. If x and y are two n-bit strings, then $x + y$ (or $x - y$) denotes bitwise addition (i.e. XOR) of x and y. We view $\mathbb{F}_2^n$ as a vector space equipped with an inner product $\langle x, y\rangle$, which we take to be the standard dot product: $\langle x, y\rangle = \sum_{i=1}^n x_i y_i$, where all operations are performed in $\mathbb{F}_2$.

Often times, it is convenient to switch the range of Boolean functions between $\{0, 1\}$ and $\{-1, 1\}$. We use $f^{\pm}$ to denote the $\{-1, 1\}$-valued Boolean function corresponding to f. They are related by $f^{\pm} = (-1)^f = 1 - 2f$ and $f = (1 - f^{\pm})/2$.

A *linear threshold* gate $T_k^{\mathbf{a}}(x_1, \ldots, x_t)$ of fan-in t outputs 1 if and only if $\sum_{i=1}^t a_i x_i \geq k$, where $\mathbf{a} = (a_1, \ldots, a_t)$ is vector of weights. The Discriminator Lemma of Hajnal et al. is a powerful tool for proving lower bounds of threshold circuits.

▶ **Lemma 5** (Discriminator lemma, Lemma 3.3 in [6]). *Let $C = T_k^{\mathbf{a}}(C_1, \ldots, C_m)$ be a circuit on n inputs with a threshold gate at the top level, and write $a = \sum_{i=1}^m |a_i|$. Let $A, B \subseteq \{0,1\}^n$ be any two disjoint sets of inputs such that the circuit C accepts A and rejects B. Then there exists a subcircuit C_i, $i \in [m]$, such that $|\Pr_A(C_i(x) = 1) - \Pr_B(C_i(x) = 1)| \geq 1/a$, where $\Pr_A(C_i(x))$ (resp., $\Pr_B(C_i(x))$) denotes the uniform probability over the set A (resp., B).*

2 Lower bound for depth-4 circuit

In this section we will show an $\tilde{\Omega}(n^2)$ lower bound for any depth-4 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit that computes $\mathrm{IP}(x, y)$. Note that all circuits here are allowed to have negations below the XOR gates; these negations are not counted in the depth of the circuit.

2.1 Depth-3 discriminator

Let C be any depth-4 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit that computes $\mathrm{IP}(x, y)$. First, without loss of generality, we may assume the top layer gate of C is an AND gate; the case that top layer gate is an OR gate follows a similar argument[3]. Second, suppose $C = \mathrm{AND}(C_1, \ldots, C_m)$, where each subcircuit C_i is a parity-DNF circuit; then because $C(x, y) = \mathrm{IP}(x, y)$ for *every* input, each subcircuit C_i must compute IP with *one-sided* error only. Specifically, for every input (x, y) with $\mathrm{IP}(x, y) = 1$ and every i, $C_i(x, y) = 1$.

We invoke a consequence of the Discriminator Lemma of Hajnal et al. [6].

▶ **Claim 6** (Consequence of Lemma 5). *There is a subcircuit C_i, $i \in [m]$, such that $\Pr_{(x,y):\, \mathrm{IP}(x,y)=1}(C_i(x, y) = 1) = 1$, and $\Pr_{(x,y):\, \mathrm{IP}(x,y)=0}(C_i(x, y) = 1) \leq 1 - 1/m$.*

We call such a depth-3 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit C_i a *one-sided $1/m$-discriminator* for IP. Our main lemma is an upper bound on the discriminator parameter $1/m$ of such discriminators in terms of its size.

▶ **Lemma 7** (Main). *Suppose that a depth-3 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit of size s is a one-sided ϵ-discriminator for IP. Then ϵ satisfies*

$$\epsilon \leq 4 \exp\left(-\sqrt{\frac{n^2}{128 s \log^2 n \log^2 s}}\right) + \frac{4s}{n^4} + 2^{-n/2}.$$

The proof of Lemma 7 is discussed in Section 2.3. Assuming Lemma 7, the proof of Theorem 3 is straightforward. If $m \geq n^2$, then we are done already. Suppose otherwise, so $\epsilon \geq 1/n^2$. Then by Lemma 7, the size of discriminator subcircuit C_i is of size at least $s = \Omega(\frac{n^2}{\log^6 n}) = \tilde{\Omega}(n^2)$.

2.2 Random y-restrictions

Let C' be a size-s depth-3 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit which is a one-sided ϵ-discriminator for IP. So $\Pr_{(x,y):\, \mathrm{IP}(x,y)=0}(C'(x, y) = 0) \geq \epsilon$, and $C' = \mathrm{OR}(f_1, f_2, \ldots, f_{s'})$, where each f_i is an AND of parities and $s' < s$. Without loss of generality, we can assume that none of these AND gates are constant (i.e., always 0 or 1).

Reducing the fan-in of AND gates

Define the *codimension* of f_i (each of which is an AND of parities) to be the codimension of the subspace corresponding to the coset of inputs on which f_i evaluates to 1.

For example, if $f_1 = \mathrm{AND}(x_1 + x_2, x_1 + x_3, \neg(x_2 + x_3))$, then $x_1 + x_2$ and $x_1 + x_3$ both evaluating to 1 necessarily implies that $\neg(x_2 + x_3)$ evaluates to 1. Hence, the set of inputs for

[3] One way to see this is to notice that our proof also shows the same lower bound for the negation of the Inner Product function (since negating only incurs an affine shift that our methods are not sensitive to). Thus it suffices to note that when the top gate is an OR one can just negate the layers and get a circuit in which the top gate is AND that computes the negation of the Inner Product.

which $f_1(x) = 1$ is the affine subspace specified by $\{x_1 + x_2 = 1 \wedge x_2 + x_3 = 1\}$; consequently, the codimension of f_1 is 2.

The codimension of f_i measures the "effective" fan-in of the AND gate in C'. It is straightforward that without loss of generality one can assume the co-dimension of each AND gate to be equal to its fan-in (one can simply eliminate linearly dependent inputs to each AND gate). From now on, we assume that all redundant parity inputs have already been removed and each AND gate in C' has its fan-in equal to its codimension. Our next step is trim those AND gates of C' whose fan-in is large.

Call an AND gate in C' "bad" if its fan-in is larger than $4 \log n$. We reduce C' to a circuit C'' by trimming all "bad" AND gates to an arbitrary set of $4 \log n$ inputs in their fan-in. Note that each trimmed AND gate may cause an error, only from 0 to 1, and only when all its (non-trimmed) inputs evaluate to 1 (an event that happens with probability at most $2^{-4 \log n}$, since the inputs of each gate are uniform and independent). Define $\tau = \Pr_{x,y}(C'(x, y) \neq C''(x, y))$. By the union bound, $\tau \leq s2^{-4 \log n} = s/n^4$. Further, if $C'(x, y) \neq C''(x, y)$ then we must have $C'(x, y) = 0$ and $C''(x, y) = 1$. In other words, if $\epsilon' := \Pr_{(x,y):\, \mathrm{IP}(x,y)=0}(C''(x, y) = 0)$, then $\epsilon' \geq \epsilon - \tau$, and moreover, if C' approximates IP with a one-sided error (i.e., $C' = 1$ whenever IP $= 1$), then so does C''.

▶ **Definition 8.** For a function $F(x, y)$ (resp., a circuit $C(x, y)$) that maps $\{0, 1\}^n \times \{0, 1\}^n$ to $\{0, 1\}$, a *y-restriction* $\rho \in \{0, 1\}^n$ is an assignment of all the y variables in the input according to ρ. Denote the resulting function F (resp., circuit C) after applying restriction ρ by $F|_\rho$ (resp., $C|_\rho$).

A simple fact exploited in the proof is that, for any y-restriction ρ, $\mathrm{IP}|_\rho$ is a parity over the x variables, which we denote by ℓ_ρ. Note that $\ell_\rho(x) = \sum_{i:\rho_i=1} x_i \bmod 2$. We next argue that for *any* fixed depth-3 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit C'', the parity function ℓ_ρ resulting from a random y-restriction ρ is of "high degree" with respect to the parity inputs of C'', and thus "hard" for the circuit.

Fix an arbitrary depth-3 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit C'', of which the fan-in of each AND gates is at most $4 \log n$. Let the parity inputs for C'' be $\ell_{(a_1, S_1^x, S_1^y)}, \ldots, \ell_{(a_{s'}, S_{s'}^x, S_{s'}^y)}$, where $a_i \in \{0, 1\}$, $S_i^x, S_i^y \subseteq [n]$, $\ell_{S_i^x, S_i^y}(x, y) = a_i + \sum_{j \in S_i^x} x_j + \sum_{j \in S_i^y} y_j$, and $s' < s$.

Observe that after applying a y-restriction ρ to C'', the inputs to $C''|_\rho$ become the x-part of the original parities or their negations, namely $\ell_{(a_i, S_i^x, S_i^y)}|_\rho = a_i' + \sum_{j \in S_i^x} x_j$ and $a_i' = a_i$ or $a_i' = 1 - a_i$. Since there is a natural one-to-one correspondence between subsets of $[n]$ and vectors in $\mathbb{F}_2^n$, we may use a set of vectors $S \subseteq \mathbb{F}_2^n$ to identify the set of parities (or their negations), namely $\{S_i^x\}_{i \in [s']}$, that are fed into $C''|_\rho$. A key point is that the subset S depends only on the circuit C'', and is essentially independent of the choice of y-restriction ρ. Note also that $|S| \leq s' < s$. In the following, we will slightly abuse notation and use a parity and the subset of $[n]$ corresponding to that parity interchangeably.

IP results in high degree parity under random restriction

Following standard additive combinatorial notation, for a subset $S \subseteq \mathbb{F}_2^n$ and a positive integer k, let $kS = \{x_1 + \cdots + x_k \colon x_1, \ldots, x_k \in S\}$. Clearly we have $|S \cup 2S \cup \cdots \cup kS| \leq (|S| + 1)^k$.

▶ **Definition 9.** For any $S \subseteq \mathbb{F}_2^n$ and $z \in \mathbb{F}_2^n$, the *S-degree* of z is the smallest integer d such that $z \in dS$, or ∞ if no such d exists. Further, the S-degree of a parity function $\langle \alpha, x \rangle$ for $\alpha \in \mathbb{F}_2^n$ is the S-degree of α.

Our next claim shows that for any fixed size-s depth-3 $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit C'', after applying a random y-restriction, then almost surely, the resulting parity function ℓ_ρ is of

high degree with respect to the parity inputs of $C''|_\rho$.

▶ **Claim 10.** *Let $S \subseteq \mathbb{F}_2^n$ be the set of input parities (or their negations) of $C''|_\rho$. Then with probability at least $1 - 2^{-n/2}$ over the choice of ρ, ℓ_ρ has S-degree larger than $n/(2\log s)$.*

Proof. Set $k = n/(2\log s)$. We have $|S \cup 2S \cup \cdots \cup kS| \leq (|S|+1)^k \leq s^k = s^{n/(2\log s)} = 2^{n/2}$, so the probability that the S-degree of ℓ_ρ being at most k is no more than $2^{n/2}/2^n = 2^{-n/2}$. ◀

We will call a y-restriction ρ *good* (for circuit C'') if the S-degree of ℓ_ρ is larger than $n/(2\log s)$ and *bad* otherwise. Therefore a random ρ is bad with probability at most $2^{-n/2}$. Let $N_\rho : \{0,1\}^n \to \mathbb{N}$ be the function that counts the number of AND gates of $C''|_\rho$ that are 1.

▶ **Lemma 11.** *Let $S \subseteq \mathbb{F}_2^n$ be the set of input parities (or their negations) of $C''|_\rho$. Suppose ℓ_ρ has S-degree larger than k and each* AND *gate in $C''|_\rho$ has fan-in at most w, then N_ρ^i is uncorrelated with ℓ_ρ for $i = 1, 2, \ldots, k/w$. In other words, $\mathbf{E}_x(N_\rho^i(x) \mid \ell_\rho(x) = 0) = \mathbf{E}_x(N_\rho^i(x) \mid \ell_\rho(x) = 1)$ for $i = 1, 2, \ldots, k/w$.*

Proof. For convenience, we switch to the $\{-1, 1\}$ representation of Boolean values for parities, i.e. $\chi(x) = (-1)^{\ell(x)}$. Let $\chi_1, \ldots, \chi_{s'}$ be the input parities of $C''|_\rho$, and let $f_1', f_2', \ldots, f_{t'}'$ (each still taking value in $\{0,1\}$) be the functions computed by the AND gates in $C''|_\rho$. Then $N_\rho(x) = f_1'(x) + f_2'(x) + \cdots + f_{t'}'(x)$. Note that since each $f_j'(x)$ is the AND of at most w parities from $\{\chi_1, \ldots, \chi_{s'}\}$, $f_j'(x)$ can be expressed as a polynomial of degree at most w with $\chi_1, \ldots, \chi_{s'}$ as variables (indeed, if $f_1'(x) = \mathrm{AND}(\chi_1(x), \ldots, \chi_w(x))$, then $f_1' = (\frac{1-\chi_1}{2}) \cdots (\frac{1-\chi_w}{2})$). Consequently, N_ρ^i is a polynomial of degree at most $i \cdot w$ in $\chi_1, \ldots, \chi_{s'}$. Now because ℓ_ρ is of S-degree larger than $k \geq i \cdot w$ for $i = 1, 2, \ldots, k/w$, we have that ℓ_ρ is not in the support of the polynomial representation of N_ρ^i. Finally, by the orthogonality of parities, letting $\chi_\rho(x) := (-1)^{\ell_\rho(x)}$, we have

$$0 = \langle N_\rho^i, \ell_\rho \rangle = \mathbf{E}_x(N_\rho^i(x) \cdot \ell_\rho(x)) = \frac{1}{2}\left(\mathbf{E}_x(N_\rho^i(x) \mid \chi_\rho(x) = 0) - \mathbf{E}_x(N_\rho^i(x) \mid \chi_\rho(x) = 1)\right). \quad \blacktriangleleft$$

Since each of the AND gates in C'' has fan-in at most $4\log n$ and the S-degree of ℓ_ρ is larger than $n/(2\log s)$ for every good ρ, Lemma 11 implies that N_ρ^i is uncorrelated with ℓ_ρ for i up to $d := n/(8\log n \log s)$ for every good y-restriction.

2.3 Linear programming and feasible solutions based on Chebyshev polynomials (Proof of Lemma 7)

In this section we prove Lemma 7. Let X_ρ (resp. Y_ρ) be the (conditional) random variable of $N_\rho(x) \mid (\ell_\rho(x) = 1)$ (resp., $N_\rho(x) \mid (\ell_\rho(x) = 0)$). Our key observation is that, by Lemma 11, these two random variables both take values in $\{0, 1, \ldots, s'\}$ and their moments match up to $n/8\log n \log s$. So intuitively, if s' is not too large, these two random variables should have close to identical distributions; in particular, we should have $\Pr(X_\rho = 0) \approx \Pr(Y_\rho = 0)$. Since C' (and thus, C'') computes IP with only one-sided error, we have that for *every* y-restriction ρ, $\Pr(C''|_\rho(x) = 1 \mid \ell_\rho(x) = 1) = 1$ and consequently $\Pr(X_\rho = 0) = 0$. Combining this with the consequence of moment-matching condition between X_ρ and Y_ρ implies that $\Pr(Y_\rho = 0) \approx 0$ for every good ρ.

Fix a good y-restriction ρ. The following key lemma provides the desired upper bound on $\Pr(Y_\rho = 0) = \Pr_x(C''|_\rho(x) = 0 \mid \ell_\rho(x) = 0)$. The lemma allows an additional parameter ξ_ρ which in our application is set to zero (since we have $\Pr_x(C''|_\rho(x) = 0 \mid \ell_\rho(x) = 1) = 0$). However, since the lemma applies to general random variables with matching moments and may be of independent interest, it is stated in the more general form.

$$
\begin{array}{lll}
\text{Max} & y_0 & \text{(primal LP)} \\
\text{s.t.} & \sum_{i=0}^{s'} i^j x_i - \sum_{i=0}^{s'} i^j y_i = 0, \quad j = 1, \ldots, d & \\
& \sum_{i=0}^{s'} x_i = 1 & \\
& \sum_{i=0}^{s'} y_i = 1 & \\
& x_0 = \xi_\rho & \\
& x_i, y_i \geq 0, \quad i = 0, \ldots, s' &
\end{array}
$$

$$
\begin{array}{lll}
\text{Min} & 1 - (1 - \xi_\rho)z & \text{(dual LP)} \\
\text{s.t.} & p \text{ is a polynomial of degree at most } d & \\
& p(0) = 0 & \\
& z \leq p(i) \leq 1 \quad i = 1, \ldots, s' &
\end{array}
$$

Figure 1 Primal LP for finding maximum $\Pr(Y = 0)$ (top), and the final dual LP for finding maximum $\Pr(Y = 0)$ (bottom).

▶ **Lemma 12.** *Let X_ρ and Y_ρ be random variables supported on $\{0, 1, \ldots, s'\}$ such that*
(i) $\mathbf{E}(X_\rho^i) = \mathbf{E}(Y_\rho^i)$ *for $i = 1, \ldots, d$; and*
(ii) $\Pr(X_\rho = 0) = \xi_\rho$.
Then $\Pr(Y_\rho = 0) \leq \xi_\rho + 4(1 - \xi_\rho)e^{-d/\sqrt{2s'}}$.

Proof. We set up a linear program to maximize $\Pr(Y_\rho = 0)$ over the choices of random variables X_ρ and Y_ρ. The variables in the LP are x_i and y_i where $x_i = \Pr(X_\rho = i)$ and $y_i = \Pr(Y_\rho = i)$. Aside from nonnegativity and an upper bound constraint for x_0, we have $d+2$ equality constraints; 2 of them to force X_ρ and Y_ρ to have probability distributions, and the other d for the moment matching condition. The linear program and the corresponding dual are listed in Figure 1. In order to upper bound the value of the primal program (i.e., $\Pr(Y_\rho = 0)$) and prove Lemma 12, it suffices to find a feasible solution to the corresponding dual program. We show that by choosing the polynomial p in the dual to be a Chebyshev polynomial (appropriately shifted and scaled), an essentially optimal bound on the primal value can be found.

Denote by P_ρ the value of the primal LP in Figure 1. The dual linear program is

$$
\begin{array}{ll}
\text{minimize} & z_{d+1} + z_{d+2} + \xi_\rho z_{d+3} \\
\text{such that} & z_{d+1} + z_{d+3} \geq 0 \\
& z_{d+2} \geq 1 \\
& (\sum_{j=1}^{d} i^j z_j) + z_{d+1} \geq 0 \qquad i = 1, \ldots, s' \\
& (\sum_{j=1}^{d} -i^j z_j) + z_{d+2} \geq 0 \qquad i = 1, \ldots, s'
\end{array}
$$

We can interpret the dual as a problem involving polynomials. The feasible solutions correspond to coefficients of degree-d polynomials $p(x) = \sum_{j=1}^{d} z_j x^j$ with $p(0) = 0$. By duality, the objective value of the dual is nonnegative for any feasible solution. Thus, by scaling, we can assume $z_{d+2} = 1$. Further, since z_{d+3} only appears in the first constraint in this minimization problem, we can always take $z_{d+3} = -z_{d+1}$.

Rearranging the last two constraints of this problem yields that $p(1), p(2), \ldots, p(s')$ must all lie in the interval $[-z_{d+1}, z_{d+2}]$. Setting $z = -z_{d+1}$, the dual problem can be rephrased

as the final Dual LP showed in Figure 1.

Denote by D_ρ the value of this dual LP. By the Strong Duality Theorem, $P_\rho = D_\rho$, and therefore if $V(p)$ is the value of any feasible solution corresponding to a polynomial p to the dual LP, we have $\Pr(Y_\rho = 0) \le P_\rho = D_\rho \le V(p)$.

The above modified problem about polynomials is strikingly similar to the problem of approximating OR functions by low-degree polynomials, for which Nisan and Szegedy gave an optimal solution based on Chebyshev polynomials [17]. Recall that Chebyshev polynomial (of the first kind) $T_k(x)$ is a degree k polynomial defined by $T_k(x) = \cos(k \arccos(x))$, or more explicitly

$$T_k(x) = \frac{1}{2}\left[\left(x + \sqrt{x^2-1}\right)^k + \left(x - \sqrt{x^2-1}\right)^k\right].$$

It is well-known that $-1 \le T_k(x) \le 1$ for all $x \in [-1,1]$ and $T_k(x) > 1$ when $x > 1$. For a detailed treatment of Chebyshev polynomials see e.g. [22].

We now construct a dual feasible polynomial p based on Chebyshev polynomials. Define

$$q(x) := 1 - \frac{T_d(\frac{s'-x}{s'-1})}{T_d(\frac{s'}{s'-1})}, \qquad p(x) := \frac{q(x)}{\max_{i\in\{1,\ldots,s'\}} q(i)}.$$

Clearly $p(x)$ is a degree d polynomial, $p(0) = 0$ and $p(i) \le 1$ for $i = 1, \ldots, s'$, hence a feasible solution to the dual LP.

▶ **Claim 13.** *The value of $p(x)$ with respect to the dual LP satisfies that $D(p) \le \xi_\rho + \frac{2(1-\xi_\rho)}{T_d(1+\frac{1}{s'})}$.*

Proof. Since $-1 \le T_d(w) \le 1$ for all $-1 \le w \le 1$, then for $i = 1, \ldots, s'$,

$$\begin{aligned} p(i) &= \frac{q(x)}{\max_{i\in\{1,\ldots,s'\}} q(i)} = \frac{T_d(1+\frac{1}{s'-1}) - T_d(\frac{s'-i}{s'-1})}{T_d(1+\frac{1}{s'-1}) - \min_{j\in[s']} T_d(\frac{s'-j}{s'-1})} \\ &\ge \frac{T_d(1+\frac{1}{s'-1}) - 1}{T_d(1+\frac{1}{s'-1}) + 1} = \frac{1 - 1/T_d(1+\frac{1}{s'-1})}{1 + 1/T_d(1+\frac{1}{s'-1})} \ge 1 - \frac{2}{T_d(1+\frac{1}{s'-1})} \ge 1 - \frac{2}{T_d(1+\frac{1}{s'})}. \end{aligned}$$

Therefore the value z in the objective function of dual LP is at least $z \ge 1 - \frac{2}{T_d(1+\frac{1}{s'})}$ and the claim follows. ◀

We will need the following two inequalities bounding $T_k(x)$'s growth when $x \ge 1$.

▶ **Claim 14.** *For any nonnegative integer k, we have*[4]

1. *$T_k(1+\mu) \ge \frac{1}{2} e^{(\sqrt{2\mu+\mu^2})k/2}$ for all real number $0 \le \mu \le 1$.*
2. *$T_k(1+\mu) \le e^{2(\sqrt{2\mu+\mu^2})k}$ for all $\mu \ge 0$.*

Proof. For the first part, using that $1 + x \ge e^{x/2}$ for $0 \le x \le 2$, we obtain

$$T_k(1+\mu) \ge \frac{1}{2}(1 + \mu + \sqrt{2\mu+\mu^2})^k \ge \frac{1}{2}(1 + \sqrt{2\mu+\mu^2})^k \ge \frac{1}{2} e^{(\sqrt{2\mu+\mu^2})k/2},$$

for all $0 \le \mu \le 1$.

For the second part, by the standard inequality $(1 + t/n)^n \le e^t$ for all nonnegative t and n,

$$T_k(1+\mu) \le (1 + \mu + \sqrt{2\mu+\mu^2})^k \le (1 + 2\sqrt{2\mu+\mu^2})^k \le e^{2(\sqrt{2\mu+\mu^2})k}.$$ ◀

[4] The second inequality also appeared in [19].

Finally, by setting $\mu = 1/s'$ in the first inequality of Claim 14, we have $T_d(1+1/s') \geq \frac{1}{2}e^{(\sqrt{2/s'+1/s'^2})d/2} \geq \frac{1}{2}e^{\sqrt{d^2/2s'}}$. Combining this with Claim 13, we get $\Pr(Y_\rho = 0) \leq \xi_\rho + 4(1-\xi_\rho)e^{-d/\sqrt{2s'}}$, which completes the proof of Lemma 12. ◀

In order to complete the proof of Lemma 7, recall that X_ρ (resp. Y_ρ) is the (conditional) random variable of $N_\rho(x) \mid (\ell_\rho(x) = 1)$ (resp., $N_\rho(x) \mid (\ell_\rho(x) = 0)$), where $N_\rho(x)$ counts the number of AND gates in $C''|_\rho$ that evaluate to 1. Since $C''|_\rho$ provides a one-sided approximation of the function ℓ_ρ, we have that $\xi_\rho := \Pr_x(N_\rho(x) = 0 \mid \ell_\rho(x) = 1) = 0$.

Taking $d = \frac{n}{8 \log n \log s}$ and $s' < s$ into Lemma 12, we have that for any good y-restriction ρ,

$$\begin{aligned}\Pr_x(C''|_\rho(x) = 0 \mid \ell_\rho(x) = 0) = \Pr(Y_\rho = 0) &\leq \xi_\rho + 4(1-\xi_\rho)\exp(-d/2\sqrt{s}) \\ &= 4\exp(-d/2\sqrt{s}). \end{aligned} \tag{1}$$

Taking into account bad ρ's, which happens with probability at most $2^{-n/2}$ (according to Claim 10), the discriminator parameter ϵ' for $C''(x,y)$ can now be upper bounded as

$$\begin{aligned}\epsilon' &= \Pr_{x,\rho}(C''|_\rho(x) = 0 \mid \ell_\rho(x) = 0) = \mathbf{E}_\rho(\Pr_x(C''|_\rho(x) = 0 \mid \ell_\rho(x) = 0)) = \mathbf{E}_\rho(Y_\rho) \\ &= \mathbf{E}_{\text{good }\rho}(Y_\rho)\Pr(\rho \text{ is good}) + \mathbf{E}_{\text{bad }\rho}(Y_\rho)\Pr(\rho \text{ is bad}) \leq 4\exp(-d/2\sqrt{s}) + 2^{-n/2}.\end{aligned}$$

Finally, since $\epsilon' \geq \epsilon - \tau$, where we recall that $\tau = \Pr_{x,y}(C'(x,y) \neq C''(x,y)) \leq s/n^4$, the proof of Lemma 7 is complete.

2.4 Limitations of our approach

We remark that the $\tilde{\Omega}(n^2)$ lower bound is optimal (up to a polylogarithmic factor) for our current approach. This follows from a theorem of Paturi [19], which states that if $p(x)$ is a degree d polynomial such that $0 \leq p(i) \leq 1$ for $i = 0, 1, \ldots, s$ and $|p(1) - p(0)| \geq c$ for some constant c, then $d = \Omega(\sqrt{s})$, or equivalently $s = O(d^2)$. Since in our setting $d = \Theta(n/\log n \log s)$, the best lower bound one can show in the current framework is $\tilde{O}(n^2)$.

3 Superlinear Lower Bound for General Circuits

In this section we prove the following superlinear lower bound for $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuits of arbitrary depth. Throughout this section we find it more convenient to use $(x_1, \ldots, x_n)$ as the entire input to IP rather than the two-input notation $(x_1, \ldots, x_n, y_1, \ldots, y_n)$ used previously. We remark that the results of this section hold for a more general class of functions than IP, namely *bent functions*[5]. We state the results here for IP, and prove them for bent functions in the appendix.

▶ **Theorem 15.** *If C is an $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit of depth k and size S that computes $\mathrm{IP}\colon \{0,1\}^n \to \{0,1\}$, then $S = \Omega(n^{1+4^{-k}})$.*

[5] A Boolean function is bent if all its Fourier coefficients are equal in magnitude. The Inner Product function (IP) is a special case. In fact, our result holds for any function whose Fourier coefficients are all exponentially small in magnitude.

Deterministic restrictions

The high level idea of the proof is to adapt the technique of "deterministic restrictions" [3] to $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuits. In contrast to *random restrictions* which simplify circuits probabilistically, deterministic restrictions aim to show that, if the circuit size is small, then one can find a (small) set of input variables *deterministically* based on the structure of the circuit, such that fixing them forces the circuit to output a constant. This implies that small circuits fail to compute functions that cannot be made constant without setting a large number of input variables. The only twist when applying this framework to $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuits is, instead of fixing independent input variables, one now fixes linear functions which in general are no longer independent. We use a folklore result (called the *Folk Lemma* below) that IP can not be made constant by imposing less than $n/2$ linear constraints on the inputs; i.e., IP is not constant on a linear subspace of dimension more than $n/2$.

The main ingredient of the proof is the following lemma, which is the exact analogue of a result of Chaudhuri and Radhakrishnan [3] for AC^0 circuits.

▶ **Lemma 16.** *Let $C(x)$ be an $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuit of depth k and size S, with variable inputs $x_1, \ldots, x_n$ and bottom parity gates $p_1(x), \ldots, p_r(x)$. Then there exists a set of t linearly independent linear restrictions, $t < 5S^{1-4^{-k}}$, such that imposing them on $x_1, \ldots, x_n$ makes $C(x)$ constant (on the restricted space).*

Proof. We will adapt the argument of [3]. The algorithm of [3] constructs a partial assignment to the inputs of an AC^0 circuits so that the output is fixed and the number of fixed variables is small. In particular, it fixes the values of gates at each level (by fixing the bottom variables and propagating the values up the circuit), starting at level 0 (the input level), and proceeding successively up to the output gate at level k. The specific way of fixing these gates ensures that after level i is fixed, all gates at levels $j \leq i$ have both small fan-in and small fan-out (fan-ins and fan-outs are defined with respect to the current partial restriction and gates that are not fixed yet). At the end of such fixing, a so-called "regular" circuit is obtained. Then it is straightforward to show that one can fix an additional small number of variables of such regular circuit to make it output a constant. Our argument proceeds in an almost identical way. However, we fix *parities* in addition to input variables, and once a new parity gate is fixed, we need to fix the free parity gates which linearly depend on the fixed parity gates. This can only possibly reduce the number of parity gates needed to be fixed in the process, thus the original proof works in the setting of $\mathrm{AC}^0 \circ \mathrm{MOD}_2$ circuits as well. We defer the details to the full version of the paper. ◀

Now we are ready to prove the main theorem of this section.

Proof of Theorem 15. Suppose C has size $S < \frac{1}{5}n^{1+4^{-k}}$ (hence, it has at most that many parity gates) and computes the IP function. By Lemma 16, there exists a set of linearly independent linear restrictions of size at most $5S^{1-4^{-k}} < (n^{1+4^{-k}})^{1-4^{-k}} = n^{1-16^{-k}} < n/2$ (for large enough n), under which C becomes a constant function. But by the Folk Lemma, we must impose at least $n/2$ linear restrictions to make IP a constant; a contradiction. ◀

References

1 A. Akavia, A. Bogdanov, S. Guo, A. Kamath, and A. Rosen. Candidate weak pseudorandom functions in $\mathrm{AC}^0 \circ \mathrm{MOD}_2$. In *Proc. 5th ITCS*, pages 251–260, 2014.

2 T. Ando. Truncated moment problems for operators. *Acta Sci. Math. (Szeged)*, 31:319–334, 1970.

3 S. Chaudhuri and J. Radhakrishnan. Deterministic restrictions in circuit complexity. In *Proc. 28th STOC*, pages 30–36, 1996.

4 G. Cohen and I. Shinkar. The complexity of DNF of parities. To appear in ITCS'16, 2016.

5 R.E. Curto and L.A. Fialow. Recursiveness, positivity, and truncated moment problems. *Houston J. Math.*, 17:603–635, 1991.

6 A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. *JCSS*, 46:129–154, 1993. Earlier version in FOCS'87.

7 R. Impagliazzo, R. Shaltiel, and A. Wigderson. Reducing the seed length in the Nisan-Wigderson generator. *Combinatorica*, 26(6):647–681, 2006. Earlier version in FOCS'01.

8 J.C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *JCSS*, 55(3):414–440, 1997.

9 S. Jukna. On graph complexity. *Combinatorics, Probability, and Computing*, 15(6):855–876, 2006.

10 A. Klivans and L. Fortnow. Efficient learning algorithms yield circuit lower bounds. *JCSS*, 75:27–36, 2009.

11 A. Klivans, P. Kothari, and I.C. Oliveira. Constructing hard functions using learning algorithms. In *Proc. 28th CCC*, pages 86–97, 2013.

12 A. Klivans and R. Meka. Moment-matching polynomials. Technical Report TR13-008, ECCC, 2013.

13 S. Kopparty and S. Srinivasan. Certifying polynomials for $AC^0(\oplus)$ circuits, with applications. In *Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 36–47, 2012.

14 N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

15 N. Linial and N. Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.

16 N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

17 N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994. Earlier version in STOC'92.

18 N. Nisan and A. Wigderson. Hardness versus randomness. *JCSS*, 49:149–167, 1994.

19 R. Paturi. On the degree of polynomials that approximate symmetric Boolean functions. In *Proc. 24th STOC*, pages 468–474, 1992.

20 S. Raskhodnikova, D. Ron, A. Shpilka, and A. Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM J. Comput.*, 39(3):813–842, 2009. Earlier version in FOCS'07.

21 A.A. Razborov. Lower bounds on the size of bounded-depth networks over a complete basis with logical addition. *Math. Notes Acad. of Sci. USSR*, 41(4):333–338, 1987.

22 T. Rivlin. *Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory*. John Wiley & Sons, Inc., 1990.

23 R. Servedio and E. Viola. On a special case of rigidity. Manuscript, 2012.

24 R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudo-random generator. *J. ACM*, 52(2):172–216, 2005. Earlier version appeared in FOCS'01.

25 R. Shaltiel and E. Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010.

26 R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. 19th STOC*, pages 77–82, 1987.

27 C. Umans. Pseudo-random generators for all hardnesses. *JCSS*, 67(2):419–440, 2003. Earlier version appeared in CCC'02.

28 L.G. Valiant. Graph-theoretic arguments in low-level complexity. In *Proc. 6th MFCS*, volume 53 of *LNCS*, pages 162–176. Springer, 1977.

29 I. Volkovich. On learning, lower bounds and (un)keeping promises. In *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 1027–1038. Springer, 2014.

Lower Bounds for Nondeterministic Semantic Read-Once Branching Programs*

Stephen Cook[1], Jeff Edmonds[2], Venkatesh Medabalimi[1], and Toniann Pitassi[1]

1 **University of Toronto, Computer Science Department, Toronto, ON, Canada**
sacook@cs.toronto.edu
2 **York University, Computer Science Department, Toronto, ON, Canada**
jeff@cs.york.ca
3 **University of Toronto, Computer Science Department, Toronto, ON, Canada**
venkatm@cs.toronto.edu
4 **University of Toronto, Computer Science Department, Toronto, ON, Canada**
toni@cs.toronto.edu

Abstract

We prove exponential lower bounds on the size of semantic read-once 3-ary nondeterministic branching programs. Prior to our result the best that was known was for D-ary branching programs with $|D| \geq 2^{13}$.

1998 ACM Subject Classification F.2.3 Tradeoffs between Complexity Measures

Keywords and phrases Branching Programs, Semantic, Non-deterministic, Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.36

1 Introduction

A major question in complexity theory is whether polynomial-time is the same as log-space or nondeterministic log-space. One approach to this problem is to study time/space tradeoffs for problems in P. For example, for natural problems in P, does the addition of a space restriction prevent a polynomial time solution? In the uniform setting, time-space tradeoffs for SAT were achieved in a series of papers [7, 15, 8, 9]. Fortnow-Lipton-Viglas-Van Melkebeek [9] shows that any algorithm for SAT running in space $n^{o(1)}$ requires time at least $\Omega(n^{\phi-\epsilon})$ where ϕ is the golden ratio $((\sqrt{5}+1)/2)$ and $\epsilon > 0$. Subsequent works [18, 6] improved the time lower bound to greater than $n^{1.759}$.

In the nonuniform setting, the standard model for studying time/space tradeoffs is the *branching program*. In this model, a program for computing a function $f(x_1, \ldots, x_n)$ (where the variables take values from a finite domain D) is represented by a directed acyclic graph with a unique source node called the start node. Each nonsink node is labelled by a variable and the edges out of a node correspond to the possible values of the variable. Each sink node is labelled by an output value. For Boolean functions, there is one sink node called the accept node (or 1-node), and all other sink nodes are rejecting nodes. Executing the program on an input corresponds to following a path from the start node, using the values of the input variables to determine which edges to follow. The output of the program is the value labelling the sink node reached. A D-ary branching program is deterministic if each non-sink node has exactly D edges, one for every value in D.

* This work was supported by the Natural Science and Engineering Research Council of Canada.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 36; pp. 36:1–36:13

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

The length of a branching program is the number of edges in the longest path. It is clear that length of a branching program can be seen as a measure of computation time. The size of a branching program is the number of nodes in the program. For a boolean function f_n of n variables, let $BP(f_n)$ denote the minimum size of a branching program computing f_n. $BP(f_n)$ is closely related to the space complexity $S(f_n)$ of a non-uniform Turing machine computing f_n: $S(f_n) = O(\log(\max\{BP(f_n), n\}))$ and $BP(f_n) = 2^{O(\max\{S(f_n), \log n\})}$ [5, 17]. This motivates the study of branching program size lower bounds. In particular, size lower bounds on length restricted branching programs translate to time/space tradeoffs.

The state of the art time/space tradeoffs for branching programs were proven in the remarkable papers by Ajtai [1] and Beame-et-al [3]. In the first paper, Ajtai exhibited a polynomial-time computable Boolean function such that any sub-exponential size deterministic branching program requires superlinear length. This result was significantly improved and extended by Beame-et-al who showed that any sub-exponential size randomized branching program requires length $\Omega(n\frac{\log n}{\log\log n})$.

Lower bounds for nondeterministic branching programs have been more difficult to obtain. In this model, there can be several arcs (or no arcs) out of a node with the same value for the variable associated with the node. An input is accepted if there exists at least one path consistent with the input from the source to the 1-node. A nondeterministic branching program computes a function f if its accepted inputs are exactly equal to $f^{-1}(1)$. From here on, we shall restrict our attention to non-deterministic branching programs.

Length-restricted nondeterministic branching programs come in two flavors: *syntactic* and *semantic*. A length l syntactic model requires that every path in the branching program has length at most l, and similarly a read-k syntactic model requires that every path in the branching program reads every variable at most k times. In the less restricted semantic model, the requirement is only for *consistent* accepting paths from the source to the 1-node; that is, accepting paths along which no two tests $x_i = d_1$ and $x_i = d_2$, $d_1 \neq d_2$ are made. This is equivalent to requiring that for every accepting path, each variable is read at most k times. Thus for a nondeterminsitic read-k semantic branching program, the overall length of the program can be unbounded.

Note that any syntactic read-once branching program is also a semantic read-once branching program, but the the opposite direction does not hold. In fact, Jukna [11] proved that semantic read-once branching programs are exponentially more powerful than syntactic read-once branching programs, via the "Exact Perfect Matching"(EPM) problem. The input is a (Boolean) matrix A, and A is accepted if and only if every row and column of A has exactly one 1 and rest of the entries are 0's i.e if it's a permutation matrix. Jukna gave a polynomial-size semantic read-once branching program for EPM, while it was known that syntactic read-once branching programs require exponential size [14, 13].

Lower bounds for syntactic read-k (nondeterministic) branching programs have been known for some time [16, 4]. However, for *semantic* nondeterministic branching programs, even for read-once, no lower bounds are known for polynomial time computable functions for the $|D| = 2$ case. The best lower bound known prior to our work is an exponential lower bound for semantic read-once (nondeterministic) $|D|$-way branching programs, where $|D| = 2^{13}$ [10]. In fact this lower bound actually holds more generally for semantic read-k but where $|D| = 2^{3k+10}$.

Jukna obtains his result by showing that any time restricted semantic branching program of small size has a large rectangle in $f^{-1}(1)$. He uses the explicit function of computing the characteristic function of a linear code having minimum distance $m + 1$ defined over $GF(q)$. Given a parity matrix Y,the function $g(Y, x) = 1$ iff x is a codeword. Since codewords in a

linear code of minimum distance $m+1$ can only have an m-rectangle of size 1 he argues that a time restricted branching program of length kn computing g requires a size of $2^{\Omega(n/k^2 4^k)}$. This exponential lower bound can be obtained whenever D is sufficiently large in comparison to k, specifically for $|D| = q \geq 2^{3k+10}$.

Jukna's result is an improvement over exponential lower bounds with a domain requirement of $2^{2^{ck}}$ obtained in [2]. Beame et.al [2] obtain their result by characterizing the function computed by a time restricted branching program of small size as a union of shallow decision forests where the size of the union depends on the size of the branching program. Each shallow forest is then shown to be representable by a collection of *small* number of βn-pseudo-rectangles in $f^{-1}(1)$. (Pseudo-rectangles are a generalization of what we call embedded rectangles later). This gives a representation of the branching program as a union of small (in the size 's') number of βn-pseudo-rectangles. Now, if for some function f the maximum size of a βn-pseudo-rectangle is $|D|^{(1-\psi_f(\beta))n}$ and the number of *yes*-instances $|f^{-1}(1)| \geq |D|^{(1-\eta(f))n}$ then the number of βn-pseudo-rectangles will be at least $|D|^{(\psi_f(\beta)-\eta(f))n}$. This yields an exponential lower bound on s for sufficiently large $|D|$ whenever $(\psi_f(\beta)-\eta(f))$ is bounded away from 0 by some $\epsilon > 0$. They then exhibit an explicit function with this property. Their function $QF_M : GF(q^n) \to \{0,1\}$ is based on quadratic forms using a modified Generalized Fourier Transform matrix. They show that there exists a constant $c > 0$ such that for all k and $\epsilon \in (0,1)$, if $D \geq 2^{2^{\frac{c}{\epsilon}k}}$ then a non-deterministic BP of length kn computing QF_M needs size at least $S = 2^{n \log^{1-\epsilon}|D|}$. For the specific case of $k = 1$, it can be shown that if their analysis of maximum size of βn-pseudo-rectangles in QF_M is tight, a domain size of at least $|D| \geq 2^{64}$ is needed.

Our main result is an exponential lower bound on the size of semantic read-once non-deterministic branching programs for a polynomial time decision problem f for 3-ary inputs. Similar in spirit to these previous results [10, 2] we show that a small sized semantic read once branching program is bound to have a large rectangle in $f^{-1}(1)$.

($\star$) However in addition, we show that one can always find a *balanced* rectangle in $f^{-1}(1)$ of size r^2 where r is some large constant.

A balanced rectangle is one which is reasonably close to being a square.

The particular polynomial time decision problem we use to prove the lower bound is: to decide if a polynomial over a finite field K evaluates to a value less than a certain threshold at a given input. The input is a pair (u, x) where u is the description of a degree $d-1$ polynomial over $[K]$ and $x \in [K]$, and we want to accept if and only if $u(x) < K^{1-\delta}$. We actually prove a stronger theorem: with high probability over all polynomials u, any nondeterministic semantic read-once branching program for what we shall call $Poly_u$ (along with a hyperplane constraint) requires exponential size. That is, even if the branching program knows the polynomial u, for a typical u it cannot efficiently do polynomial evaluation. The main properties of polynomials over finite fields we are using are polynomial interpolation, and Lemma 7, which might be interpreted to mean something like: the spread of values of a typical random polynomial of degree d over a field K is roughly close to being uniform over K, provided K is sufficiently large.

Continuing with the above observation ($\star$) that we can find a balanced rectangle in $f^{-1}(1)$ for a function with a small semantic read once branching program, since the number of balanced rectangles of a certain size $d = r^2$ is *small* and since each one of them can be a rectangle in $f^{-1}(1)$ for a relatively *small* number of degree d polynomials over K as a consequence of polynomial interpolation, we argue that there must be a polynomial with no balanced rectangle of this size in $f^{-1}(1)$ and hence the branching program computing it

should be *large*. A key idea of this argument is that for a balanced rectangle the sum of the lengths of the rectangle can be at most a *small* fraction of its area.

By a simple padding argument, we can modify our problem $Poly_u$ and actually achieve the lower bound for domain size $2+\epsilon$ for arbitrarily small $\epsilon > 0$. In this model, we can define the problem to have $N = n + M$ variables, $M = \Theta(N)$ of them with domain size 3 and the rest, with domain size 2, do not affect the output. In section 5, we show why it might be harder to prove lower bounds for semantic read-once branching programs when $|D| = 2$ by showing how these branching programs can altogether evade having an exponential number of states in many purported choices of bottleneck layer by giving polynomial upper bounds.

2 Definitions

Throughout this article, D denotes a finite set. For finite set N, D^N is the set of maps from N to D. An element of N is called a *variable index* or simply an *index*. We normally take N to be $[n]$ for some integer n, and write D^N for $D^{[n]}$. If $A \subseteq N$, a point $\sigma \in D^A$ is a *partial input* on A. For a partial input σ, $fixed(\sigma)$ denotes the index set A on which it is defined and $unfixed(\sigma)$ denote the set $N - A$. If σ and π are partial inputs with $fixed(\sigma) \cap fixed(\pi) = \emptyset$, then $\sigma\pi$ denote the partial input on $fixed(\sigma) \cup fixed(\pi)$ that agrees with σ on $fixed(\sigma)$ and with π on $fixed(\pi)$.

For $x \in D^N$ and $A \subseteq N$, the *projection* x_A of x onto A is the partial input on A that agrees with x. For $S \subseteq D^N$, $S_A = \{x_A \mid x \in S\}$.

2.1 Nondeterministic Semantic Read-Once Branching Programs

Let $f : D^N \to \{0, 1\}$ be a boolean function whose input is given in $|D|$-ary. Let the input variables be $x_1, \ldots, x_n$ where $x_i \in D$ for all $i \leq n$. A $|D|$-way nondeterministic branching program (for f) is an acyclic directed graph G with a distinguished source node q_{start} and a distinguished sink node (the accept node) q_{accept}. We refer to nodes as *states*. Each non-sink state is labeled with some input variable x_i, and each edge directed out of a state is labeled with some value $b \in D$ for x_i. For each $Z \in D^N$, the branching program accepts Z if and only if there exists at least one (directed) path starting at the q_{start} and leading to the accepting state q_{accept}, and such that all labels along this path are consistent with Z. The size of a branching program is the number states (i.e. nodes) in the graph.

A branching program is *semantic read-k* if for every path from q_{start} to q_{accept} that is consistent with some input, each variable occurs at most k times along the path. For the read-once case, a semantic branching program allows variables to be read more than once, but each accepting path may only query each variable at most once.

2.2 Polynomial Evaluation Problem

Our hard computational problem is the polynomial evaluation problem, $Poly$, with parameters K, d, δ, where $0 < \delta < 1$. The input is a pair (u, x) where $u \in [K]^d$ specifies a degree $d-1$ polynomial over the field $[K]$ (K a prime power), and $x \in [K]$ specifies a field value. $Poly(u, x) = 1$ if and only if the polynomial specified by u on input x evaluates to a number less than $K^{1-\delta}$. (We compare two field elements by comparing them using the natural ordering on ternary strings.)

We will work with $|D|$-ary branching programs (with $|D|$ prime), and let $K = |D|^n$. The input will be given as a vector in $D^{(d+1)n}$. The first dn coordinates specify u and the last n coordinates specify x. Thus the total input length is $(d+1)n$. In the remainder of the

paper, $|D| = 3$, and thus the parameters of $Poly$ are d, δ, n. Both d and δ will be fixed constants. Let $Poly_u$ denote the polynomial evaluation problem with parameters d, δ, n where the polynomial u is fixed.

The actual lower bounds we show will be for a *sensitive* function f_u obtained from $Poly_u$ as follows. Let $a \in GF(q)$ where $q = |D|$ is a prime number. Let $h : D^n \to \{0, 1\}$ be the characteristic function of the hyperplane at a:

$$h_a(x) = 1 \quad \text{iff} \quad x_1 + x_2 + ... + x_n = a \mod q\,.$$

Fix an element $a(u)$ for which h_a accepts the largest number of vectors accepted by $Poly_u$ and define the function

$$f_u(x) = Poly_u(x) \wedge h_{a(u)}(x)\,.$$

We call f_u sensitive because it has the property that changing the value of exactly one variable in a yes input always gives an input vector that is a no instance. As a result any two accepted inputs differ in the value of at least two variables. Similarly for the polynomial evaluation problem $Poly$, where the coefficient vector u is part of the input, we define $f(u, x) = Poly(u, x) \wedge h_{a(u)}(x)$, which is sensitive in x.

2.3 Rectangles and Embedded Rectangles

We use the same definitions and conventions as in [3]. A product $U \times V$ is called a (combinatorial) rectangle. If $A \subseteq N$ is an index subset and $Y \subseteq D^A$ and $Z \subseteq D^{N-A}$, then the product set $Y \times Z$ is naturally identified with the subset $R = \{\sigma\rho \mid \sigma \in Y,\ \rho \in Z\}$ of D^N, and a set of this form is called a *rectangle* in D^N.

An *embedded rectangle* R in D^N is a triple $(\pi_{red}, \pi_{white}, C)$ where π_{red}, π_{white} are disjoint subsets of N, and $C \subseteq D^N$ satisfies: (i) The projection $C_{N-\pi_{red}-\pi_{white}}$ consists of a single partial input w, (ii) if $\tau_1 \in C_{\pi_{red}}$ and $\tau_2 \in C_{\pi_{white}}$, then the point $\tau_1\tau_2 w \in C$. C is called the *body* of R. The sets π_{red}, π_{white} are called the *feet* of the rectangle; the sets $C_{\pi_{red}}$ and $C_{\pi_{white}}$ are the *legs*, and w is the *spine*. We can also specify an embedded rectangle by its feet, legs and spine: $(\pi_{red}, \pi_{white}, A, B, w)$ where π_{red}, π_{white} are the feet, $A = C_{\pi_{red}}, B = C_{\pi_{white}}$ are the legs, and w is the spine.

We will sometimes refer to A as the *red* side of the rectangle and to B as the *white* side of the rectangle. The *size* of the rectangle is $|A| \cdot |B|$, and the dimension of the rectangle is m_r-by-m_w where $m_r = |\pi_{red}|$ and $m_w = |\pi_{white}|$.

3 Lower Bound for $|D| = 3$

▶ **Theorem 1.** *There exists constants d, δ such that for sufficiently large n, for a random u, with probability greater than 1/4, any 3-ary nondeterministic semantic read-once branching program for f_u requires size at least $2^{\Omega(n)}$.*

▶ **Corollary 2.** *There exists constants d, δ such that for sufficiently large n, any 3-ary nondeterministic semantic read-once branching program for $f(u, x)$ with parameters d, δ, n requires size at least $2^{\Omega(n)}$.*

Overview of Proof. Call a degree $d - 1$ polynomial "good" if the fraction of accepting instances is roughly what you would expect from a random function; that is, the fraction of yes instances is at least $\frac{1}{2}K^{-\delta}$. Lemma 7 shows that at least half of all degree $d - 1$ polynomials are good.

The main lemma (Lemma 3) shows that for all good polynomials $Poly_u$ to their corresponding sensitive function f_u, we can associate with every size $s = 2^{o(n)}$ branching program $\mathcal{P}$ computing f_u, an m_r-by-m_w embedded rectangle $R_{\mathcal{P}}$ of size r^2, where r will be a large constant, and m_r and m_w will be roughly equal, and will each be a constant fraction of n. For simplicity of calculations for now, assume that $m_r = m_w = m$. The rectangle will have the property that $\mathcal{P}$ accepts every input in $R_{\mathcal{P}}$; in other words, $R_{\mathcal{P}}$ is a 1-rectangle of $\mathcal{P}$. Choosing $d = r^2$, each rectangle of size r^2 can be a 1-rectangle for very few degree $d-1$ polynomials – at most a $|D|^{-n\delta r^2}$ fraction of all degree $d-1$ polynomials. (This is Lemma 6.) Secondly, the total number of such rectangles is fairly small – of size roughly $|D|^{O(rm)}$ (Lemma 5). The key point is that the *number* of rectangles is roughly $|D|^{2rm}$ – the exponent grows linearly in r. (More precisely it grows linearly in the sum of the lengths of the sides of the rectangle, $|A| + |B|$). But on the other hand, the probability that a degree $d = r^2$ polynomial takes on values less than $K^{1-\delta}$ within the rectangle is roughly $|D|^{-mr^2}$ – that is, the exponent grows *quadratically* with r. (More precisely it grows linearly in the size of the rectangle $|A|.|B|$). Because $|D|^{-n\delta r^2}|D|^{O(rn)}$ is less than $1/4$, this implies that many good degree $d-1$ polynomials have no size r^2 1-rectangle, thus proving the theorem.

Note that we set our parameters so that the area of the rectangle $R_{\mathcal{P}}$ is at least the degree d of the polynomial u. (Thus $r^2 \geq d$.) A crucial point in the above argument is that the sum of the lengths of the sides of $R_{\mathcal{P}}$ must be at most a fraction of its area. This requires that the rectangle is reasonably close to being square. We put extra effort into making sure that the rectangle is square (without compromising too much of its size in order to make it square). This enables us to achieve domain size 3; a somewhat simpler argument achieves domain size 5.

▶ **Lemma 3** (Main Lemma). *Let $f : D^n \to \{0,1\}$ be any sensitive boolean function such that the density of 1's is at least $\frac{1}{2|D|}K^{-\delta}$. Suppose that the following inequalities are satisfied for our parameters:*

(1) $m_w = 4m_r = \gamma n$;
(2) $|D|^{m_r} \leq |D|^{m_w}(1/2 - 2\gamma)^{m_w}$;
(3) $r \leq \frac{1}{4|D|s}(1/2 - \gamma)^{m_r}|D|^{m_r - \delta n}$.

Then if $\mathcal{P}$ is a $|D|$-way nondeterministic semantic read-once branching program of size s for f then there is an m_r-by-m_w embedded rectangle $R = (\pi_{red}, \pi_{white}, A, B, w)$ such that every input in R is accepted by $\mathcal{P}$, and where $|A|, |B| = r$.

Proof. Let f be a sensitive function such that the density of 1's is at least $\frac{1}{2|D|}K^{-\delta}$. Suppose there is a size s nondeterministic semantic read-once branching program, $\mathcal{P}$ for f. Let S_0 be the set of inputs that are accepted by $\mathcal{P}$; since $\mathcal{P}$ is assumed to be correct for all inputs of f, we have $|S_0| \geq \frac{1}{2|D|}K^{-\delta}|D|^n$. For each accepted instance $I \in S_0$, fix one accepting path, p_I, in the branching program. Since the function is sensitive each of the n variables must be read along any accepting path. For if some variable is not read along a computation path then changing the value of that variable alone would produce an accepting instance. However, this can't be the case for a sensitive function since any two accepted inputs will have to differ in at least two positions. So each of the n variables must be read along this path exactly once and thus each accepting instance I has an associated permutation π_I of the n variables associated with its accepting path p_I. Designate state q_I as the state in p_I which occurs just after the first half of the variables in π_I. Now define q to be the most common designated state (over all accepting inputs $I \in S_0$), and let $S_1 \subseteq S_0$ denote the corresponding set of inputs whose designated state is q. Thus for each input I in S_1, there is an accepting path

p_I that passes through state q. Because $\mathcal{P}$ has size s, it follows that

$$|S_1| \geq |S_0|/s \geq \frac{1}{2|D|s} K^{-\delta} |D|^n = \frac{1}{2|D|s} |D|^{n-\delta n} \tag{1}$$

We now want to pick two subsets of coordinates $\pi_{red} \subseteq N$ and $\pi_{white} \subseteq N$, of size m_r and m_w respectively, and a set $S^* \subseteq S_1$ of inputs with the property that for every input $I \in S^*$, and associated accepting path p_I, not only does it pass through state q, but every coordinate in π_{red} is read before state q, and every coordinate in π_{white} read at or after state q.

We will first pick π_{red} greedily. For each $I \in S_1$, at least $n/2$ of the n coordinates in p_I occur in π_I before reaching state q, and thus there is some coordinate i such that for at least half of the inputs $I \in S_1$, i occurs in π_I before reaching state q. After choosing the first coordinate, there are at least $|S_1|/2$ inputs remaining. Continue greedily until we pick m_r coordinates, π_{red}, always choosing the most popular coordinate that occurs in π_I before reaching state q. By averaging, when the i^{th} coordinate, $i \leq m_r < \gamma n$ is chosen, the fraction of inputs that remain is at least $\frac{(n/2-i)}{(n-i)} \geq \frac{(n/2-\gamma n)}{(n-\gamma n)} \geq \frac{(n/2-\gamma n)}{n} = (1/2-\gamma)$. Let $S_2 \subseteq S_1$ denote the set of inputs such that all coordinates in π_{red} are read before reaching q. It follows that

$$|S_2| \geq (1/2-\gamma)^{m_r} |S_1| \tag{2}$$

By assumption (3) in the statement of the Lemma, we have

$$r \leq \frac{1}{4|D|s} (1/2-\gamma)^{m_r} |D|^{m_r - \delta n} \tag{3}$$

Then from (1), (2), and (3) we have

$$|S_2| \geq 2r|D|^{n-m_r} \tag{4}$$

For each $w \in D^{n-\pi_{red}}$, the average number of extensions of w in S_2 is $2r$. We want to prune S_2 such that every $w \in D^{n-\pi_{red}}$ has at least r extensions. To do this, define $S_3 \subseteq S_2$, where we remove all inputs $(w, *)$ from S_2 such that w has less than r extensions in S_2. Since we delete at most $r|D|^{n-m_r}$ elements from S_2, and $|S_2| \geq 2r|D|^{n-m_r}$, it follows that

$$|S_3| \geq r|D|^{n-m_r} \tag{5}$$

Next we will choose m_w coordinates, π_{white} in the same greedy fashion, and let S_4 denote the set of all inputs in S_3 such that all coordinates in π_{white} are read after reaching q. Again by averaging,

$$|S_4| \geq (1/2-2\gamma)^{m_w} |S_3| \tag{6}$$

We will express S_4 as the disjoint union of sets R_w: choose a value w for the coordinates outside of $\pi_{red} \cup \pi_{white}$. The corresponding set $R_w \subseteq S_4$ consists of all inputs (α, w, β) such that α is an assignment to the variables in π_{red}, β is an assignment to the variables in π_{white}, and $(\alpha, w, \beta) \in S_4$.

▶ **Lemma 4.** *For each w: (i) R_w is an embedded rectangle and (ii) as long as R_w is not empty, the size of its red leg is at least r.*

Proof. We will first show that R_w is an embedded rectangle. Let $S_{red} \subseteq D^{\pi_{red}}$ be the projection of R_w onto the coordinates of π_{red} and let $S_{white} \in D^{\pi_{white}}$ be the projection of R_w onto the coordinates of π_{white}. Setting $A = S_{red}$, $B = S_{white}$ and $w = w$, we claim

that R_w is equal to the embedded rectangle defined by $(\pi_{red}, \pi_{white}, A, B, w)$. To see this, consider $x, x' \in A$ and $y, y' \in B$ such that $xyw \in R_w$, and $x'y'w \in R_w$. Let I be the input corresponding to xyw and let p_I be the corresponding path going through state q. Note that in p_I the x-variables are all read prior to reaching q, and the y-variables are read after reaching q, and there is some split of the w variables into w_1, w_2 where the w_1 variables are read prior to q and the w_2 variables are read after q. Similarly, let I' be the input corresponding to $x'y'w$ and let $p_{I'}$ be the corresponding path. There is now a possibly different split of w into w_1', w_2', so x', w_1' are read before q and y', w_2' are read after q. We claim that $xy'w \in R_w$: consider the path (x, w_1) (the first half of p_I) and (y', w_2') (the second half of $p_{I'}$). This path must be consistent since w_1 and w_2' are consistent and x, y' are on disjoint variables. Thus there is an input consistent with this path; it is an accepting path going through q and consistent with w; the variables in π_{red} are all read before q, and the variables in π_{white} are all read after q. Thus it is in R_w. An analogous argument shows that $x'yw \in R_w$. Thus R_w is an embedded rectangle.

Secondly we will show (ii) for each $R_w \subseteq S_4$, the size of the red leg is at least r. (That is, $|A| \geq r$.) Consider a nonempty rectangle R_w with red leg A, white leg B and spine w. Recall that the inputs in S_3 consist of a partial input $w+ \in D^{N-\pi_{red}}$ together with a set $A \subseteq D^{\pi_{red}}$ such that $|A| \geq r$. We obtain S_4 from S_3 by selecting m_w coordinates from $N - \pi_{red}$, one at a time, choosing each coordinate greedily, where a coordinate is chosen if it is read *after* state q in the most inputs. Consider a block of inputs $(A, w+) \in S_3$. If some input $(\alpha, w+) \in (A, w+)$ survives, then all coordinates in π_{white} that were chosen must all be read after state q on input $(\alpha, w+)$. But this means that for every input $(\alpha', w+) \in (A, w+)$, all coordinates in π_{white} are also read after q. (Otherwise, some coordinate would be read twice along this accepting input, violating the read-once condition.) Thus, either the entire block $(A, w+)$ is in S_4, or the entire block is removed from S_4.

Now let $R_w = (\pi_{red}, \pi_{white}, A, B, w) \subseteq S_4$ be a nonempty rectangle, $w \in D^{N-\pi_{red}-\pi_{white}}$. R_w is obtained by taking the union of (nonempty) blocks $(A', w+) \in S_4$, $w+ \in D^{N-\pi_{white}}$. Since as we argued above, for each such block, $|A'| \geq r$, it follows that $|A| \geq r$ as well. ◀

Let r_{avg} denote the average size of the white leg of the rectangle over all rectangles $R_w \subseteq S_4$. It is easy to see that $|D|^{n-m_w} r_{avg} \geq r|D|^{n-m_r}(1/2 - 2\gamma)^{m_w}$. It follows that $r_{avg} \geq r$ if $|D|^{m_w - m_r}(1/2 - 2\gamma)^{m_w} \geq 1$. The latter inequality follows from condition (2). Thus, by condition (2) assumed in the hypothesis of Lemma 3, we can pick some setting w^* to the remaining $n - m_r - m_w$ uncoloured coordinates (the coordinates that are not in π_{red} or π_{white}) such that the white leg of the rectangle R_{w^*} has size at least $r_{avg} = r$. Let S^* equal R_{w^*}. By construction, both the red leg of $S^* = R_{w^*}$ and the white leg of R_{w^*} have size at least r. Prune S^* so that each leg has size exactly r, thus completing the proof of the lemma. ◀

▶ **Lemma 5.** *Let $\mathcal{R}$ be the set of all m_r-by-m_w embedded rectangles over D^N such that $|A| = |B| = r$, where $m_w = \gamma n$ and $m_r = m_w/4$. Then $|\mathcal{R}| \leq (e/\gamma)^{\frac{5}{4}m_w} |D|^{\frac{5}{4}rm_w + m_w/\gamma}$.*

Proof. The number of choices for π_{red}, the coordinates of A, is $\binom{n}{m_r}$. Given π_{red}, we choose r vectors from the $|D|^{m_r}$ possible values for the elements of A. Thus the total number of possible sets A is at most $\binom{n}{m_r}|D|^{rm_r}$. Similarly the number of choices for the set B is at most $\binom{n}{m_w}|D|^{rm_w}$. The number of choices for $w \in D^{N-\pi_{red}-\pi_{white}}$ is $|D|^{n-m_r-m_w}$. Thus $|\mathcal{R}|$ is at most $\binom{n}{m_r}\binom{n}{m_w}|D|^{rm_r}|D|^{rm_w}|D|^{n-\frac{5}{4}m_w}$. Using the inequality $\binom{n}{k} \leq (\frac{en}{k})^k$ we conclude the number of choices for $|\mathcal{R}|$ is at most $(en/m_w)^{m_w}(4en/m_w)^{\frac{1}{4}m_w}|D|^{n-\frac{5}{4}m_w}|D|^{\frac{5}{4}rm_w} \leq (e/\gamma)^{\frac{5}{4}m_w}|D|^{\frac{5}{4}rm_w + m_w/\gamma}$ ◀

▶ **Lemma 6.** *Define the predicate $Good(R,u)$ to be true if for every input x in the rectangle R, the polynomial u on input x is less than $K^{1-\delta}$ (i.e. $Poly_u(x)$ is true). Then for all embedded rectangles R of size d, $Pr_u[Good(R,u)] \leq p$ where $p = |D|^{-\delta nd}$.*

Proof. Assume $Good(R,u)$. Suppose that $|R| = d$ and let $B' \in [K^{1-\delta}]^d$ specify a vector of d accepting values. Let $Good_{B'}(R,u)$ to be the event that for all $x \in R$, $Poly_u(x) = B'(x)$. Then $Pr_u[Good(R,u)] = K^{(1-\delta)d} \cdot Pr_u[Good_{B'}(R,u)]$.

To bound $Pr_u[Good_{B'}(R,u)]$, suppose that it is true that $\forall x \in R$, $F_u(x) = \sum_{i<d} u_i x^i = B'(x)$. Note that this fixes the output of the degree $d-1$ polynomial for d values of x. By interpolation, this uniquely determines the polynomial, u'. Thus, $Pr_u[Good_{B'}(R,u)] = Pr_u[u = u'] = K^{-d} = |D|^{-nd}$. Overall, $Pr_u[Good(R,u)] \leq K^{(1-\delta)d}|D|^{-nd} = |D|^{n(1-\delta)d}|D|^{-nd} = |D|^{-\delta nd}$. This completes the proof of Lemma 6. ◀

▶ **Lemma 7.** *For a random u,for fixed parameters d, δ the probability that $Poly_u(x)$ does not accept a $(1 \pm o(1))K^{-\delta}$ fraction of all the inputs is at most $o(1)$. (Here both $o(1)$ are $K^{-(1-\delta)/3}$.)*

Proof. Randomly choose the coefficients $u \in [K]^d$ of the $d-1$ degree polynomial. For each instance $x \in [K]$ (and value $b \in [K]$), let $A_{\langle x,b\rangle}$ denote the event that the output of this polynomial on input x is b. Let a_x denote the event that this value is less than $K^{1-\delta}$ so that x is a yes instance. Let $Y = \sum_{x\in K} a_x$ denote the number of yes instances for the chosen u. Note $p = \Pr_u[a_x] = K^{1-\delta}/K = K^{-\delta}$ because just choosing the constant coefficient u_0 of the polynomial randomly makes the polynomial's output on x uniformly random in $[K]$. Hence, by linearity of expectation $\overline{Y} = \text{Exp}[Y] = K \cdot \Pr_u[a_x] = K^{1-\delta}$. We show that the $A_{\langle x,b\rangle}$ events for different x are d-wise independent as follows. Consider any subset $\{x_1, x_2, \ldots, x_d\} \subset [K]$ of the instances. Knowing the value of the polynomial at each of these instances, by interpolation, uniquely determines the coefficients u of the polynomial. Hence, if all you know about u is the values on $d-1$ of these instances, then the value on the remaining is still uniformly random within $[K]$. Formally stated, $\Pr_u[A_{\langle x_d,b_d\rangle} \mid A_{\langle x_1,b_1\rangle}, \ldots, A_{\langle x_{d-1},b_{d-1}\rangle}] = \Pr_u[A_{\langle x_d,b_d\rangle}]$. Not fully knowing the value of the first $d-1$ of the instances, but only that their value is small, give you even less information. Hence, $\Pr_u[a_{x_d} \mid a_{x_1}, \ldots, a_{x_{d-1}}] = \Pr_u[a_{x_d}]$. It follows that $\Pr_u[a_{x_1} \wedge \ldots \wedge a_{x_d}] = \Pr_u[a_{x_1}] \cdot \ldots \cdot \Pr_u[a_{x_d}]$. Because the a_x events are d-wise independent, it follows that the d^{th} order standard deviation of their sum Y is the same as it would be if they were completely independent events. We, however, only need to consider the variance. More formally, for each x, let a'_x be an independent event with probability $K^{-\delta}$ of success and $Y' = \sum_{x\in K} a'_x$. The variance is $\text{Var}[Y] = \text{Exp}_u[(Y-\overline{Y})^2] = \text{Exp}_u[(\sum_x a_x - \overline{Y})^2]$. The non-linear part of this is $\text{Exp}_u[(\sum_x a_x)^2] = \sum_{x,x'} \text{Exp}_u[a_x \cdot a_{x'}]$, which we know from pair-wise independence is $\sum_{x,x'} \text{Exp}_u[a_x] \cdot \text{Exp}_u[a_{x'}] = \sum_{x,x'} \text{Exp}_u[a'_x] \cdot \text{Exp}_u[a'_{x'}]$. The same computation for the a'_x, gives that $\sigma^2 = \text{Var}[Y] = \text{Var}[Y'] = K \cdot p(1–p) \approx K\dot{K}^{-\delta} = K^{1-\delta} = \overline{Y}$. By Chebycheff's inequality, $\forall \eta > 0$ we have $Pr_u(|Y - \overline{Y})| \geq \eta\sigma) < \frac{1}{\eta^2}$. Setting $\eta = \overline{Y}^{\frac{1}{6}}$, gives $Pr_u(Y \notin (1 \pm \overline{Y}^{-\frac{1}{3}})\overline{Y}) \leq \overline{Y}^{-\frac{1}{3}}$. ◀

We are now ready to complete the proof of the theorem. Call a polynomial u "good" if $Poly_u$ accepts at least a $\frac{1}{2}K^{-\delta}$ fraction of all inputs. By Lemma 7, we know that at least half of all u's are good. For each good u, the corresponding sensitive function f_u has density at least $\frac{1}{2|D|}K^{-\delta}$. Since f_u is sensitive and has sufficient density Lemma 3 tells us that any small branching program for f_u implies that there exists an m_r-by-m_w embedded rectangle size r^2 that is accepted (assuming that conditions (1), (2), and (3) are satisfied).

On the other hand, by union bound Lemmas 5 and 6 together tell us that at most a $p|\mathcal{R}|$ fraction of degree $d-1$ polynomials u have such m_r-by-m_w embedded rectangles of size r^2

that are accepted. Suppose we can choose a setting of the parameters so that $p|\mathcal{R}| < 1/4$. If follows that for at least $\frac{1}{4}^{th}$ of all good polynomials the corresponding sensitive functions f_u do not have such m_r-by-m_w embedded rectangles of size r^2 that are accepted since the hyperplane constraint $h_{a(u)}(x)$ can only shrink an accepting rectangle. Then by Lemma 3 this implies that at least as many f_u cannot have small branching programs, and thus the theorem is proven.

It is left to show that we can set the parameters such that $p|\mathcal{R}| < 1/4$, and properties (1), (2), and (3) of Lemma 3 are satisfied. We will set the parameters as follows: $|D| = 3$, $m_w = 4m_r = \gamma n$, $\gamma = .01$, $\delta = \gamma/300$, $r = 3000$, and $d = r^2$. To achieve $p|\mathcal{R}| < 1/4$, we require $|D|^{\delta m_w r^2/\gamma - m_w/\gamma - \frac{5}{4}rm_w} > 4(e/\gamma)^{\frac{5}{4}m_w}$. Using $|D| = 3$ and factoring out m_w, it is sufficient if we have $3^{\delta r^2/\gamma - 1/\gamma - \frac{5}{4}r} > 4(e/\gamma)^{\frac{5}{4}}$. With our choice of parameters, this is satisfied for $r \geq 3000$.

For Lemma 3, we also require assumptions (2) and (3). First for (2): $|D|^{m_r} \leq |D|^{m_w}(1/2 - 2\gamma)^{m_w}$. For $|D| = 3$ and $m_w = 4m_r$, this is satisfied. For (3) we require: $r \leq \frac{1}{4|D|}(1/2 - \gamma)^{m_r}|D|^{m_r - \delta n} = \frac{1}{4|D|}(1/2 - \gamma)^{m_r}|D|^{m_r(1-4\delta/\gamma)}$. For $|D| = 3$, $\gamma = .01$, $\delta = \gamma/300$, we have $(1/2 - \gamma)|D|^{(1-4\delta/\gamma)} \geq 1.44$ and thus it suffices to show $r \leq \frac{1}{12}(1.44)^{m_r}/s$. This holds for our choice $r = 3000$ when $s \leq 2^{cm_r} = 2^{cn/(4\gamma)}$ for some $c > 0$ and sufficiently large n. Note that $|D| > 2$ helps us in ensuring assumptions (2) and (3) hold.

4 Conclusion

We have proved an exponential lower bound on the size of non-deterministic semantic read once branching programs computing a polynomial time computable function $f : D^n \to \{0, 1\}$ when $D = \{0, 1, 2\}$ with just three elements. Our contribution is that we bring down the size of the domain required to achieve this. Prior to our result the best that was known was for D-ary branching programs with $|D| \geq 2^{13}$. The explicit function f for which we show the lower bound is the decision problem of determining whether a certain degree d polynomial over a finite field K evaluates to a value less than a certain threshold at a given input (along with a hyperplane constraint). This result brings down the focus to the first non-boolean case, $|D| = 3$ vs the boolean case, $|D| = 2$, since, interestingly the case where D is boolean $\{0, 1\}$ still remains open and no non-trivial lower bounds are known for binary non-deterministic semantic read once branching programs [12]. In the next section we explore the Booelan case.

5 Semantic Branching Programs with $|D| = 2$ can evade Large Bottleneck Rectangles

In this section we show how binary non-deterministic semantic read once branching programs can behave differently by evading lower bounds in certain bottleneck layers by having a small number of states in those layers irrespective of what function $f : \{0, 1\}^n \to \{0, 1\}$ they are computing. The example upper bounds we give demonstrate why it is likely harder to prove lower bounds for semantic read once branching programs with domain size $|D| = 2$.

When the domain size is $|D| > 2$, the technique is to prove that the set of yes instances handled by any one state of the branching program contains a rectangle and then identify a computational problem that has no large rectangle of yes instances. Hence, the rectangle for each state must be small. Because there are exponentially many yes instances and each must be handled by at least one state at a selected *bottle neck* level of the branching program, there must be an exponential number of states at that level. We show here that for domain

size $|D| = 2$, the set of yes instances handled by one state can be quite arbitrary and quite large. This does not mean that the total number of branching program states can necessarily be small. But it does mean that at the one level of the branching program that the prover is hoping to use for a bottleneck, the number of states might be quite small.

A lower bound that attempts to prove that a selected bottleneck level of the branching program must have many states, must start by selecting which level of the branching program will be the level in question. It might do this by specifying how many or which variables have been read so far. Given *any boolean computational problem* with input from $\{0,1\}^n$ and a criteria for choosing the bottle neck level chosen from a wide (but not exhaustive) range of possible choices, we now show how to fool such a lower bound, by giving a branching program that gives a polynomial upper bound on the number states at the selected layer.

The branching program is constructed as follows. For each yes instance $A \in \{0,1\}^n$, we form an accepting branching program path $\langle C_1(A), q(A), C_2(A)\rangle$ where $q(A)$ denotes the state A passes through at the bottleneck level, $C_1(A)$ the path before this level and $C_2(A)$ that after. Note that to get a counter example to a lower bound technique using some bottleneck layer, we don't need to give a full poly-size branching program. We only need the number of states $q(A)$ to be small. It can have exponential number of states before and after this layer. Hence, we will have all of these paths $C_1(A)$ for different yes instances A be completely disjoint from each other. Similarly for $C_2(A)$. These paths only come together and interact at the special layer of states $q(A)$. In order to make the properties of this level more arbitrary, let $A_1, A_2 \subset [n]$ be any partition of the input variables into two parts. Let $C_1(A)$ read all the ones in A_1 and all the zeros in A_2. Let $C_2(A)$ read all the zeros in A_1 and all the ones in A_2. Let $q(A) = \langle u, v\rangle$ be the state, where $u \in [n]$ is the number of ones in A_1 and $v \in [n]$ is the number of the zeros in A_2. Hence only n^2 states are needed in the layer. Note that because we have allowed the computational problem to be arbitrary, other than partitioning its yes instances based on their hamming weights, the sets of instances handled by a state $q(A)$ is completely arbitrary.

Note that as long as A_1, A_2 are comparable in size, for most of the inputs, the incoming path $C_1(A)$ and $C_2(A)$ are of comparable length. However, the purported bottleneck layer for which we give the above upperbound is not identical to the one we use for our lower bound for $|D| \geq 3$ in the sense that the bottleneck states like $q(A)$ do not appear exactly midway through the accepting paths at length $n/2$ on all the paths as is required in Lemma 3. Nevertheless, the upper bound is interesting because for most inputs A the incoming and outgoing paths through a state in the layer are of comparable length.

We will now in two ways, prove that this branching program solves the given computational problem. We will start with a *communication game interpretation.* Think about the algorithm as a game between two players C_1 and C_2 and Charlie who they don't trust. Charlie shows C_1 the ones in the first part A_1 of the input and the zeros in the second part A_2. Assuming he trusts Charlie, this lets C_1 know the entire input. Hence, he can answer *any* question about the input. The only way that Charlie can cheat is to not show all of the entries. In order to verify that he is not lying, C_1 sends to C_2 the number of ones in A_1 and the number of zeros in A_2. C_2 can then check that they have both been shown all of what they were supposed to see.

Now lets consider the *branching program interpretation.* Clearly, the branching program described accepts all yes instances of the given problem, because it has a separate accepting path $\langle C_1(A), q(A), C_2(A)\rangle$ for each yes instance A. What remains is to prove 1) the branching program is semantic read once and 2) that no no instances are accepted. We do this by showing for every pair $\langle A, B\rangle$ of different yes instances that pass through the same bottleneck

states $q(A) = q(B) = \langle u, v \rangle = q$, that the cross path $\langle C_1(A), q, C_2(B) \rangle$ is inconsistent in that it reads some variable twice with different values. Hence, by the definition of *semantic*, it does not matter that this path is not read once and because it is inconsistent, it cannot be accepting a no instance. Because A and B are different, either A_1 and B_1 are different and/or A_2 and B_2 are different. Assume A_1 and B_1 are different. Because A_1 and B_1 have the same number u of ones, there is an element that is one in A_1 and zero in B_1. Hence $C_1(A)$ and $C_2(B)$ both read it. This element is read twice in $\langle C_1(A), q, C_2(B) \rangle$ with different values and so is inconsistent.

So for $D = 2$, presence of a small number of states in a supposed bottleneck layer of a branching program need not imply that there exists a balanced embedded rectangle of accepting instances. In particular, our lower bound finds a rectangle within the set of yes instances handled by narrowing the set down to a subset within which for many variables it is fixed whether it is read before or after the state. However in this upper bound, whether a variable is read before or after the state $q(A)$ is completely determined by whether its value is 0 or 1. Hence, fixing this fixes its value. If this is done for every variable, the set of inputs left in the eventual rectangle identified by this lower bound method is narrowed down to a singleton.

Acknowledgements We thank Paul Beame and Siu Man Chan for helpful discussions and an anonymous reader for many helpful suggestions.

References

1 M. Ajtai. A non-linear time lower bound for boolean branching programs. In *Proceedings 40th FOCS*, pages 60–70, 1999.

2 P. Beame, T.S. Jayram, and M. Saks. Time-space tradeoffs for branching programs. *J. Comput. Syst. Sci*, 63(4):542–572, 2001.

3 P. Beame, M. Saks, X. Sun, and E. Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *Journal of the ACM*, 50(2):154–195, 2003.

4 Allan Borodin, A Razborov, and Roman Smolensky. On lower bounds for read-k-times branching programs. *Computational Complexity*, 3(1):1–18, 1993.

5 Alan Cobham. The recognition problem for the set of perfect squares. In *Switching and Automata Theory, 1966., IEEE Conference Record of Seventh Annual Symposium on*, pages 78–87. IEEE, 1966.

6 Scott Diehl and Dieter Van Melkebeek. Time-space lower bounds for the polynomial-time hierarchy on randomized machines. *SIAM Journal on Computing*, 36(3):563–594, 2006.

7 L. Fortnow. Nondeterministic polynomial time versus nondeterministic logarithmic space: Time space tradeoffs for satifiability. In *Proceedings 12th Conference on Computational Complexity*, pages 52–60, 1997.

8 L. Fortnow and D. Van Melkebeek. Time-space tradeoffs for nondeterministic computation. In *Proceedings 15th Conference on Computational Complexity*, pages 2–13, 2000.

9 Lance Fortnow, Richard Lipton, Dieter Van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM (JACM)*, 52(6):835–865, 2005.

10 S. Jukna. A nondeterministic space-time tradeoff for linear codes. *Information Processing Letters*, 109(5):286–289, 2009.

11 Stasys Jukna. A note on read-k times branching programs. *Informatique théorique et applications*, 29(1):75–83, 1995.

12 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

13 Stasys P Jukna. The effect of null-chains on the complexity of contact schemes. In *Fundamentals of Computation Theory*, pages 246–256. Springer, 1989.

14 Matthias Krause, Christoph Meinel, and Stephan Waack. Separating the eraser turing machine classes le, nle, co-nle and pe. In *Mathematical Foundations of Computer Science 1988*, pages 405–413. Springer, 1988.

15 R. Lipton and A. Viglas. Time-space tradeoffs for sat. In *Proceedings 40th FOCS*, pages 459–464, 1999.

16 EA Okolnishnikova. On lower bounds for branching programs. *Siberian Advances in Mathematics*, 3(1):152–166, 1993.

17 Pavel Pudlak and Stanislav Zak. Space complexity of computations. *Preprint Univ. of Prague*, 1983.

18 Ryan Williams. Better time-space lower bounds for sat and related problems. In *Computational Complexity, 2005. Proceedings. Twentieth Annual IEEE Conference on*, pages 40–49. IEEE, 2005.

Improved Bounds on the Sign-Rank of AC^0*

Mark Bun[1] and Justin Thaler[2]

1 **Harvard University, Cambridge, MA, USA**
mbun@seas.harvard.edu
2 **Yahoo Labs, New York, NY, USA**
jthaler@fas.harvard.edu

Abstract

The *sign-rank* of a matrix A with entries in $\{-1,+1\}$ is the least rank of a real matrix B with $A_{ij} \cdot B_{ij} > 0$ for all i,j. Razborov and Sherstov (2008) gave the first exponential lower bounds on the sign-rank of a function in AC^0, answering an old question of Babai, Frankl, and Simon (1986). Specifically, they exhibited a matrix $A = [F(x,y)]_{x,y}$ for a specific function $F\colon \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ in AC^0, such that A has sign-rank $\exp(\Omega(n^{1/3}))$.

We prove a generalization of Razborov and Sherstov's result, yielding exponential sign-rank lower bounds for a non-trivial class of functions (that includes the function used by Razborov and Sherstov). As a corollary of our general result, we improve Razborov and Sherstov's lower bound on the sign-rank of AC^0 from $\exp(\Omega(n^{1/3}))$ to $\exp(\tilde{\Omega}(n^{2/5}))$. We also describe several applications to communication complexity, learning theory, and circuit complexity.

1998 ACM Subject Classification F.2.0 Analysis of Algorithms and Problem Complexity – General

Keywords and phrases Sign-rank, circuit complexity, communication complexity, constant-depth circuits

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.37

1 Introduction

The *sign-rank* of a matrix A with entries in $\{-1,+1\}$ is the least rank of a real matrix B with $A_{ij} \cdot B_{ij} > 0$ for all i,j. This fundamental matrix-theoretic complexity measure has diverse applications in theoretical computer science. For example:

- Upper bounds on sign-rank underly many state of the art learning algorithms, including the fastest known algorithms for PAC learning DNF and read-once formulas. Algorithms based on sign-rank are additionally robust to random classification noise, a property not satisfied by the handful of known PAC learning algorithms that cannot be captured in the sign-rank framework (all of which are based on Gaussian Elimination) [12].
- In communication complexity, sign-rank is known to characterize *unbounded error communication*. Introduced by Paturi and Simon [16] and captured by the communication complexity class $\mathbf{UPP}^{cc}$, this is a powerful communication model that lies at the frontier of our understanding. It is essentially the most powerful communication model against which we know how to prove lower bounds. In fact, the only known communication models that $\mathbf{UPP}^{cc}$ cannot efficiently simulate are the communication analogues of the polynomial hierarchy introduced by Babai, Frankl, and Simon [4]. We direct the interested reader to the recent paper of Göös et al. [11] for a detailed overview of communication complexity classes and their relationships.

* A full version of the paper is available at http://eccc.hpi-web.de/report/2016/075/.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 37; pp. 37:1–37:14


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

- In circuit complexity, sign-rank lower bounds on a matrix $A = [F(x,y)]_{x,y}$ imply lower bounds on the size of threshold-of-majority circuits computing F.

Despite the importance of these applications, our understanding of sign-rank remains rather limited, and it is possible to summarize relevant prior work in a single paragraph. Alon et al. [2] proved lower bounds on the sign-rank of random matrices. The first nontrivial lower bounds for explicit matrix families was obtained in a breakthrough work of Forster [8], who proved strong lower bounds on the sign-rank of Hadamard matrices, and more generally of any sign matrix with small spectral norm. Several subsequent works improved and generalized Forster's method [9, 10, 14, 3]. Nearly tight estimates of the sign-rank were obtained by Sherstov in [20] for all symmetric predicates, i.e., matrices of the form $[D(\sum_i x_i \vee y_i)]_{x,y}$ where $D : \{0,1,\ldots,n\} \to \{0,1\}$ is a given predicate and x, y range over $\{0,1\}^n$. Razborov and Sherstov [17] answered an old question of Babai, Frankl, and Simon [4] by giving the first exponential sign-rank lower bounds on a function in AC^0. Specifically, they gave a matrix $A = [F(x,y)]_{x,y}$ for a function $F\colon \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ in AC^0, such that A has sign-rank $\exp(\Omega(n^{1/3}))$.

Our work strengthens and generalizes the results of Razborov and Sherstov on the sign-rank of AC^0.

1.1 Our Results

The *threshold degree* of a function $h\colon \{-1,1\}^n \to \{-1,1\}$, denoted $\deg_\pm(h)$, is the least degree of a real polynomial that agrees in sign with h at all inputs. Minsky and Papert [15] famously showed that the threshold degree of the DNF formula $\mathrm{MP}_n(x) = \vee_{i=1}^{n^{1/3}} \wedge_{j=1}^{n^{2/3}} x_{ij}$ – now known as the Minsky-Papert DNF – is $\Omega(n^{1/3})$. This is the same function that Razborov and Sherstov used to prove their sign-rank lower bounds, and their analysis is highly tailored to the Minsky-Papert DNF. We generalize their result as follows.

For any $d > 0$, we identify a class $\mathcal{C}_d$ of functions $f : \{-1,1\}^k \to \{-1,1\}$ such that any $f \in \mathcal{C}_d$ can be transformed into a function $F : \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$, where $n = O(dk)$, for which $A = [F(x,y)]_{x,y}$ has sign rank $\exp(\Omega(d))$. Crucially, this transformation is simple in the sense that if f is computed by a polynomial-size circuit of depth t, then F is computed by a polynomial-size circuit of depth at most $t+1$ (and in some cases, F may be shallower).

In particular, the k-variate AND_k function is in $\mathcal{C}_d$ for some $d = \Omega(k^{1/2})$. Our transformation of AND_k into a function $F : \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ for $n = O(k^{3/2})$ recovers Razborov and Sherstov's function, with the same sign-rank bound of $\exp(\Omega(k^{1/2})) = \exp(\Omega(n^{1/3}))$. We also identify a k-variate AC^0 function that is in $\mathcal{C}_d$ for some $d = \tilde{\Omega}(k^{2/3})$, which in turn yields new sign-rank lower bounds for AC^0.

The precise definition of $\mathcal{C}_d$ is rather technical, so for expository purposes, we restrict ourselves to an informal statement of this result in this introduction. We define $\mathcal{C}_d$ formally in Section 2.2.

Informal description of the class $\mathcal{C}_d$. Our class $\mathcal{C}_d$ consists of all functions of the form $f : \{-1,1\}^k \to \{-1,1\}$, where f satisfies the following (informally stated) property: there exists a "small" set $S \subseteq f^{-1}(+1)$ such that f cannot be uniformly approximated to error $1/2$ by degree d polynomials, even under the promise that the input x is in $f^{-1}(-1) \cup S$. The precise definition of $\mathcal{C}_d$ is based on a *dual* (in the sense of linear programming duality) interpretation of this property.

Transforming functions in $\mathcal{C}_d$ to functions with high sign-rank. For $g : \{-1,1\}^m \to \{-1,1\}$ and $f : \{-1,1\}^k \to \{-1,1\}$, the notation $g \circ f = g(f,\ldots,f)$ denotes the function on $n = mk$ bits obtained by block-composing g with f. Let OR_m and AND_m denote the logical OR and AND functions on m bits, respectively. Let C be a sufficiently large universal constant. Given a function $f : \{-1,1\}^k \to \{-1,1\}$, let $F : \{-1,1\}^n \times \{-1,1\}^n \to \{-1,1\}$ be defined by

$$F = \mathrm{OR}_{2d} \circ f \circ \mathrm{AND}_C \circ \mathrm{OR}_2,$$

and hence $n = 2Cdk = O(dk)$.

▶ **Theorem 1** (Informal). *For any $f \in \mathcal{C}_d$, the matrix $A = [F(x,y)]_{x,y}$ has sign-rank* $\exp(\Omega(d))$.

Examples of functions in $\mathcal{C}_d$. We consider two prominent examples of functions in $\mathcal{C}_d$. As mentioned above, the first is the function $\mathrm{AND}_k : \{-1,1\}^k \to \{-1,1\}$, which we show is in $\mathcal{C}_d$ for $d = \Omega(k^{1/2})$. Hence, we recover a new proof of Razborov and Sherstov's lower bound.

▶ **Corollary 2.** *Let $\mathrm{MP}_n = \mathrm{OR}_{n^{1/3}} \circ \mathrm{AND}_{n^{2/3}}$ be the Minsky-Papert DNF. Then $A = [\mathrm{MP}_n(x \vee y)]_{x,y}$ has sign-rank* $\exp(\Omega(n^{1/3}))$.

Let $\mathrm{ED}_k \colon \{-1,1\}^k \to \{-1,1\}$ denote the well-known Element Distinctness function (defined in Section 2.6). As we will show, the function ED_k is in $\mathcal{C}_d$ for some $d = \tilde{\Omega}(k^{2/3})$. Hence, we obtain the following corollary, which improves Razborov and Sherstov's lower bound on the sign-rank of AC^0 from $\exp(\Omega(n^{1/3}))$ to $\exp(\tilde{\Omega}(n^{2/5}))$.

▶ **Corollary 3.** *Let $F_n^{\mathrm{ED}} = \mathrm{OR}_{n^{2/5}} \circ \mathrm{ED}_{n^{3/5}} \circ \mathrm{AND}_C \circ \mathrm{OR}_2$. Then $A = [F_n^{\mathrm{ED}}(x,y)]_{x,y}$ has sign-rank* $\exp(\tilde{\Omega}(n^{2/5}))$.

As discussed in Section 2.6, the function F_n^{ED} is computed by a depth-3 AC^0 circuit with logarithmic bottom fan-in.

1.2 Applications

We describe applications of Corollary 3 to communication complexity, learning theory, and circuit complexity in detail in the full version of this work. Here, we briefly describe these applications.

- Razborov and Sherstov's result yielded a function in the communication complexity class $\mathbf{PH}^{\mathbf{cc}}$ (the communication analog of the polynomial hierarchy) that requires unbounded error communication complexity $\Omega(n^{1/3})$. This was the first separation between the communication complexity classes $\mathbf{PH}^{\mathbf{cc}}$ and $\mathbf{UPP}^{\mathbf{cc}}$, answering a longstanding open problem of Babai, Frankl, and Simon [4]. We improve this separation, giving a function in the communication complexity class $\mathbf{PH}^{\mathbf{cc}}$ (indeed, in $\mathbf{\Sigma_2^{cc}}$) that requires unbounded error communication complexity $\tilde{\Omega}(n^{2/5})$.
- Razborov and Sherstov's result implied that learning algorithms in the sign-rank framework cannot PAC learn DNF formulae in time less than $\exp(O(n^{1/3}))$. This essentially matches the $\exp(\tilde{O}(n^{1/3}))$ runtime of the sign-rank based algorithm of Klivans and Servedio [13]. It is reasonable to ask whether the sign-rank framework can be used to learn depth-3 (or deeper) AC^0 circuits in the same $\exp(\tilde{O}(n^{1/3}))$ time bound. Our results rule this out, showing that sign-rank based learning algorithms require time $\exp(\tilde{\Omega}(n^{2/5}))$ to learn depth-3 AC^0 circuits, even when the bottom fan-in is $O(\log n)$.

- Razborov and Sherstov's result implied an exponential (specifically, $\exp(\Omega(n^{1/3}))$) lower bound on the size of threshold-of-majority circuits computing a function in AC^0. We improve their lower bound to $\exp\left(\tilde{\Omega}(n^{2/5})\right)$.

1.3 Our Techniques

It is well-known that the threshold degree of any function $h\colon \{-1,1\}^n \to \{-1,1\}$ is characterized by an (exponentially large) linear program. Using this formulation, if $\deg_{\pm}(h) = d$, then strong LP duality guarantees the existence of a dual solution μ that *witnesses* the fact that $\deg_{\pm}(h) \geq d$. Specifically, μ takes the form of a distribution on $\{-1,1\}^n$ such that h is uncorrelated under μ with all polynomials of degree at most d, i.e., $\sum_{x\in\{-1,1\}^n} \mu(x)\cdot h(x)\cdot p(x) = 0$ for all polynomials p of degree at most d. Razborov and Sherstov refer to μ as a *d-orthogonalizing distribution* for h (see Section 2.5 below for details).

In order to establish sign-rank lower bounds for the matrix $A = [(h \circ \mathrm{AND}_C)(x \vee y)]_{x,y}$, Razborov and Sherstov extended a lemma of Forster to show that it is enough to give an orthogonalizing distribution μ for h that additionally satisfies a *smoothness* property (cf. Theorem 18 in Section 4 for details). Specifically, a d-orthogonalizing distribution for h is said to be smooth if $\mu(x) = \exp(-O(d))$ for all but an $\exp(-\Omega(d))$ fraction of inputs $x \in \{-1,1\}^n$. Intuitively, this means that μ is smooth if it places "noticeable" mass on "almost all" inputs.

Razborov and Sherstov proved (non-constructively) that there exists a smooth d-orthogonalizing distribution for the Minsky-Papert DNF, for $d = n^{1/3}$. To generalize their result, for any $d > 0$ and any function $f \in \mathcal{C}_d$, we explicitly construct a smooth d-orthogonalizing distribution for the function $\mathrm{OR}_d \circ f$. Our construction combines new ideas with insights of Razborov and Sherstov, and ideas from prior works by the authors and Sherstov [7, 23] that constructed (non-smooth) orthogonalizing distributions for functions of the form $\mathrm{OR} \circ f$.

2 Preliminaries

2.1 Notation

We work with Boolean functions $f : \{-1,1\}^k \to \{-1,1\}$, where -1 corresponds to logical TRUE and $+1$ corresponds to logical FALSE. For $x \in \{-1,1\}^k$, let $|x| = \#\{i : x_i = -1\}$ denote the Hamming weight of x. Note that $|x|$ is computed by the linear function $|x| = \frac{k}{2} - \frac{1}{2}\sum_{i=1}^{k} x_i$.

2.2 Symmetrization

▶ **Definition 4.** Let $T : \{-1,1\}^k \to D$, where D is a finite subset of $\mathbb{R}^n$ for some $n \in \mathbb{N}$. The map T is *degree non-increasing* if for every polynomial $p : \{-1,1\}^k \to \mathbb{R}$, there exists a polynomial $q : D \to \mathbb{R}$ with $\deg q \leq \deg p$ such that

$$q(T(x)) = \mathop{\mathbb{E}}_{y \text{ s.t. } T(y)=T(x)} [p(y)]$$

for every $x \in \{-1,1\}^k$. We say that a degree non-increasing map T *symmetrizes* a function $f : \{-1,1\}^k \to \mathbb{R}$ if $f(x) = f(y)$ whenever $T(x) = T(y)$, and in this case we say that T is a *symmetrization* for f.

The canonical example of a degree non-increasing map is that which computes the Hamming weight.

▶ **Lemma 5** (Minsky and Papert [15]). *The map $T : \{-1,1\}^k \to \{0,1,\ldots,k\}$ defined by $T(x) = |x|$ is degree non-increasing. Hence, T is a symmetrization for any symmetric Boolean function.*

For any function $\psi\colon \{-1,1\}^k \to \mathbb{R}$, a symmetrization $T\colon \{-1,1\}^k \to D$ for ψ induces a symmetrized function $\widetilde{\psi}\colon D \to \mathbb{R}$ defined via $\widetilde{\psi}(z) := \mathbb{E}_{x \in T^{-1}(z)} \psi(x)$. (If $T^{-1}(z)$ is empty, then we define $\widetilde{\psi}(z) = 0$). It will also be convenient to define an "unnormalized" version $\hat{\psi}$ of $\widetilde{\psi}$, defined via $\hat{\psi}(z) := \sum_{x \in T^{-1}(z)} \psi(x)$. Observe that if μ is a distribution on $\{-1,1\}^k$, then $\hat{\mu}$ is a distribution on D.

Similarly, let $T\colon \{-1,1\}^k \to D$ be a degree non-increasing map. A function $\hat{\psi} : D \to \mathbb{R}$ naturally induces an un-symmetrized function $\psi : \{-1,1\}^k \to \mathbb{R}$ by setting $\psi(x) = \frac{1}{|T^{-1}(z)|}\hat{\psi}(z)$ where $z = T(x)$. That is, ψ spreads the mass of $\hat{\psi}(z)$ out evenly over points $x \in T^{-1}(z)$. Observe that, for any $\hat{\psi}$ and any degree non-increasing map T, the induced function ψ is symmetrized by T.

We will often pass back and forth between a function ψ on $\{-1,1\}^k$ and its symmetrized versions $\widetilde{\psi}$ and $\hat{\psi}$ on D, when the underlying symmetrization $T\colon \{-1,1\}^k \to D$ is understood.

2.3 Norms and Inner Products

For a function $\psi : \{-1,1\}^k \to \mathbb{R}$, define the ℓ_1 norm of ψ by $\|\psi\|_1 = \sum_{x \in \{-1,1\}^k} |\psi(x)|$. For functions $\psi, \varphi : \{-1,1\}^k \to \mathbb{R}$, denote the inner product $\langle \psi, \varphi \rangle = \sum_{x \in \{-1,1\}^k} \psi(x)\varphi(x)$. We say a function $\psi : \{-1,1\}^k \to \mathbb{R}$ has *pure high degree d* if $\langle \psi, p \rangle = 0$ for every polynomial $p : \{-1,1\}^k \to \mathbb{R}$ of degree less than d.

2.4 Dual Objects and the Class $\mathcal{C}_d$

Central to our work is the following definition of a "dual object." We show that whenever a Boolean function f can be associated with such a dual object, then f can be transformed into a function F such that $[F(x,y)]_{x,y}$ has high sign-rank.

▶ **Definition 6.** Let $f : \{-1,1\}^k \to \{-1,1\}$, and let $T : \{-1,1\}^k \to D$ be a (degree non-increasing) symmetrization for f. Let $\hat{\psi}\colon D \to \mathbb{R}$ be any function, and let ψ be the associated function on $\{-1,1\}^k$ induced by T. We say that $\hat{\psi}$ is a (d, ε, η)-*dual object* for f (with respect to T) if:

$$\langle \psi, f \rangle \geq \varepsilon \tag{1}$$

$$\|\psi\|_1 = 1 \tag{2}$$

$$\langle \psi, p \rangle = 0 \text{ for every polynomial } p : \{-1,1\}^k \to \mathbb{R} \text{ with } \deg p < d \tag{3}$$

$$f(x) = -1 \implies \psi(x) < 0 \tag{4}$$

$$\hat{\psi}(z_+) \geq \eta \text{ for some } z_+ \in D \text{ satisfying } \widetilde{f}(z_+) = 1 \tag{5}$$

Definition 6 is motivated by a recent line of work establishing lower bounds for polynomial approximations via linear programming duality. We direct the reader to [21, 5, 25, 7, 23, 22, 19, 18, 6] for thorough discussions of this technique and its applications to longstanding open questions in complexity theory. In short, one can use linear programming duality to show that the existence of a $(d, 2\eta, \eta)$-dual object for a function f is implied by the *non-existence* of a degree d polynomial that approximates f in a certain precise sense. In a bit more detail (and still simplifying a little), a $(d, 2\eta, \eta)$-dual object for f always exists if f cannot be uniformly approximated to error 2η by any degree d polynomial, even under the promise

that the input x is in $f^{-1}(-1) \cup T^{-1}(z_+)$. We will not use this *primal* interpretation of dual objects in our analysis, but we spell out this implication in the full version of this work for completeness and intuition.

Motivated by the study of uniform approximation of Boolean functions by polynomials, several works [24, 5, 6] have constructed dual objects directly. In particular, work of Špalek [24] and the authors [5] explicitly constructed an appropriate dual object for the AND function.

▶ **Lemma 7** (cf. [24, 5]). *Let $T : \{-1,1\}^k \to \{0,1,\ldots,k\}$ be the degree non-increasing map $T(x) = |x|$ that computes the Hamming weight. The function AND_k has a $(d, 1/2, 1/4)$-dual object with respect to T for $d = \Omega(\sqrt{k})$.*

We are now ready to define the class $\mathcal{C}_d$ of functions to which our techniques can be applied to yield sign-rank lower bounds.

▶ **Definition 8.** Let $f : \{-1,1\}^k \to \{-1,1\}$ be a Boolean function, and let $d > 0$. Then f is in the class $\mathcal{C}_d$ if there exists a symmetrization $T : \{-1,1\}^k \to D$ for f such that:

- there exists a $(d, 1/2, 1/4)$-dual object for f with respect to T, and
- the function f evaluates to TRUE (i.e. $f(x) = -1$) for at most a 2^{-d} fraction of inputs $x \in \{-1,1\}^k$.

2.5 Orthogonalizing Distributions

As indicated in Section 1.3, our analysis will make essential use of orthogonalizing distributions, which represent a dual formulation of the notion of threshold degree.

▶ **Definition 9.** A distribution $\mu : \{-1,1\}^n \to [0,1]$ is *d-orthogonalizing* for a function $h : \{-1,1\}^n \to \{-1,1\}$ if

$$\mathop{\mathbb{E}}_{x \sim \mu} [h(x)p(x)] = 0$$

for every polynomial $p : \{-1,1\}^n \to \mathbb{R}$ with $\deg p < d$. In other words, μ is d-orthogonalizing for h if the function $\mu(x)h(x)$ has pure high degree d.

2.6 The Element Distinctness Function

The Boolean function $\mathrm{ED}_k : \{-1,1\}^k \to \{-1,1\}$ is defined as follows. For simplicity, assume that $k = K \log_2 K$, where K is a power of 2. The function interprets its input x as blocks $x_1, \ldots, x_K$, where each $x_i \in \{-1,1\}^{\log_2 K}$. Each x_i is interpreted as the binary representation of $g_x(i)$ for a function $g_x : [K] \to [K]$. $\mathrm{ED}_k(x)$ is defined to equal -1 iff the function g_x is 1-to-1.

Observe that ED_k is symmetric with respect to permutations of the domain and range of g_x. That is, if $x, y \in \{-1,1\}^k$ are such that there exist permutations π, σ of $[K]$ with $g_x = \pi \circ g_y \circ \sigma$, then $\mathrm{ED}_k(x) = \mathrm{ED}_k(y)$.

In the full version of this work, we show that these symmetries imply the existence of a symmetrization T for ED_k and an associated dual object.

▶ **Lemma 10.** *There exists a symmetrization $T : \{-1,1\}^k \to [K]^K$ for the* Element Distinctness *function $\mathrm{ED}_k : \{-1,1\}^k \to \{-1,1\}$ such that ED_k has a $(d, 1/2, 1/4)$-dual object (with respect to the map T), for some $d = \Omega(K^{2/3}/\log K)$.*

▶ Remark. In fact, an explicit dual object for ED_k was constructed in our prior work [6]. In the full version of this work, we give an alternative *primal*-based proof of the existence of a dual object for ED_k. The proof is based on Aaronson and Shi's [1] influential lower bound of $\Omega(K^{2/3})$ on the *approximate degree*[1] of ED_k.

The Element Distinctness function is computed by a natural CNF formula:

$$\mathrm{ED}_k(x_1, \ldots, x_K) = \bigwedge_{r=1}^{K} \bigwedge_{i \neq j} ((x_i \neq r) \vee (x_j \neq r)).$$

Notice that the fan-in of each bottom OR gate is only $2K \leq 2\log_2 k$. Recall (cf. Corollary 3) that our aim is to prove a sign-rank lower bound for the function $F_n^{\mathrm{ED}}(x, y) = (\mathrm{OR}_{n^{2/5}} \circ \mathrm{ED}_{n^{3/5}} \circ \mathrm{AND}_C)(x \vee y)$. Using the CNF for Element Distinctness described above, the function F_n^{ED} is naturally computed by an AC^0 circuit Γ of depth 5, with an OR gate at the top. However, as we now explain, F_n^{ED} is actually computable by a *depth-3* AC^0 circuit with logarithmic bottom fan-in.

Number the layers of Γ from 1 to 5, with layer 1 corresponding to the OR gate at the top. Since each OR gate at layer 3 of Γ has fan-in $O(\log n)$ (and the gates at layers 4 and 5 have constant fan-in), the sub-circuits rooted at each gate at layer 3 of Γ are functions of only $O(\log n)$ bits of x. Since any function on $O(\log n)$ inputs can be computed by a $\mathrm{poly}(n)$ size CNF with logarithmic bottom fan-in, we can replace each sub-tree rooted at layer 3 of Γ with such a CNF, to obtain a circuit Γ' of depth 4, in which layers 2 and 3 of Γ' both consist of AND gates. Collapsing layers 3 and 4 into a single layer yields a polynomial size depth 3 circuit with logarithmic bottom fan-in that computes F_n^{ED}.

3 Constructing a Smooth Orthogonalizing Distribution

Sherstov [23] showed that whenever f has a $(d_1, 1/2, 0)$-dual object[2], the function $h_m := \mathrm{OR}_m \circ f$ has a d-orthogonalizing distribution for $d = \min\{m, d_1\}$. The goal of this section, and the main technical contribution of the paper, is to prove that whenever f has a $(d_1, 1/2, \eta)$-dual object for $\eta > 0$, the function h_m has a d-orthogonalizing distribution that places significant mass on each input $x \in h_m^{-1}(1)$. More precisely, we show:

▶ **Theorem 11.** *Suppose that $f : \{-1,1\}^k \to \{-1,1\}$ has a $(d_1, 1/2, \eta)$-dual object, and let $h_m = \mathrm{OR}_m \circ f$. Then there exists a d-orthogonalizing distribution $\mu : \{-1,1\}^{mk} \to [0,1]$ for h_m such that $\mu(x) \geq 4^{-(m+d+1)} \eta^{-m/2} 2^{-mk}$ for every $x \in h_m^{-1}(1)$, where $d = \min\{m/2, d_1\}$.*

Combining this theorem with Lemmas 7 and 10 yields smooth orthogonalizing distributions for the functions $\mathrm{OR}_{n^{1/3}} \circ \mathrm{AND}_{n^{2/3}}$ and $\mathrm{OR}_{n^{2/5}} \circ \mathrm{ED}_{n^{3/5}}$.

▶ **Corollary 12.** *There exists a d-orthogonalizing distribution μ for $h = \mathrm{OR}_{n^{1/3}} \circ \mathrm{AND}_{n^{2/3}}$ such that $\mu(x) \geq 2^{-O(d)} 2^{-n}$ on each $x \in h^{-1}(1)$, for some $d = \Omega(n^{1/3})$.*

▶ **Corollary 13.** *There exists a d-orthogonalizing distribution μ for $h = \mathrm{OR}_{n^{2/5}} \circ \mathrm{ED}_{n^{3/5}}$ such that $\mu(x) \geq 2^{-O(d)} 2^{-n}$ on each $x \in h^{-1}(1)$, for some $d = \tilde{\Omega}(n^{2/5})$.*

---

[1] The approximate degree of a Boolean function f is the minimum degree of a real polynomial for which $|p(x) - f(x)| \leq 1/3$ for all Boolean inputs x.

[2] The existence of a $(d_1, 1/2, 0)$-dual object for f is in fact a dual formulation of the property that f has *one-sided approximate degree* at least d_1. See [5, 23] for the definition of one-sided approximate degree.

3.1 Proof of Theorem 11

3.1.1 Notation

Let $f : \{-1,1\}^k \to \{-1,1\}$ be as in the statement of Theorem 11, and let $T : \{-1,1\}^k \to D$ be the symmetrization for f associated with the assumed $(d_1, 1/2, \eta)$-dual object for f. Define $T^m \colon \{-1,1\}^{mk} \to D^m$ by $T^m(x_1, \ldots, x_m) := (T(x_1), \ldots, T(x_m))$. Since T is degree non-increasing, it is easy to see that T^m is also degree non-increasing. Moreover, T^m is a symmetrization for h_m. The map T^m induces a symmetrized version $\widetilde{h}_m \colon D^M \to \mathbb{R}$ of h_m given by $\widetilde{h}_m = \mathrm{OR}_m \circ \widetilde{f}$.

3.1.2 Proof Outline

Let $Z^+ := \widetilde{h}_m^{-1}(1) \subseteq D^m$. At a high level, our proof will produce, for every $z \in Z^+$, a d-orthogonalizing distribution μ_z that is targeted to z, in the sense that

$$\hat{\mu}_z(z) \geq 2^{-O(m+d)} \cdot \eta^{-O(m)}.$$

Since the property of d-orthogonalization is preserved under averaging, the distribution $\mu = \frac{1}{|Z^+|} \sum_{z \in Z^+} \mu_z$ remains d-orthogonalizing, and places the required amount of probability mass on each input $x \in T^{-1}(Z^+) = h_m^{-1}(1)$. The goal therefore becomes to construct these targeted distributions μ_z. We do this in two stages.

Stage 1. In the first stage (see Claim 15 below), we construct distributions μ_z for every z belonging to a highly structured subset $G \subset Z^+$ that we now describe. Let $c \in \widetilde{f}^{-1}(1)$ denote the point on which the dual object $\hat{\psi}$ for f has $\hat{\psi}(c) \geq \eta$ (cf. Condition (5) within Definition 6). The set G consists of inputs in Z^+ for which $c \in D$ is repeated many times (specifically, at least $m/2$ times).

Stage 2. In the second stage (see Claim 16 below), we show that given the family of distributions $\{\mu_z : z \in G\}$ constructed in Stage 1, we can construct appropriate distributions μ_z for z belonging to the entire set Z^+.

Both stages can be viewed as generalized dual counterparts to analogous statements in the work of Razborov and Sherstov (cf. [17, Lemma 3.4] and [17, Theorem 3.6] respectively). Taking a dual perspective allows us to identify general properties (Definition 6) of a dual object for f that enable the construction of a smooth orthogonalizing distribution. This results in a much more general and modular framework for proving the existence of these distributions. Our framework also has the advantage of constructing smooth orthogonalizing distributions explicitly.

3.1.3 Proof Details

We begin with a relatively simple lemma that shows that the function $\mathrm{OR}_{m/2} \circ f$ has a d-orthogonalizing distribution μ such that $\hat{\mu}$ places a lot of probability mass on a particular highly structured input, where $d = \min\{d_1, m/2\}$. This distribution is an important building block in the proof of Claim 15 below.

▶ **Lemma 14.** *Let $\ell = m/2$, and let f, T, and T^ℓ be as above. Consider the function $h_\ell : \{-1,1\}^{k\ell} \to \{-1,1\}$ defined by $h_\ell(x_1, \ldots, x_\ell) = \mathrm{OR}_\ell(f(x_1), \ldots, f(x_\ell))$. There exists a*

function $\psi : \{-1,1\}^{k\ell} \to [0,1]$ *symmetrized by* T^ℓ *with the following properties.*

$$\psi \text{ agrees in sign with } h_\ell. \text{ That is, } \psi(x) \cdot h_\ell(x) \geq 0 \text{ for all } x \in \{-1,1\}^{k\ell} \tag{6}$$

$$\|\psi\|_1 = 1 \tag{7}$$

$$\psi \text{ has pure high degree at least } d = \min\{\ell, d_1\} \tag{8}$$

$$\text{There exists a } c \in D \text{ such that } \widetilde{f}(c) = 1 \text{ and } \hat{\psi}(\underbrace{c, \ldots, c}_{\ell \text{ times}}) \geq \eta^{-\ell}/2 \tag{9}$$

We remark that Conditions (6)–(8) are equivalent to requiring that $\mu := \psi \cdot h_\ell$ is a d-orthogonalizing distribution for h_ℓ, where $d = \min\{\ell, d_1\}$.

Proof Sketch. Sherstov [23] showed that when the function f has a $(d_1, 1/2, 0)$-dual witness, then there is a function ψ satisfying Conditions (6)–(8). In the full version of this work, we show that if f additionally has a $(d_1, 1/2, \eta)$-dual witness with $\eta > 0$, then Sherstov's construction yields a function $\hat{\psi}$ that also satisfies Condition (9). ◀

To complete Stage 1 of our proof, we show that for every input $w \in D^m$ that is close in Hamming distance to the special point $(\underbrace{c, \ldots, c}_{m \text{ times}})$, there is an orthogonalizing distribution for h_m that places substantial weight on w. Let $G \subset Z^+ = \widetilde{h}_m^{-1}(1)$ denote the set of inputs in $\widetilde{h}_m^{-1}(1)$ that take the value c on at least $m/2$ coordinates. That is,

$$G = \{z \in Z^+ : \exists i_1, \ldots, i_{m/2} \text{ s.t. } z_{i_1} = \cdots = z_{i_{m/2}} = c\}.$$

▶ **Claim 15.** *Let* G *be as above. For every* $w = (w_1, \ldots, w_m) \in G$*, there exists a* d*-orthogonalizing distribution* $\nu_w : \{-1,1\}^{km} \to [0,1]$ *for* h_m *such that* ν_w *is symmetrized by* T^m *and* $\hat{\nu}_w(w) \geq \eta^{m/2}/2$*.*

Proof Sketch. Let $I = \{i_1, \ldots, i_{m/2}\}$ denote the first $m/2$ coordinates on which w takes the value c. Define the distribution $\hat{\nu}_w$ by

$$\hat{\nu}_w(z) = \begin{cases} |\hat{\psi}(z_{i_1}, \ldots, z_{i_{m/2}})| & \text{if } z_i = w_i \text{ for all } i \notin I \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{\psi}$ is the function from Lemma 14 for $\ell = m/2$. It is immediate from the definition that $\hat{\nu}_w$ is a distribution on D^m, and hence ν_w is a distribution on $\{-1,1\}^{km}$. Moreover, $\hat{\nu}_w(w) \geq \eta^{m/2}/2$. The fact that ν_w is d-orthogonalizing follows from the fact that ψ has pure high degree at least d. This calculation appears in the full version of this work. ◀

We now proceed to Stage 2 of our proof, in which we use the distributions constructed in Claim 15 to give orthogonalizing distributions that place significant weight on *any* input $x \in h_m^{-1}(1)$.

▶ **Claim 16.** *Let* G *be as before, and suppose that for every* $w \in G$ *there exists a* d*-orthogonalizing distribution* $\nu_w : \{-1,1\}^{km} \to [0,1]$ *for* h_m *that is symmetrized by* T^m*, and satisfies* $\hat{\nu}_w(w) \geq \delta$*. Then for every* $v \in (Z^+ \setminus G)$*, there exists a* d*-orthogonalizing distribution* ρ_v *that is symmetrized by* T^m*, and* $\hat{\rho}_v(v) \geq \delta/4^{m+d}$*.*

The main technical ingredient in the proof of Claim 16 is the construction of a function $\varphi : \{0,1\}^m \to \mathbb{R}$ of pure high degree d for which $\varphi(1^m)$ is "large". This can be viewed as a dual formulation of a bound on the growth of low-degree polynomials. The construction of φ appears as part of the proof of such a bound in [17].

▶ Remark. We choose to state Lemma 17 below for a function $\varphi : \{0,1\}^m \to \mathbb{R}$, rather than applying our usual convention of working with functions over $\{-1,1\}^m$, because it makes various statements in the proof of Claim 16 cleaner. To clarify the terminology below, we say a function $\varphi : \{0,1\}^m \to \mathbb{R}$ has *pure high degree* d if $\sum_{x\in\{0,1\}^m} \varphi(x) \cdot p(x) = 0$ for every polynomial $p : \{0,1\}^m \to \mathbb{R}$ of degree less than d. The Hamming weight function $|\cdot| : \{0,1\}^m \to [m]$ counts the number of 1's in its input, i.e. $|s| = s_1 + s_2 + \cdots + s_m$.

▶ **Lemma 17** (cf. [17, Proof of Lemma 3.2]). *Let d be an integer with $0 \le d \le m-1$. Then there exists a function $\varphi : \{0,1\}^m \to \mathbb{R}$ such that*

$$\varphi(1^m) = 1 \tag{10}$$

$$\varphi(x) = 0 \text{ for all } d \le |x| < m \tag{11}$$

$$\varphi \text{ has pure high degree at least } d \tag{12}$$

$$\sum_{|x|\le d} |\varphi(x)| \le 2^d \binom{m}{d} \tag{13}$$

Proof of Claim 16. Fix $v \in (Z^+ \setminus G)$. Define an auxiliary function $\hat{\varphi}_v : D^m \to [0,1]$ as follows. For any $z = (z_1, \ldots, z_m)$, let

$$\hat{\varphi}_v(z) := \sum_{\substack{s\in\{0,1\}^m \text{ s.t.} \\ \forall i\ z_i = s_i c + (1-s_i)v_i}} \varphi(s),$$

where φ is as in Lemma 17, with d set as in the conclusion of Claim 15 (observe that if there is some z_i such that $z_i \neq c$ and $z_i \neq v_i$, then $\hat{\varphi}_v(z) = 0$).

Letting φ_v denote the function on $\{-1,1\}^{km}$ induced from $\hat{\varphi}_v$ by T^m, we record some properties of φ_v and $\hat{\varphi}_v$.

$$\hat{\varphi}_v(v) = \varphi(1^m) = 1 \tag{14}$$

$$\operatorname{supp} \hat{\varphi}_v \subset G \cup \{v\} \tag{15}$$

$$\varphi_v \text{ has pure high degree at least } d \tag{16}$$

$$\|\varphi_v\|_1 \le 2^d \binom{m}{d} + 1 \tag{17}$$

$$\hat{\varphi}_v \text{ is supported on at most } \frac{1}{2}2^m + 1 \text{ points in } D^m \tag{18}$$

Verifying Conditions (14)–(18). Conditions (14), (15), and (18) are immediate from the definition of $\hat{\varphi}_v$, combined with Conditions (10) and (11) of Lemma 17. For Condition (16), it is enough to show that if $p_1, \ldots, p_m$ are polynomials over $\{-1,1\}^k$ whose degrees sum to less than d, then $\sum_{x=(x_1,\ldots,x_m)\in\{-1,1\}^{km}} \varphi_v(x) \prod_{i=1}^m p_i(x_i) = 0$. To establish this, let $q_1, \ldots, q_m \colon D \to \mathbb{R}$ denote polynomials satisfying $\deg(q_i) \le \deg(p_i)$, and such that for all i and all z_i in the image of T, $q_i(z_i) := \mathbb{E}_{x\in T^{-1}(z_i)}[p_i(z_i)]$. Such polynomials are guaranteed

to exist, since T is degree non-increasing. Then:

$$\begin{aligned}\sum_{x=(x_1,\ldots,x_m)\in\{-1,1\}^{km}} \varphi_v(x)\prod_{i=1}^m p_i(x_i) &= \sum_{z=(z_1,\ldots,z_m)\in D^m} \hat{\varphi}_v(z)\prod_{i=1}^m q_i(z_i)\\ &= \sum_{z=(z_1,\ldots,z_m)\in D^m}\left(\sum_{\substack{s\in\{0,1\}^m \text{ s.t.}\\ \forall i\ z_i=s_ic+(1-s_i)v_i}}\varphi(s)\right)\prod_{i=1}^m q_i(z_i)\\ &= \sum_{s\in\{0,1\}^m}\varphi(s)\prod_{i=1}^m q_i(s_ic+(1-s_i)v_i)\\ &= 0,\end{aligned}$$

To see that the final equality holds, recall that that degrees of the polynomials q_i sum to strictly less than d. Hence, $p(s_1,\ldots,s_m) := \prod_{i=1}^m q_i(s_ic+(1-s_i)v_i)$ is a polynomial of degree strictly less than d over $\{-1,1\}^m$. The final equality then follows from the fact that φ has pure high degree at least d.

To establish Condition (17), we check that

$$\sum_{z\in D^m, z\neq v}|\hat{\varphi}_v(z)| \le \sum_{s\in\{0,1\}^m, s\neq 1^m}|\varphi(s)| \le 2^d\binom{m}{d},$$

where the final inequality holds by Condition (13).

Construction and analysis of ρ_v. Up to normalization, the function $\varphi_v\cdot h_m$ has all of the properties that we need to establish Claim 16, except that there are locations where it may be negative. We obtain our desired orthogonalizing distribution ρ_v by adding correction terms to $\hat{\varphi}_v$ in the locations where $\hat{\varphi}_v$ may disagree with $\widetilde{h}_m$ in sign. These correction terms are derived from the distributions $\hat{\nu}_w$ whose existence are hypothesized in the statement of Claim 16. We start by defining

$$\hat{P}_v(z) = \frac{\delta}{2^d\binom{m}{d}+1}\widetilde{h}_m(z)\hat{\varphi}_v(z) + \sum_{w\in(\operatorname{supp}\hat{\varphi}_v\setminus\{v\})}\hat{\nu}_w(z). \tag{19}$$

Observe that each w appearing in the sum on the right hand side of Eq. (19) is in the set G, owing to Condition (15). This guarantees that each term $\hat{\nu}_w$ in the sum is well-defined.

Now we check that $\hat{P}_v$ is nonnegative. Since each term $\hat{\nu}_w$ appearing in the sum on the right hand side of Eq. (19) is a distribution (and hence non-negative), it suffices to check that $\hat{P}_v(z)\ge 0$ for each point $z\in\operatorname{supp}\hat{\varphi}_v$. On each such point with $z\neq v$, Condition (17) guarantees that $\frac{\delta}{2^d\binom{m}{d}+1}\widetilde{h}_m(z)\hat{\varphi}_v(z)\ge -\delta$. Moreover, the contribution of the sum is at least $\hat{\nu}_z(z)\ge\delta$ by hypothesis. Hence, $\hat{P}_v$ is a non-negative function.

Next, we check that normalizing $\hat{P}_v$ yields a distribution $\hat{\rho}_v := \hat{P}_v/\|P_v\|_1$ for which $\hat{\rho}_v(v)\ge\delta/4^{m+d}$ as required. By construction, $\hat{P}_v(v) = \delta/\left(2^d\binom{m}{d}+1\right)$. Moreover, Conditions (14), (17), and (18) together show that $\|\hat{P}_v\|_1\le\delta+\frac{1}{2}2^m\le 2^m$. Hence, $\hat{P}_v(v)\ge\delta/\left(2^m\cdot\left(2^d\binom{m}{d}+1\right)\right)\ge\delta/\left(2^{m+d+1}\binom{m}{d}\right)\ge\delta/2^{2m+d+1}\ge\delta/4^{m+d}$.

Finally, we must check that $\rho_v = P_v/\|P_v\|_1$ is d-orthogonalizing for h_m. To see this, observe that $P_v\cdot h_m$ is a linear combination of the functions φ_v and $\nu_w\cdot h_m$ for $w\in(\operatorname{supp}\hat{\varphi}_v\setminus\{v\})$. Moreover, each of these functions has pure high degree at least d (φ_v does so by Condition (16), while $\nu_w\cdot h_m$ does by the fact that ν_w is d-orthogonalizing for h_m). By

linearity, it follows that $P_v \cdot h_m$ has pure high degree at least d, so ρ_v is d-orthogonalizing for h_m as desired.

This completes the proof of Claim 16. ◀

We are now ready to combine the claims above to prove Theorem 11.

Proof of Theorem 11. By Claim 15, for every $w \in G$ there exists a d-orthogonalizing distribution $\nu_w : \{-1,1\}^{km} \to [0,1]$ for h_m that is symmetrized by T^m, with $\hat{\nu}_w(w) \geq \eta^{m/2}/2$. Thus, by Claim 16, it is also true that for every $v \in (Z^+ \setminus G)$, there is a d-orthogonalizing distribution $\rho_v : \{-1,1\}^{km} \to [0,1]$ that is symmetrized by T^m, with $\hat{\rho}_v(v) \geq \eta^{m/2} 4^{-(m+d+1)}$. Now consider the distribution

$$\hat{\mu}(z) = \frac{1}{|Z^+|} \left(\sum_{w \in G} \hat{\nu}_w(z) + \sum_{v \in (Z^+ \setminus G)} \hat{\rho}_v(z) \right).$$

The (un-symmetrized) distribution $\mu : (\{-1,1\}^k)^m \to [0,1]$ satisfies $\mu(x) \geq \eta^{m/2} 4^{-(m+d+1)} 2^{-km}$ for every point $x \in T^{-1}(Z^+) = h_m^{-1}(1)$. Moreover, μ remains d-orthogonalizing for h_m, as it is a sum of d-orthogonalizing distributions for h_m. ◀

4 Sign Rank Lower Bounds for AC^0

We now use the machinery developed by Razborov and Sherstov to translate our construction of a smooth orthogonalizing distribution into a sign-rank lower bound.

▶ **Theorem 18** (Implicit in [17, Theorem 1.1])**.** *Let $h\colon \{-1,1\}^n \to \{-1,1\}$ be a Boolean function, and suppose there exists a d-orthogonalizing distribution μ for h such that $\mu(x) \geq 2^{-cd} 2^{-n}$ for all but a 2^{-cd} fraction of inputs $x \in \{-1,1\}^n$. Then there exists a constant C (depending only on c) such that if $F(x,y) := h(\ldots, \wedge_{j=1}^{C}(x_{ij} \vee y_{ij}), \ldots)$, then the matrix $[F(x,y)]_{x,y}$ has sign-rank $\exp(\Omega(d))$.*

Combining Theorem 18 with Theorem 11 yields the main result of this work.

▶ **Theorem 19.** *Let $f : \{-1,1\}^k \to \{-1,1\}$ be a Boolean function in the class $\mathcal{C}_d$. Let $F : \{-1,1\}^n \to \{-1,1\}^n$ be defined by*

$$F = \mathrm{OR}_{2d} \circ f \circ \mathrm{AND}_C \circ \mathrm{OR}_2,$$

where C is the universal constant of Theorem 18 (and hence $n = O(dk)$). The sign-rank of the matrix $[F(x,y)]_{x,y}$ is $\exp(\Omega(d))$.

Proof. Let $h_{2d} : \{-1,1\}^{2dk} \to \{-1,1\}$ denote the function $h_{2d} = \mathrm{OR}_{2d} \circ f$. By Theorem 11, there exists a d-orthogonalizing distribution μ for h_{2d} such that $\mu(x) \geq 2^{-9d} 2^{-2dk}$ for every $x \in h_{2d}^{-1}(1)$. Since $f \in \mathcal{C}_d$, we have by a union bound that $h_{2d}^{-1}(1)$ contains all but a $(2d) \cdot 2^{-d} \leq 2^{-d/2}$ fraction of the points in $\{-1,1\}^{2dk}$. Thus, by Theorem 18, there is a universal constant C for which $[F(x,y)]_{x,y}$ has sign-rank $\exp(\Omega(d))$. ◀

▶ **Corollary 20.** *Let $\mathrm{MP}_n = \mathrm{OR}_{n^{1/3}} \circ \mathrm{AND}_{n^{2/3}}$ be the Minsky-Papert DNF. Then $[\mathrm{MP}_n(x \vee y)]_{x,y}$ has sign-rank $\exp(\Omega(n^{1/3}))$*

Proof. The function AND_k evaluates to TRUE on exactly 1 out of 2^k inputs. Hence, by Lemma 7, we have $\mathrm{AND}_k \in \mathcal{C}_d$ for $d = \Omega(k^{1/2})$. Let $F = \mathrm{MP}_n \circ \mathrm{AND}_C \circ \mathrm{OR}_2$. Applying Theorem 19 implies that the sign-rank of $[F(x,y)]_{x,y} = \exp(\Omega(n^{1/3}))$. Merging the two adjacent layers of AND gates in the natural circuit computing F yields the desired result. ◀

▶ **Corollary 21.** *Let* $F_n^{\mathrm{ED}} = \mathrm{OR}_{n^{2/5}} \circ \mathrm{ED}_{n^{3/5}} \circ \mathrm{AND}_C$. *Then* $[F_n^{\mathrm{ED}}(x \vee y)]_{x,y}$ *has sign-rank* $\exp(\tilde{\Omega}(n^{2/5}))$

Proof. Assume for simplicity that $k = K \log K$. The function ED_k evaluates to TRUE on exactly $K!$ inputs, which is an $\exp(-O(K))$ fraction of the $2^k = K^K$ total inputs. Hence, by Lemma 10, we have $\mathrm{ED}_k \in \mathcal{C}_d$ for $d = \Omega(K^{2/3}/\log K)$. The result follows by applying Theorem 19. ◀

References

1. Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004.
2. Noga Alon, Peter Frankl, and Vojtech Rödl. Geometrical realization of set systems and probabilistic communication complexity. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 277–280, 1985.
3. Noga Alon, Shay Moran, and Amir Yehudayoff. Sign rank, VC dimension and spectral gaps. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:135, 2014.
4. László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 337–347, 1986.
5. Mark Bun and Justin Thaler. Dual lower bounds for approximate degree and Markov-Bernstein inequalities. In *ICALP (1)*, pages 303–314, 2013.
6. Mark Bun and Justin Thaler. Dual polynomials for collision and element distinctness. *CoRR*, abs/1503.07261, 2015. URL: `http://arxiv.org/abs/1503.07261`.
7. Mark Bun and Justin Thaler. Hardness amplification and the approximate degree of constant-depth circuits. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 268–280, 2015.
8. Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *J. Comput. Syst. Sci.*, 65(4):612–625, 2002.
9. Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans-Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science, 21st Conference, Bangalore, India, December 13-15, 2001, Proceedings*, pages 171–182, 2001.
10. Jürgen Forster and Hans-Ulrich Simon. On the smallest possible dimension and the largest possible margin of linear arrangements representing given concept classes uniform distribution. In *Algorithmic Learning Theory, 13th International Conference, ALT 2002, Lübeck, Germany, November 24-26, 2002, Proceedings*, pages 128–138, 2002.
11. Mika Göös, Toniann Pitassi, and Thomas Watson. The landscape of communication complexity classes. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:49, 2015. URL: `http://eccc.hpi-web.de/report/2015/049`.
12. Lisa Hellerstein and Rocco A. Servedio. On PAC learning algorithms for rich boolean function classes. *Theor. Comput. Sci.*, 384(1):66–76, 2007.
13. Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *J. Comput. Syst. Sci.*, 68(2):303–318, 2004.
14. Nati Linial, Shahar Mendelson, Gideon Schechtman, and Adi Shraibman. Complexity measures of sign matrices. *Combinatorica*, 27(4):439–463, 2007.
15. Marvin Minsky and Seymour Papert. *Perceptrons – an introduction to computational geometry*. MIT Press, 1969.

16 Ramamohan Paturi and Janos Simon. Probabilistic communication complexity. *J. Comput. Syst. Sci.*, 33(1):106–123, 1986.

17 Alexander A. Razborov and Alexander A. Sherstov. The sign-rank of AC^0. *SIAM J. Comput.*, 39(5):1833–1855, 2010.

18 A. A. Sherstov. The power of asymmetry in constant-depth circuits. In *Foundations of Computer Science*, 2015.

19 Alexander A. Sherstov. Communication lower bounds using dual polynomials. *Bulletin of the EATCS*, 95:59–93, 2008.

20 Alexander A. Sherstov. The unbounded-error communication complexity of symmetric functions. *Combinatorica*, 31(5):583–614, 2011.

21 Alexander A. Sherstov. Approximating the AND-OR tree. *Theory of Computing*, 9(20):653–663, 2013.

22 Alexander A. Sherstov. The intersection of two halfspaces has high threshold degree. *SIAM J. Comput.*, 42(6):2329–2374, 2013.

23 Alexander A. Sherstov. Breaking the Minsky-Papert barrier for constant-depth circuits. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 223–232, 2014.

24 Robert Spalek. A dual polynomial for OR. *CoRR*, abs/0803.4516, 2008. URL: `http://arxiv.org/abs/0803.4516`.

25 Justin Thaler. Lower bounds for the approximate degree of block-composed functions. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:150, 2014. URL: `http://eccc.hpi-web.de/report/2014/150`.

On the Sensitivity Conjecture

Avishay Tal*

Institute for Advanced Study, Princeton, USA
avishay.tal@gmail.com

Abstract

The sensitivity of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is the maximal number of neighbors a point in the Boolean hypercube has with different f-value. Roughly speaking, the block sensitivity allows to flip a set of bits (called a block) rather than just one bit, in order to change the value of f. The sensitivity conjecture, posed by Nisan and Szegedy (CC, 1994), states that the block sensitivity, $bs(f)$, is at most polynomial in the sensitivity, $s(f)$, for any Boolean function f. A positive answer to the conjecture will have many consequences, as the block sensitivity is polynomially related to many other complexity measures such as the certificate complexity, the decision tree complexity and the degree. The conjecture is far from being understood, as there is an exponential gap between the known upper and lower bounds relating $bs(f)$ and $s(f)$.

We continue a line of work started by Kenyon and Kutin (Inf. Comput., 2004), studying the ℓ-block sensitivity, $bs_\ell(f)$, where ℓ bounds the size of sensitive blocks. While for $bs_2(f)$ the picture is well understood with almost matching upper and lower bounds, for $bs_3(f)$ it is not. We show that any development in understanding $bs_3(f)$ in terms of $s(f)$ will have great implications on the original question. Namely, we show that either $bs(f)$ is at most sub-exponential in $s(f)$ (which improves the state of the art upper bounds) or that $bs_3(f) \geq s(f)^{3-\varepsilon}$ for some Boolean functions (which improves the state of the art separations).

We generalize the question of $bs(f)$ versus $s(f)$ to bounded functions $f : \{0,1\}^n \to [0,1]$ and show an analog result to that of Kenyon and Kutin: $bs_\ell(f) = O(s(f))^\ell$. Surprisingly, in this case, the bounds are close to being tight. In particular, we construct a bounded function $f : \{0,1\}^n \to [0,1]$ with $bs(f) \geq n/\log n$ and $s(f) = O(\log n)$, a clear counterexample to the sensitivity conjecture for bounded functions.

Finally, we give a new super-quadratic separation between sensitivity and decision tree complexity by constructing Boolean functions with $\mathrm{DT}(f) \geq s(f)^{2.115}$. Prior to this work, only quadratic separations, $\mathrm{DT}(f) = s(f)^2$, were known.

1998 ACM Subject Classification F.1.3 Computation by Abstract Devices – Complexity Measures and Classes

Keywords and phrases sensitivity conjecture, decision tree, block sensitivity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.38

1 Introduction

A long-standing open problem in complexity and combinatorics asks what is the relationship between two complexity measures of Boolean functions: the sensitivity and block-sensitivity. We first recall the definition of the two complexity measures.

* Research supported by the Simons Foundation and by the National Science Foundation grant No. CCF-1412958. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 38; pp. 38:1–38:13

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

▶ **Definition 1.** Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and $x \in \{0,1\}^n$ be a point. The sensitivity of f at x is the number of neighbors y of x in the Hamming cube such that $f(y) \neq f(x)$, i.e., $s(f,x) \triangleq |\{i \in [n] : f(x) \neq f(x \oplus e_i)\}|$.[1] The (maximal) sensitivity of f is defined as $s(f) \triangleq \max_{x \in \{0,1\}^n} s(f,x)$.

▶ **Definition 2.** Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and $x \in \{0,1\}^n$ be a point. For a block $B \subseteq [n]$, denote by $\mathbb{1}_B \in \{0,1\}^n$ its characteristic vector, i.e., $(\mathbb{1}_B)_i = 1$ iff $i \in B$. We say that a block B is sensitive for f on x if $f(x) \neq f(x \oplus \mathbb{1}_B)$. The block-sensitivity of f at x $x \in \{0,1\}^n$ is the maximal number of disjoint sensitive blocks for f at x, i.e.,

$$bs(f,x) = \max\{r : \exists \text{ disjoint } B_1, B_2, \ldots, B_r \subseteq [n] \ , f(x) \neq f(x \oplus \mathbb{1}_{B_i})\} \ .$$

The (maximal) block-sensitivity of f is defined as $bs(f) \triangleq \max_{x \in \{0,1\}^n} bs(f,x)$.

For shorthand, we will denote $(x \oplus e_i)$ and $(x \oplus \mathbb{1}_B)$ by $(x + e_i)$ and $(x + B)$ respectively. By definition, the block-sensitivity is at least the sensitivity by considering only blocks of size 1. The sensitivity conjecture, posed by Nisan and Szegedy [14], asks if a relation in the other direction holds as well.

▶ **Conjecture 3** (The Sensitivity Conjecture). $\exists d \ \forall f : \ bs(f) \leq s(f)^d$.

A stronger variant of the conjecture states that d can be taken to be 2. Despite much work on the problem [13, 14, 15, 12, 8, 20, 4, 11, 6, 1, 2, 5, 3, 9, 17, 10] there is still an exponential gap between the best known separations and the best known relations connecting the two complexity measures.

Known Separations. An interesting example due to Rubinstein [15] shows a quadratic separation between the two measures: $bs(f) = \frac{1}{2} \cdot s(f)^2$. This example was improved by [20] and then by [4] to $bs(f) = \frac{2}{3} \cdot s(f)^2 \cdot (1 - o(1))$ which is current state of the art.

Known Relations. Simon [16] proved (implicitly) that $bs(f)$ is at most $4^{s(f)} \cdot s(f)$. The upper bound was improved by Kenyon and Kutin [12] who showed that $bs(f) \leq O(e^{s(f)} \cdot \sqrt{s(f)})$. Recently, Ambainis et al. [1] improved this bound to $bs(f) \leq 2^{s(f)-1} \cdot s(f)$. Even more recently, Ambainis et al. [3] improved this bound slightly to $bs(f) \leq 2^{s(f)-1} \cdot (s(f) - 1/3)$.

To sum up, while the best known upper bound on the block-sensitivity in terms of sensitivity is exponential, the best known lower bound is quadratic. Indeed, we seem far from understanding the right relation between the two complexity measures.

1.1 ℓ-block sensitivity

All mentioned examples that exhibit quadratic separations between the sensitivity and block sensitivity ([15, 20, 4]) have the property that the maximal block sensitivity is achieved on blocks of size at most 2. For this special case, Kenyon and Kutin [12] showed that the block sensitivity is at most $2 \cdot s(f)^2$. Hence, these examples are essentially tight for this subcase.

Kenyon and Kutin introduced the notion of ℓ-block sensitivity (denoted $bs_\ell(f)$): the maximal number of disjoint sensitive blocks where each block is of size at most ℓ. Note that without loss of generality we may consider only sensitive blocks that are minimal with respect to set-inclusion (since otherwise we could of picked smaller blocks that are still disjoint). A

[1] e_i is the vector whose i-th entry equals 1 and all other entries equal 0.

well-known fact (cf. [7, Lemma 3]) asserts that any minimal sensitive block for f is of size at most $s(f)$, thus $bs(f) = bs_{s(f)}(f)$. Kenyon and Kutin proved the following inequalities relating the ℓ-block sensitivity of different ℓ-s:

$$bs_\ell(f) \leq \frac{4}{\ell} \cdot s(f) \cdot bs_{\ell-1}(f) \tag{1}$$

$$bs_\ell(f) \leq \frac{e}{(\ell-1)!} \cdot s(f)^\ell \tag{2}$$

for all $2 \leq \ell \leq s(f)$. Plugging $\ell = s(f)$ gives the aforementioned bound $bs(f) \leq O(e^{s(f)} \cdot \sqrt{s(f)})$.

1.2 Our Results

1. In the full version [19], we refine the argument of Kenyon and Kutin giving a better upper bound on the ℓ-block sensitivity in terms of $(\ell-1)$-block sensitivity. We show that

$$bs_\ell(f) \leq \frac{e}{\ell} \cdot s(f) \cdot bs_{\ell-1}(f) \tag{3}$$

   improving the bound in Eq. (1). On the other hand, Kenyon and Kutin gave examples with $bs_\ell(f) \geq \frac{1}{\ell} \cdot s(f) \cdot bs_{\ell-1}(f)$. Hence, Eq. (3) (and in fact, also Eq. (1)) is tight up to a constant. Interestingly, our analysis uses (a very simple) ordinary differential equation.
2. In Section 2, we put focus on understanding $bs_3(f)$ in terms of the sensitivity. We show that an upper bound of the form $bs_3(f) \leq s(f)^{3-\varepsilon}$ for some constant ε implies a sub-exponential upper bound for the sensitivity conjecture: $\forall f : bs(f) \leq 2^{s(f)^{1-\delta}}$, for $\delta > 0$. On the other hand, the best known separation (i.e., the aforementioned example by [4]) gives examples with $bs_3(f) \geq bs_2(f) \geq \Omega(s(f)^2)$. Thus, improving either the upper or lower bound for $bs_3(f)$ in terms of $s(f)$ will imply a breakthrough in our understanding of the sensitivity conjecture.
3. In Section 3, we consider an extension of the sensitivity conjecture to bounded functions $f : \{0,1\}^n \to [0,1]$. We show that while Kenyon and Kutin's approach works in this model, it is almost tight, i.e., we give functions for which $bs_\ell(f) = \Omega((s(f)/\ell)^\ell)$. In particular, we give a function with sensitivity $O(\log n)$ and block sensitivity $\Omega(n/\log n)$ – a clear counterexample for the sensitivity conjecture in this model.
4. In Section 4, we find better-than-quadratic separations between the sensitivity and the decision tree complexity. We construct functions based on minterm cyclic functions (as coined by Chakraborty [8]), that were found using computer search. In particular, we give an infinite family of functions $\{f_n\}_{n \in I}$ with $\mathrm{DT}(f_n) = n$ and $s(f_n) = O(n^{0.48})$. In addition, we give an infinite family of functions $\{g_n\}_{n \in I}$ with $s(g_n) = O(\mathrm{DT}(g_n)^{0.473})$.

2 Understanding $bs_3(f)$ is Important

As the upper and lower bounds for $bs_2(f)$ are almost matching, it seems that the next challenge is understanding the asymptotic behavior of $bs_3(f)$. A more modest challenge is the following.

▶ **Open Problem 4.** *Improve either the upper or lower bound on $bs_3(f)$.*

Recall that the upper bound on $bs_3(f)$ is $O(s(f)^3)$ (see Eq. (2)) and the lower bound is $(2/3) \cdot s(f)^2 \cdot (1 - o(1))$. It is somewhat surprising that any slight improvement on either the lower or upper bound on bs_3 would be a significant step forward in our understanding of the general question. The following claim shows that a slightly better than quadratic gap on a single example implies a better than quadratic gap on an infinite family of examples.

▶ **Claim 5.** *If there exists a function such that $bs_3(f) > s(f)^2$ then there exists a family of functions $\{f_n\}_{n\in\mathbb{N}}$ with $bs(f_n) > s(f_n)^{2+\varepsilon}$ for some constant $\varepsilon > 0$ (dependant on f).*

This family is simply $f_1 = f$, $f_n = f \circ f_{n-1}$ where $\circ$ stands for Boolean function composition as in [18]. Next, we prove a theorem exhibiting the self-reducibility nature of the problem.

▶ **Theorem 6.** *Let $k, \ell, a \in \mathbb{N}$ such that $\ell > k$ and let $T : \mathbb{N} \to \mathbb{R}$ be a monotone function. If $\forall f : bs_\ell(f) \leq T(bs_k(f))$, then $\forall f' : bs_{\ell a}(f') \leq T(bs_{ka}(f'))$.*

Proof. Assume by contradiction that there exists a function f' such that $bs_{\ell a}(f') > T(bs_{ka}(f'))$. We will show that there exists a function f such that $bs_\ell(f) > T(bs_k(f))$. We shall assume WLOG that the maximal $bs_{\ell a}$ of f' is achieved on $\vec{0}$. Let $B_1, B_2, \ldots, B_m$ be a family of disjoint sensitive blocks for f at $\vec{0}$, each B_i of size at most ℓa. Split every block B_i to ℓ sets $B_{i,1}, \ldots, B_{i,\ell}$ of size at most a. The function f will have a variable $x_{i,j}$ corresponding to every set $B_{i,j}$ of size at most a. The value of $f(x_{1,1}, \ldots, x_{m,\ell})$ is defined to be the value of f' where the variable in each $B_{i,j}$ equal $x_{i,j}$, and all other variables equal 0. $bs_\ell(f, \vec{0}) \geq bs_{\ell a}(f', \vec{0})$, since for any sensitive block $B_1, \ldots, B_m$ for f', there exists a corresponding sensitive block $B'_1, \ldots, B'_m$ for f of size ℓ, where $B'_i = \{x_{i,j} : j \in [\ell]\}$.

On the other hand, any set of disjoint sensitive blocks of size at most k for f corresponds to a disjoint set of sensitive blocks of size at most ka for f'. Thus $bs_k(f) \leq bs_{ka}(f')$, giving

$$T(bs_k(f)) \leq T(bs_{ka}(f')) < bs_{\ell a}(f') \leq bs_\ell(f) \ ,$$

where we used the monotonicity of T in the first inequality. ◀

Using Theorem 6 we get that any upper bound of the form $bs_\ell(f) \leq s(f)^{\ell-\varepsilon}$ implies a sub-exponential upper bound on $bs(f)$ in terms of $s(f)$.

▶ **Theorem 7.** *Let $k \in \mathbb{N}, \varepsilon > 0$ be constants. If for all Boolean functions $bs_k(f) \leq s(f)^{k-\epsilon}$, then for the constant $\gamma = \frac{\log(k-\varepsilon)}{\log(k)} < 1$ it holds that $bs(f) \leq 2^{O(s(f)^\gamma \cdot \log s(f))}$ for all f.*

For example, Theorem 7 shows that if $\forall f : bs_3(f) \leq s(f)^2$, then $\forall f : bs(f) \leq 2^{O(s^{0.631}\cdot\log(s))}$.

Proof. Using the hypothesis and Theorem 6 one can show by induction on t that

$$\forall f : bs_{k^t}(f) \leq s(f)^{(k-\epsilon)^t} \ . \tag{4}$$

The base case $t = 1$ is simply the hypothesis. We assume the claim is true for $1, \ldots, t-1$, and show the claim is true for t. Using Theorem 6 with $T(x) = x^{k-\epsilon}$ and $a = k^{t-1}$ we get $bs_{k^t}(f) \leq T(bs_{k^{t-1}}(f)) = (bs_{k^{t-1}}(f))^{k-\epsilon}$. By induction $bs_{k^{t-1}}(f) \leq s(f)^{(k-\epsilon)^{t-1}}$. Hence, we get $bs_{k^t}(f) \leq s(f)^{(k-\epsilon)^t}$, which finishes the induction proof.

Fix f and let $s = s(f)$. Recall that $bs(f) = bs_s(f)$ since each minimal block that flips the value of f is of size at most s. Hence,

$$\begin{aligned} bs(f) &= bs_s(f) = bs_{k^{\lceil \log_k(s) \rceil}}(f) \\ &\leq s^{(k-\epsilon)^{\lceil \log_k(s) \rceil}} \leq s^{(k-\epsilon)^{\log_k(s)+1}} = 2^{\log(s)\cdot s^{\log(k-\epsilon)/\log(k)}\cdot(k-\epsilon)} = 2^{O(s^\gamma \cdot \log(s))} \ . \end{aligned}$$ ◀

3 The Sensitivity Conjecture for Bounded Functions

In this section, we generalize the definitions of sensitivity and block sensitivity to bounded functions $f : \{0,1\}^n \to [0,1]$, extending the definitions for Boolean functions. We generalize the result of Kenyon and Kutin to this setting (after removing some trivial obstucles). Given

that, one may hope that the sensitivity conjecture holds also for bounded functions, i.e., that the block-sensitivity is at most polynomial in the sensitivity. However, we give a counterexample to this question, by constructing functions on n variables with sensitivity $O(\log n)$ and block sensitivity $n/\log(n)$. In fact, we show that the result of Kenyon and Kutin is essentially tight by giving examples for which $bs_\ell(f) = n/\ell$ and $s(f) = O(\ell \cdot n^{1/\ell})$ for any $\ell \leq \log n$.

We begin by generalizing the definitions of sensitivity and block-sensitivity. For $f : \{0,1\}^n \to [0,1]$ and $x \in \{0,1\}^n$, we denote the sensitivity of f at a point x by

$$s(f,x) = \sum_{i=1}^{n} |f(x) - f(x \oplus e_i)|. \tag{5}$$

Similarly we define the block sensitivity and ℓ-block sensitivity as

$$bs(f,x) = \max \left\{ \sum_i |f(x) - f(x + B_i)| \ : \ B_1, \ldots, B_k \subseteq [n] \text{ are disjoint} \right\} . \tag{6}$$

and

$$bs_\ell(f,x) = \max \left\{ \sum_i |f(x) - f(x + B_i)| \ : \ B_1, \ldots, B_k \subseteq [n] \text{ are disjoint and } \forall i. |B_i| \leq \ell \right\} .$$

Naturally we denote by $s(f) = \max_x s(f,x)$, by $bs(f) = \max_x bs(f,x)$ and by $bs_\ell(f) = \max_x bs_\ell(f,x)$. It is easy to see that for a Boolean function these definitions match the standard definitions of sensitivity, block sensitivity and ℓ-block sensitivity.

We wish to prove an analog of Kenyon-Kutin result, showing that $bs_\ell(f) \leq c_\ell \cdot s(f)^\ell$. However, stated as is the claim is false for a "silly" reason. Take any Boolean function f with a gap between the sensitivity and the ℓ-block sensitivity and take $g(x) = f(x)/s(f)$. Then, we get $s(g) = 1$ and $bs_\ell(g) = bs_\ell(f)/s(f)$. As there are examples with $bs_2(f) = n/2$ and $s(f) = \sqrt{n}$, we get that $bs_2(g) = \sqrt{n}/2$ while $s(g) = 1$, where n grows to infinity. This seems to rule out any relation between the sensitivity and block sensitivity (and even 2-block sensitivity) in the case of bounded functions. To overcome this triviality, we insist that the block sensitivity is close to n, or alternatively that changing each block dramatically changes the value of the function. Surprisingly, under this requirement we are able to retrieve known relations between sensitivity and block sensitivity that were established in the Boolean setting by Kenyon and Kutin [12].

▶ **Theorem 8.** *Let $c > 0$ and $f : \{0,1\}^n \to [0,1]$. Assume that there exists a point $x_0 \in \{0,1\}^n$ and disjoint blocks $B_1, \ldots, B_k$ of size at most ℓ such that $|f(x_0) - f(x_0 + B_i)| \geq c$ for all $i \in k$. Furthermore, assume that $2 \leq \ell \leq \log(k)$. Then, $s(f) \geq \Omega(k^{1/\ell} \cdot c)$.*

We get the following corollary, whose proof is deferred to Appendix A.

▶ **Corollary 9.** *Let $f : \{0,1\}^n \to [0,1]$ with $bs(f) \geq n/\ell$. Then, $s(f) \geq \Omega(n^{1/2\ell}/\ell)$.*

Unlike in the Boolean case, we are able to show that Theorem 8 is essentially tight! That is, for any ℓ and n we have a construction with $bs_\ell(f) \geq n/\ell$ and $s(f) = O(\ell \cdot n^{1/\ell})$. In particular, picking $\ell = \log(n)$ gives an exponential separation between block sensitivity (which is at least $n/\log n$) and sensitivity (which is $O(\log n)$).

▶ **Theorem 10.** *Let $\ell, n \in \mathbb{N}$ with $2 \leq \ell \leq n$. Then, there exists a function $h : \{0,1\}^n \to [0,1]$ with $bs_\ell(h) \geq \lfloor n/\ell \rfloor$ and $s(h) \leq 3 \cdot \ell \cdot n^{1/\ell}$.*

3.1 Proof of Kenyon-Kutin Result for Bounded Functions

Proof Overview. We start by giving a new proof for Kenyon-Kutin result, based on random walks on the hypercube. We assume by contradiction that $f(x_0) = 0$ and $f(x_0 + B_i) = 1$ for all $i \in [k]$ and that the sensitivity is $o(k^{1/\ell})$. Taking a random walk of length $r = n/k^{1/\ell}$ starting from x_0 will end up in point y where with high probability $f(y) = f(x_0)$. This is true since in each step with probability at least $1 - s(f)/n$ we are maintaining the value of f, hence by union bound with probability at least $1 - r \cdot s(f)/n$ we maintain the value of f in the entire walk. On the contrast, choosing a random $i \in [k]$ and starting a random walk of length $r - |B_i|$ starting from $(x_0 + B_i)$ will lead to a point y' where with high probability $f(y') = f(x_0 + B_i) = 1$. However, as we show in the proof below, the distributions of y and y' are similar (close in statistical distance). This leads to a contradiction as $f(y)$ tends to be equal to 0 and $f(y')$ tends to be equal to 1.

A simple observation, which allows us to generalize the argument above to bounded functions, is that for a given point $x \in \{0,1\}^n$ and a random neighbor in the hypercube, $y \sim x$, the expected value of $f(y)$ is close to $f(x)$. This follows from Eq. (5). Thus, the only difference in the argument for bounded functions will be that $\mathbf{E}[f(y)]$ is close to 0 and $\mathbf{E}[f(y')]$ is close to 1, leading to a contradiction as well.

Proof of Theorem 8. First, we make a few assumptions that are without loss of generality, in order to make the argument later clearer. We assume $x_0 = 0^n$ and $f(x_0) = 0$. We assume $n = k \cdot \ell$ and that the blocks are given by $B_i = \{(i-1)\ell + 1, \ldots, i\ell\}$ for $i \in [k]$. We assume that $c = 1$, since for $c < 1$ one can take $f'(x) = \min\{f(x)/c, 1\}$, and note that f' is a bounded function with $f'(x_0 + B_i) = 1$. Proving the theorem for f' gives $s(f) \geq s(f') \cdot c \geq \Omega(c \cdot k^{1/\ell})$.

Let $r = \lfloor \frac{n}{(2k)^{1/\ell}} \rfloor$, by the assumption $2 \leq \ell \leq \log(k)$ we have $\sqrt{n} \leq r \leq n/2$. Assume by contradiction that $s(f) \leq \varepsilon \cdot k^{1/\ell}$ for some sufficiently small constant $\varepsilon > 0$ to be determined later. Consider the following two random processes.

Algorithm 1 Process A

1: $X_0 \leftarrow 0^n$
2: **for** $t = 1, \ldots, r$ **do**
3: Select a random $i \in [n]$ among the coordinates for which X_{t-1} is 0 and let $X_t \leftarrow X_{t-1} + e_i$.
4: **end for**

Algorithm 2 Process B

1: Select uniformly $i \in [k]$ and let $Y_0 \leftarrow B_i$
2: **for** $t = 1, \ldots, r - \ell$ **do**
3: Select a random $i \in [n]$ among the coordinates for which Y_{t-1} is 0 and let $Y_t \leftarrow Y_{t-1} + e_i$.
4: **end for**

For each $t \in \{0, \ldots, r-1\}$, we claim that

$$\mathbf{E}[f(X_{t+1}) - f(X_t)] = \mathbf{E}\left[\frac{1}{n-t} \cdot \sum_{i:(X_t)_i=0} f(X_t + e_i) - f(X_t)\right]$$
$$\leq \frac{1}{n-t} \cdot \mathbf{E}[s(f(X_t))] \leq \frac{s(f)}{n-t} .$$

By telescoping this implies that

$$\mathbf{E}[f(X_r)] = \mathbf{E}[f(X_0)] + \sum_{t=0}^{r-1} \mathbf{E}[f(X_{t+1}) - f(X_t)] \leq 0 + \frac{r \cdot s(f)}{n-r} \leq O(\varepsilon) \ .$$

In a symmetric fashion, for each $t \in \{1, \ldots, r-\ell\}$ we have $\mathbf{E}[f(Y_{t+1}) - f(Y_t)] \geq -\frac{s(f)}{n-t-\ell}$. Again, telescoping implies that

$$\mathbf{E}[f(Y_{r-\ell})] \geq \mathbf{E}[f(Y_0)] - \frac{(r-\ell) \cdot s(f)}{n-r} \geq 1 - \frac{r \cdot s(f)}{n-r} \geq 1 - O(\varepsilon) \ .$$

So it seems that the distribution of X_r and $Y_{r-\ell}$ are very different from one another. However, we shall show that conditioned on a probable event, X_r and $Y_{r-\ell}$ are identically distributed. To define the event, consider the sets

$$U_i = \{\mathbb{1}_A \ \mid \ A \subseteq [n], |A| = r, B_i \subseteq A, \forall j \neq i : B_j \nsubseteq A\}$$

for $i \in [k]$ and their union

$$U = \bigcup_{i=1}^{k} U_i = \{\mathbb{1}_A \ \mid \ A \subseteq [n], |A| = r, \exists! i \in [k] : B_i \subseteq A\} \ .$$

Let E_X be the event that $X_r \in U$, and E_Y be the event that $Y_{r-\ell} \in U$. We show that

▶ **Claim 11.** *The following hold:*

1. $X_r|E_X$ *is identically distributed as* $Y_{r-\ell}|E_Y$.
2. $\mathbf{Pr}[E_Y] = \Omega(1)$
3. $\mathbf{Pr}[E_X] = \Omega(1)$

We defer the proof of Claim 11 for later. We derive a contradiction from all of the above by showing that $\mathbf{E}[f(X_r)|E_X] < \mathbf{E}[f(Y_{r-\ell})|E_Y]$ (this is indeed a contradiction because by the claim $X_r|E_X$ and $Y_{r-\ell}|E_Y$ should be identically distributed and hence the expected values of $f(\cdot)$ on each of them should be the same). To show this, we note that

$$\begin{aligned}\mathbf{E}[f(X_r)|E_X] &= \mathbf{E}[f(X_r) \cdot \mathbb{1}_{E_X}]/\mathbf{Pr}[E_X] \\ &\leq \mathbf{E}[f(X_r)]/\mathbf{Pr}[E_X] = O(\mathbf{E}[f(X_r)]) = O(\varepsilon) \ .\end{aligned}$$

On the other hand

$$\begin{aligned}\mathbf{E}[f(Y_{r-\ell})|E_Y] &= 1 - \mathbf{E}[1 - f(Y_{r-\ell})|E_Y] \\ &\geq 1 - \mathbf{E}[1 - f(Y_{r-\ell})]/\mathbf{Pr}[E_Y] = 1 - O(\mathbf{E}[1 - f(Y_{r-\ell})]) = 1 - O(\varepsilon) \ .\end{aligned}$$

Choosing ε to be a small enough constant implies that $\mathbf{E}[f(X_r)|E_X] < \mathbf{E}[f(Y_{r-\ell})|E_Y]$, which completes the proof. ◀

Proof of Claim 11. We shall use in the proof of Items 2 and 3 the fact that $1/3 \leq \frac{r^\ell k}{n^\ell} \leq 1/2$ which follows from the choice of $r = \lfloor \frac{n}{(2k)^{1/\ell}} \rfloor$ (for large enough n and k).

1. First note that X_r is distributed uniformly over the set of vectors in $\{0,1\}^n$ with hamming weight r. In particular, conditioning that X_r is in a set U of such vectors, makes it uniform over U. We are left to show that $Y_{r-\ell}|E_Y$ is distributed uniformly over U. Given that $Y_0 = B_i$, we have that $Y_{r-\ell}$ is the OR of $\mathbb{1}_{B_i}$ with a random vector of weight $r-\ell$ on $[n] \setminus B_i$. Conditioned on E_Y the only way to reach U_i is if $Y_0 = B_i$, hence, by the above, all points in U_i are attained with the same probability. Using symmetry, all points in $U = \bigcup_i U_i$ are attained with the same probability.

2. Let B_i be the block selected in the first step of Process B. We analyze the probability that all indices in B_j for some $j \neq i$ are chosen in the $r - \ell$ iterations of Process B.

$$\begin{aligned}\mathbf{Pr}[B_j \text{ is selected}] &= \frac{(\#\text{ of sequences where } B_j \text{ is selected})}{(\#\text{ of sequences})} \\ &= \frac{(r-\ell)^{\underline{\ell}} \cdot (n-2\ell)^{\underline{r-2\ell}}}{(n-\ell)^{\underline{r-\ell}}} = \frac{(r-\ell)!(n-2\ell)!(n-r)!}{(r-2\ell)!(n-r)!(n-\ell)!} \\ &= \frac{(r-\ell)!(n-2\ell)!}{(r-2\ell)!(n-\ell)!} = \frac{(r-\ell)\cdots(r-2\ell+1)}{(n-\ell)\cdots(n-2\ell+1)} \leq \left(\frac{r}{n}\right)^{\ell}\end{aligned}$$

(recall that $n^{\underline{k}} \triangleq \frac{n!}{(n-k)!}$). Hence, $\mathbf{Pr}[\exists j \neq i : B_j \text{ is selected}] \leq k \cdot (r/n)^{\ell} \leq 1/2$ and we have $\mathbf{Pr}[E_Y] \geq 1/2$.

3. Let $\pi_1, \ldots, \pi_r \in [n]$ be the sequence of choices made by Process A. For $i \in [k]$, let $E_{X,i}$ be the event that $X_r \in U_i$. By the uniqueness of the block contained in X_r the events $E_{X,i}$ are disjoint, hence $\mathbf{Pr}[E_X] = \sum_{i=1}^{k} \mathbf{Pr}[E_{X,i}]$. By symmetry, $\mathbf{Pr}[E_X] = k \cdot \mathbf{Pr}[E_{X,1}]$. The event $E_{X,1}$ is simply the event that there exists a set $S \subseteq [r]$ of size ℓ such that $\{\pi_j\}_{j \in S} = B_1$ and the sequence $\{\pi_j : j \in [r] \setminus S\}$ is a sequence of choices for which E_Y holds, when starting Process B from $Y_0 = B_1$. This shows that $\mathbf{Pr}[E_{X,1}] = \mathbf{Pr}[E_Y | Y_0 = B_1] \cdot \mathbf{Pr}[B_1 \subseteq \{\pi_1, \ldots, \pi_r\}]$. By Symmetry, $\mathbf{Pr}[E_Y | Y_0 = B_i] = \mathbf{Pr}[E_Y] = \Omega(1)$ from the previous item. In addition,

$$\begin{aligned}\mathbf{Pr}[B_1 \subseteq \{\pi_1, \ldots, \pi_r\}] &= \frac{r^{\underline{\ell}} \cdot (n-\ell)^{\underline{r-\ell}}}{n^{\underline{r}}} = \frac{r!(n-\ell)!(n-r)!}{(r-\ell)!(n-r)!n!} \\ &= \frac{r!(n-\ell)!}{(r-\ell)!n!} = \frac{r \cdots (r-\ell+1)}{n \cdots (n-\ell+1)} \geq \left(\frac{r-\ell}{n}\right)^{\ell} \\ &= \left(\frac{r}{n}\right)^{\ell} \cdot (1-\ell/r)^{\ell} = \left(\frac{r}{n}\right)^{\ell} \cdot (1-o(1))\end{aligned}$$

where $(1-\ell/r)^{\ell} = 1 - o(1)$ follows from $\ell \leq \log(k)$ and $r \geq \sqrt{n} \geq \sqrt{k}$. Thus,

$$\begin{aligned}\mathbf{Pr}[E_X] = k \cdot \mathbf{Pr}[E_{X,1}] &= k \cdot \mathbf{Pr}[B_1 \text{ is selected}] \cdot \mathbf{Pr}[E_Y | Y_0 = B_1] \\ &\geq k \cdot \left(\frac{r}{n}\right)^{\ell} \cdot (1-o(1)) \cdot \frac{1}{2} \geq \frac{1}{3} \cdot (1 - o(1) \cdot \frac{1}{2} = \Omega(1)\,.\end{aligned}$$

◀

3.2 Separating Sensitivity and Block Sensitivity of Bounded Functions

The Lattice Variant of The Sensitivity Conjecture

The proof of Theorem 10 is more natural in the lattice-variant of the sensitivity conjecture as suggested by Aaronson (see [6]). In this variant, instead of talking about functions over $\{0,1\}^n$ we are considering functions over $\{0,1,\ldots,\ell\}^k$ for $\ell, k \in \mathbb{N}$. Given a function $g : \{0,1,\ldots,\ell\}^k \to \mathbb{R}$ one can define a Boolean function $f : \{0,1\}^{\ell \cdot k} \to \mathbb{R}$ by the following equation:

$$f(x_{1,1}, \ldots, x_{k,\ell}) = g\left(\sum_{i=1}^{\ell} x_{1,i}, \ldots, \sum_{i=1}^{\ell} x_{k,i}\right). \tag{7}$$

For a point $y \in \{0,1,\ldots,\ell\}^k$ and function $g : \{0,\ldots,\ell\}^k \to \mathbb{R}$ one can define the sensitivity of g at y as

$$s(g,y) = \sum_{y' \sim y} |g(y') - g(y)|$$

where $y' \sim y$ if $y' \in \{0,\ldots,\ell\}^k$ is a neighbor of y in the grid $\{0,\ldots,\ell\}^k$, i.e., if y and y' agree on all coordinates except for one coordinate, say $j \in [k]$, on which $|y_j - y'_j| = 1$. The following claim relates the sensitivity of f to that of g.

▶ **Claim 12.** *Let $g : \{0,\ldots,\ell\}^k \to \mathbb{R}$ and let f be the function defined by Eq. (7). Then $s(f) \leq \ell \cdot s(g)$.*

Proof. Let $x = (x_{1,1},\ldots,x_{k,\ell}) \in \{0,1\}^{kl}$ and and let $x' \in \{0,1\}^{kl}$ be a neighbor of x, obtained by flipping the (i,j)-th coordinate. Let $y = (\sum_{i=1}^{\ell} x_{1,i},\ldots,\sum_{i=1}^{\ell} x_{k,i})$ and similarly let $y' = (\sum_{i=1}^{\ell} x'_{1,i},\ldots,\sum_{i=1}^{\ell} x'_{k,i})$. Then y and y' differ only on the i-th coordinate, and on this coordinate they differ by a ± 1. If $y'_i = y_i + 1$, then the number of neighbors $x' \sim x$ that are mapped to y' by $y' = (\sum_i x'_{1,i},\ldots,\sum_i x'_{k,i})$ equals the number of zeros in the i-th block of x, i.e., it equals $\ell - y_i$. Similarly, in the case $y'_i = y_i - 1$ the number of $x' \sim x$ that are mapped to y' equals y_i. In both cases, there are between 1 to ℓ points $x' \sim x$ that are mapped to each neighbor $y' \sim y$. Thus,

$$\sum_{x' \sim x} |f(x') - f(x)| = \sum_{x' \sim x} |g(y') - g(y)| \leq \ell \cdot \sum_{y' \sim y} |g(y') - g(y)| \,. \qquad \blacktriangleleft$$

Construction of a Separation. Let k, ℓ be integers. We construct $f : \{0,1,\ldots,\ell\}^k \to [0,1]$ such that $f(0) = 0$, $f(e_i \cdot \ell) = 1$ for all $i \in [k]$ and $s(f) \leq O(k^{1/\ell})$.

Define a weight function $w : \{0,1,\ldots,\ell\} \to [0,1]$ as follows: $w(a) = k^{a/\ell}/k$ for $a \in \{1,\ldots,\ell\}$ and $w(0) = 0$. Take $g : \{0,\ldots,\ell\}^k \to \mathbb{R}^+$ to be the function $g(x_1,\ldots,x_n) = \sum_{i=1}^{k} w(x_i)$ and take $f : \{0,\ldots,\ell\}^k \to [0,1]$ to be $f(x) = \min\{1, g(x)\}$. Then $f(0^k) = 0$ and $f(\ell \cdot e_i) = 1$ for all $i \in [k]$.

▶ **Theorem 13.** *$s(f) \leq 3 \cdot k^{1/\ell}$.*

Proof. Let $x \in \{0,1,\ldots,\ell\}^k$ be a point in the lattice. We distinguish between two cases $g(x) \geq 2$ and $g(x) < 2$. In the first case, all neighbors $x' \sim x$ have $g(x') \geq 1$ since the sums $\sum_i w(x_i)$ and $\sum_i w(x'_i)$ differ by at most 1. Since both $g(x)$ and $g(x')$ are at least 1 we get that $f(x) = f(x') = 1$ and the sensitivity of f at x is 0.

In the latter case, $g(x) < 2$, we bound the sensitivity as well. For ease of notation we extend w to be defined over $\{-1,\ldots,\ell+1\}$ by taking $w(\ell+1) = w(\ell)$ and $w(-1) = w(0)$. We extend also g to $\{-1,0,\ldots,\ell+1\} \to \mathbb{R}^+$ by taking $g(x_1,\ldots,x_n) = \sum_i w(x_i)$. We have

$$\begin{aligned}
s(f,x) \leq s(g,x) &= \sum_{i=1}^{k} |g(x+e_i) - g(x)| + |g(x) - g(x-e_i)| \\
&= \sum_{i=1}^{k} |w(x_i+1) - w(x_i)| + |w(x_i) - w(x_i-1)| \\
&= \sum_{i=1}^{k} w(x_i+1) - w(x_i-1) && (w \text{ is monotone}) \\
&\leq \sum_{i=1}^{k} w(x_i+1) && (w \text{ is non-negative}) \\
&\leq \sum_{i:x_i=0} w(1) + \sum_{i:x_i>0} w(x_i) \cdot k^{1/\ell} \\
&\leq k \cdot \frac{k^{1/\ell}}{k} + \sum_i w(x_i) \cdot k^{1/\ell} \\
&= k^{1/\ell} + g(x) \cdot k^{1/\ell} \leq 3k^{1/\ell}.
\end{aligned}$$ ◀

We show that Theorem 10 is a corollary of Theorem 13.

Proof of Theorem 10. Let $k = n/\ell$. Let $f : \{0,1,\ldots,\ell\}^k \to [0,1]$ be the function in Theorem 13. Take $h(x_{1,1},\ldots,x_{k,\ell}) = f\left(\sum_{i=1}^{\ell} x_{1,i},\ldots,\sum_{i=1}^{\ell} x_{n,i}\right)$. For $x = 0^n$, there are k disjoint blocks $B_1,\ldots,B_k$ of size ℓ each such that $h(x+B_i) = 1$. Hence, $bs_\ell(h) \geq k = n/\ell$. By Claim 12, the sensitivity of h is at most $s(f)\cdot\ell \leq 3\cdot k^{1/\ell}\cdot\ell \leq 3\cdot n^{1/\ell}\cdot\ell$ which completes the proof. ◀

4 New Separations between Decision Tree Complexity and Sensitivity

We report a new separation between the decision tree complexity and the sensitivity of Boolean functions. We construct an infinite family of Boolean functions with

$$\mathrm{DT}(f_n) \geq s(f_n)^{1+\log_{14}(19)} \geq s(f_n)^{2.115}.$$

Our functions are transitive functions, and are inspired by the work of Chakraborty [8].

Our construction is based on finding a "gadget" Boolean function f, defined over a constant number of variables, such that $s^0(f) = 1$, $s^1(f) = k$ and $\mathrm{DT}(f) = \ell$ for $\ell > k$ (recall that $s^0(f) = \max_{x:f(x)=0} s(f,x)$ and similarly $s^1(f) = \max_{x:f(x)=1} s(f,x)$). Given the gadget f, we construct an infinite family of functions with super-quadratic gap between the sensitivity and the decision tree complexity using compositions (which is a well-used trick in query complexity separations, cf. [18]).

▶ **Lemma 14.** *Let $f : \{0,1\}^c \to \{0,1\}$ such that $s^0(f) = 1$, $s^1(f) = k$ and $\mathrm{DT}(f) = \ell > k$. Then, there exists an infinite family of functions $\{g_i\}_{i\in\mathbb{N}}$ such that $s(g_i) = k^i$ and $\mathrm{DT}(g_i) = (k\ell)^i = s(g_i)^{1+\log(k)/\log(\ell)}$.*

Proof. Take $g = \mathrm{OR}_k \circ f$. It is easy to verify that $s(g) = k$, and that $\mathrm{DT}(g) = \mathrm{DT}(\mathrm{OR}_k)\cdot \mathrm{DT}(f) = k\ell$ (for the latter, one can use [18, Lemma 3.1]). For $i \in \mathbb{N}$, we take $g_i = g^i$. It is well-known (cf. [18, Lemma 3.1]) that $s(g^i) \leq s(g)^i$ and that $\mathrm{DT}(g^i) = \mathrm{DT}(g)^i$, which completes the proof. ◀

4.1 Finding a Good Gadget

The gadget f will be a minterm-cyclic function. Roughly speaking, a function $f : \{0,1\}^n \to \{0,1\}$ is minterm-cyclic if there exists pattern $p \in \{0,1,*\}^n$ such that the function f simply checks if x matches one of the cyclic shifts of p. The formal definition follows

▶ **Definition 15.** A pattern $p \in \{0,1,*\}^n$ is a partial assignment to the variables $x_1,\ldots,x_n$. We say that a point $x \in \{0,1\}^n$ matches the pattern p, denoted by $p \subseteq x$, if for all $i \in [n]$ such that $p_i \in \{0,1\}$ we have $p_i = x_i$. Given a pattern p, let $CS(p) = \{p^1,\ldots,p^n\}$ be the set of cyclic shifts of p, where the i-th cyclic shift of p is given by $p^i = (p_i, p_{i+1},\ldots,p_n,p_1,\ldots,p_{i-1})$. For a pattern $p \in \{0,1,*\}^n$ we denote by $f_p : \{0,1\}^n \to \{0,1\}$ the function defined by

$$f_p(x) = 1 \quad \Leftrightarrow \quad \exists p^i \in CS(p) : p^i \subseteq x$$

and call f_p the minterm cyclic function defined by p.

For example, the pattern $p = \texttt{0011**}$ defines a function f_p that checks if there's a sequence of two zeros followed by two ones in x, when x is viewed as a cyclic string. We say that two patterns $p, q \in \{0,1,*\}^n$ disagree on a coordinate i if both p_i and q_i are in $\{0,1\}$ and $p_i \neq q_i$.

▶ **Claim 16.** *Let $p \in \{0,1,*\}^n$ be a pattern defining $f_p : \{0,1\}^n \to \{0,1\}$. Assume that any two different cyclic-shifts of p disagree on at least 3 coordinates. Then, $s^0(f_p) = 1$.*

Proof. Let $x \in \{0,1\}^n$ with $f_p(x) = 0$ and assume by contradiction that $s(f_p, x) \geq 2$. In such a case, there are two indices i and j such that $f_p(x + e_i) = 1$ and $f_p(x + e_j) = 1$. Let q and q' be the patterns among $CS(p)$ that $x + e_i$ and $x + e_j$ satisfy respectively. If $q = q'$, then since both $x + e_i$ and $x + e_j$ satisfy q and they differ on coordinates i and j, it must be the case that $q_i = q_j = *$. However, this implies that x satisfy q as well, which is a contradiction. If $q \neq q'$, then we get that q and q' may disagree only on coordinates i and j, which is also a contradiction. ◀

The following fact is easy to verify.

▶ **Fact 17.** *Let $p \in \{0,1,*\}^n$ be a pattern defining $f_p : \{0,1\}^n \to \{0,1\}$. Then, $s^0(f_p) \leq c^0(f_p) \leq |\{i \in [n] : p_i \in \{0,1\}\}|$.*

Next, we demonstrate a simple example with better-than-quadratic separation between $\mathrm{DT}(f)$ and $s(f)$. Take the pattern $p = *001011$. Denote by $p^1, \ldots, p^7$ all the cyclic shifts of p, where in p^i the i-th coordinate equals $*$. It is easy to verify that any p^i and p^j for $i \neq j$ disagree on at least 3 coordinates. Hence, $s^0(f_p) = 1$ and $s^1(f_p) \leq 6$. We wish to show that any decision tree T for f_p is of depth 7. Let x_i be the first coordinate read by a decision tree T for f_p. Our adversary will answer 0, and will continue to answer as if x matches p^i. Assume the decision tree made a decision before reading the entire input. The decision tree must decide 1 since the adversary answered according to x which satisfies p^i. However, if the decision tree hasn't read the entire input, there is still an unread coordinate j, where $j \neq i$. Let $x' = x + e_j$. Then, the decision tree answers 1 on x' as well. However x' does not match pattern p^i as $(p^i)_j \in \{0,1\}$ and it must be the case that $x_j = (p^i)_j \neq x'_j$.

We also need to rule out that x' matches some other pattern. Indeed, if x' matches some other pattern p^k it means that p^k and p^i disagree only on at most one coordinate, which as discussed above cannot happen.

Using Lemma 14 the function f_p can be turned into an infinite family of functions g_i with $\mathrm{DT}(g_i) = (6 \cdot 7)^i$ and $s(g_i) \leq 6^i$. This gives a super-quadratic separation since

$$\mathrm{DT}(g_i) \geq s(g_i)^{1+\log(7)/\log(6)} \geq s(g_i)^{2.086}.$$

In a similar fashion, one can show that for the pattern $p = \texttt{**0*10000*101}$ after reading any two input bits from the input there exists a cyclic shift p^i of the pattern from which no $\{0,1\}$ coordinate has been read yet. However, to verify that the input x matches p^i we must read all $\{0,1\}$ positions in p^i, which gives $\mathrm{DT}(f_p) \geq 9 + 2$ where 9 is the number of $\{0,1\}$-s in the pattern p.

The decision tree complexity analysis for the other patterns written below is more involved. We computed it using a computer program written to calculate the decision tree complexity in this special case. In the list below, we report several patterns yielding super-quadaratic separations. For each pattern p we report its length n, the decision tree complexity of f_p, the maximal sensitivity of f_p (which equals the number of $\{0,1\}$-s in p) and the resulting exponent one get by using Lemma 14 (i.e., $1 + \frac{\log \mathrm{DT}(f_p)}{\log s(f_p)}$).

```
p = *001011,                          n = 7,  DT = 7,  s = 6,  exp = 2.086
p = **0*10000*101,                    n = 13, DT = 11, s = 9,  exp = 2.091
p = ******01*1*01100000,              n = 19, DT = 14, s = 11, exp = 2.100
p = ******00*0*0010**1*00*011,        n = 25, DT = 17, s = 13, exp = 2.104
p = ******1**0**0**1**0**00*0*10*1011, n = 33, DT = 19, s = 14, exp = 2.115
```

Acknowledgements. I wish to thank my PhD advisor, Ran Raz, for lots of stimulating and helpful discussions about this problem. I wish to thank Scott Aaronson for his encouragement.

References

1. A. Ambainis, M. Bavarian, Y. Gao, J. Mao, X. Sun, and S. Zuo. Tighter relations between sensitivity and other complexity measures. In *ICALP (1)*, pages 101–113, 2014. doi:10.1007/978-3-662-43948-7_9.
2. A. Ambainis and K. Prusis. A tight lower bound on certificate complexity in terms of block sensitivity and sensitivity. In *MFCS*, pages 33–44, 2014.
3. A. Ambainis, K. Prusis, and J. Vihrovs. Sensitivity versus certificate complexity of boolean functions. *CoRR*, abs/1503.07691, 2015.
4. A. Ambainis and X. Sun. New separation between *s(f)* and *bs(f)*. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:116, 2011.
5. A. Ambainis and J. Vihrovs. Size of sets with small sensitivity: A generalization of simon's lemma. In *Theory and Applications of Models of Computation – 12th Annual Conference, TAMC 2015, Singapore, May 18-20, 2015, Proceedings*, pages 122–133, 2015.
6. M. Boppana. Lattice variant of the sensitivity conjecture. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:89, 2012.
7. H. Buhrman and R. de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.
8. S. Chakraborty. On the sensitivity of cyclically-invariant boolean functions. *Discrete Mathematics & Theoretical Computer Science*, 13(4):51–60, 2011.
9. J. Gilmer, M. Koucký, and M. E. Saks. A new approach to the sensitivity conjecture. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 247–254, 2015.
10. P. Gopalan, N. Nisan, R. A. Servedio, K. Talwar, and A. Wigderson. Smooth boolean functions are easy: Efficient algorithms for low-sensitivity functions. In *ITCS*, pages 59–70, 2016.
11. P. Hatami, R. Kulkarni, and D. Pankratov. Variations on the sensitivity conjecture. *Theory of Computing, Graduate Surveys*, 2:1–27, 2011.
12. C. Kenyon and S. Kutin. Sensitivity, block sensitivity, and l-block sensitivity of boolean functions. *Inf. Comput.*, 189(1):43–53, 2004. doi:10.1016/j.ic.2002.12.001.
13. N. Nisan. Crew prams and decision trees. In *STOC*, pages 327–335, 1989. doi:10.1145/73007.73038.
14. N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.
15. D. Rubinstein. Sensitivity vs. block sensitivity of boolean functions. *Combinatorica*, 15(2):297–299, 1995. doi:10.1007/BF01200762.
16. H. U. Simon. A tight $\Omega(\log\log n)$-bound on the time for parallel ram's to compute nondegenerated boolean functions. In *Foundations of computation theory*, pages 439–444. Springer, 1983.
17. M. Szegedy. An $O(n^{0.4732})$ upper bound on the complexity of the GKS communication game. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:102, 2015.
18. A. Tal. Properties and applications of boolean function composition. In *ITCS*, pages 441–454, 2013.
19. A. Tal. On the sensitivity conjecture. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:62, 2016.
20. M. Virza. Sensitivity versus block sensitivity of boolean functions. *Inf. Process. Lett.*, 111(9):433–435, 2011. doi:10.1016/j.ipl.2011.02.001.

A Proof of Corollary 9

Proof. Let $x \in \{0,1\}^n$ and $B_1, \ldots, B_m$ be the blocks that achieve $bs(f)$. Assume without loss of generality that $B_1, \ldots, B_{m'}$ are of size at most 2ℓ and that $B_{m'+1}, \ldots, B_m$ are of size larger than 2ℓ. Then, by the disjointness of $B_{m'+1}, \ldots, B_m$ we have that $m - m' \leq \frac{n}{2\ell}$. Thus,

$$\begin{aligned} bs_\ell(f,x) &\geq \sum_{i=1}^{m'} |f(x) - f(x+B_i)| = \sum_{i=1}^{m} |f(x) - f(x+B_i)| - \sum_{i=m'+1}^{m} |f(x) - f(x+B_i)| \\ &\geq bs(f,x) - (m-m') \geq bs(f,x) - \frac{n}{2\ell} \geq \frac{n}{2\ell} \,. \end{aligned}$$

Assume without loss of generality that $B_1, \ldots, B_{m''}$ are blocks such that $|f(x) - f(x+B_i)| \geq \frac{1}{4\ell}$ and that $B_{m''+1}, \ldots, B_{m'}$ are not. Then, $\sum_{i=m''+1}^{m'} |f(x) - f(x+B_i)| \leq \frac{m''-m'}{4\ell} \leq \frac{n}{4\ell}$. This implies that $\sum_{i=1}^{m''} |f(x) - f(x+B_i)| \geq \frac{n}{4\ell}$, and in particular that $m'' \geq \frac{n}{4\ell}$. Thus, there are $m'' \geq n/4\ell$ disjoint blocks of size at most 2ℓ which change the value of f by at least $\frac{1}{4\ell}$. Theorem 8 gives that $s(f) \geq \Omega((m'')^{1/2\ell}/\ell) \geq \Omega(n^{1/2\ell}/\ell)$. ◀

Randomization Can Be as Helpful as a Glimpse of the Future in Online Computation*

Jesper W. Mikkelsen†

University of Southern Denmark, Odense, Denmark
jesperwm@imada.sdu.dk

Abstract

We provide simple but surprisingly useful direct product theorems for proving lower bounds on online algorithms with a limited amount of advice about the future. Intuitively, our direct product theorems say that if b bits of advice are needed to ensure a cost of at most t for some problem, then $r \cdot b$ bits of advice are needed to ensure a total cost of at most $r \cdot t$ when solving r independent instances of the problem. Using our direct product theorems, we are able to translate decades of research on randomized online algorithms to the advice complexity model. Doing so improves significantly on the previous best advice complexity lower bounds for many online problems, or provides the first known lower bounds. For example, we show that

- A paging algorithm needs $\Omega(n)$ bits of advice to achieve a competitive ratio better than $H_k = \Omega(\log k)$, where k is the cache size. Previously, it was only known that $\Omega(n)$ bits of advice were necessary to achieve a constant competitive ratio smaller than $5/4$.
- Every $O(n^{1-\varepsilon})$-competitive vertex coloring algorithm must use $\Omega(n \log n)$ bits of advice. Previously, it was only known that $\Omega(n \log n)$ bits of advice were necessary to be optimal.

For certain online problems, including the MTS, k-server, metric matching, paging, list update, and dynamic binary search tree problem, we prove that randomization and sublinear advice are equally powerful (if the underlying metric space or node set is finite). This means that several long-standing open questions regarding randomized online algorithms can be equivalently stated as questions regarding online algorithms with sublinear advice. For example, we show that there exists a deterministic $O(\log k)$-competitive k-server algorithm with sublinear advice if and only if there exists a randomized $O(\log k)$-competitive k-server algorithm without advice.

Technically, our main direct product theorem is obtained by extending an information theoretical lower bound technique due to Emek, Fraigniaud, Korman, and Rosén [ICALP'09].

1998 ACM Subject Classification F.1.2 Models of Computation (online computation)

Keywords and phrases online algorithms, advice complexity, information theory, randomization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.39

1 Introduction

The model of online computation deals with optimization problems where the input arrives sequentially over time. Usually, it is assumed that an online algorithm has no knowledge of future parts of the input. While this is a natural assumption, it leaves open the possibility that a tiny amount of information about the future (which might be available in practical applications) could dramatically improve the performance guarantee of an online algorithm.

* Most proofs have been omitted due to space restrictions. A full version of the paper containing all proofs and technical details is available at http://arxiv.org/abs/1511.05886.

† This work was partially supported by the Villum Foundation and the Danish Council for Independent Research, Natural Sciences.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 39; pp. 39:1–39:14

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

Recently, the notion of advice complexity [17, 29, 34, 28] was introduced in an attempt to provide a quantitative and problem-independent framework for studying *semi-online* algorithms with limited (instead of non-existing) knowledge of the future. In this framework, the limited knowledge is modeled as a number of advice bits provided to the algorithm by an oracle (see Definition 7). The goal is to determine how much advice (measured in the length, n, of the input) is needed to achieve a certain competitive ratio. In particular, one of the most important questions is how much advice is needed to break the lower bounds for (randomized) online algorithms without advice. It has been shown that for e.g. bin packing and makespan minimization on identical machines, $O(1)$ bits of advice suffice to achieve a better competitive ratio than what is possible using only randomization [5, 3]. On the other hand, for a problem such as edge coloring, it is known that $\Omega(n)$ bits of advice are needed to achieve a competitive ratio better than that of the best deterministic online algorithm without advice [41]. However, for many online problems, determining the power of a small amount of advice has remained an open problem.

With a few notable exceptions (e.g. [15, 29, 16, 20]), most of the previous research on advice complexity has been problem specific. In this paper, we take a more complexity-theoretic approach and focus on developing techniques that are applicable to many different problems. Our main conceptual contribution is a better understanding of the connection between advice and randomization. Before explaining our results in details, we briefly review the most relevant previous work.

Standard derandomization techniques imply that a randomized algorithm can be converted (maintaining its competitiveness) into a deterministic algorithm with advice complexity $O(\log\log I(n) + \log n)$, where $I(n)$ is the number of inputs of length n [16]. Clearly, there are problems where even a single bit of advice is much more powerful than randomization, and so we cannot in general hope to convert an algorithm with advice into a randomized algorithm. However, using machine learning techniques, Blum and Burch have shown that a metrical task system algorithm with advice complexity $O(1)$ can be converted into a randomized algorithm without advice [12]. Problems such as paging, k-server, and list update can be modeled as metrical task systems (see e.g. [18]).

2 Overview of results and techniques

We will give a high-level description of the results and techniques introduced in this paper. For simplicity, we restrict ourselves to the case of minimization problems[1].

Direct product theorems. Central to our work is the concept of an *r-round input distribution*. Informally, this is a distribution over inputs that are made up of r rounds such that the requests revealed in each round are selected independently of all previous rounds. Furthermore, there must be a fixed upper bound on the length of each round (see Definition 8).

As the main technical contribution of the paper, we prove direct product theorems for r-round input distributions. Intuitively, a direct product theorem says that if b bits of advice are needed to ensure a cost of at most t for each individual round, then rb bits of advice are needed to ensure a cost of at most rt for the entire input. This gives rise to a useful technique for proving advice complexity lower bounds. In particular, it follows that a linear number of advice bits are needed to get a (non-trivial) improvement over algorithms without any advice at all.

[1] All of our results (and their proofs) are easily adapted to maximization problems. See [42] for details.

We provide two different theorems formalizing the above intuitive statement in different ways: Our main direct product theorem (Theorem 10) is based on an information theoretical argument similar to that of [29]. In the full paper [42], we also provide an alternative direct product theorem based on martingale theory. In this extended abstract, we will only consider the information theoretical version.

Repeatable online problems. We combine our direct product theorems with the following very simple idea: Suppose that we have a lower bound on the competitive ratio of randomized online algorithms without advice. Often, such a lower bound is proved by constructing a hard input distribution and then appealing to Yao's principle. For some online problems, it is always possible to combine (in a meaningful way) a set of input sequences $\{\sigma_1, \ldots, \sigma_r\}$ into one long input sequence $\sigma = g(\sigma_1, \ldots, \sigma_r)$ such that serving σ essentially amounts to serving the r smaller inputs individually and adding the costs incurred for serving each of them. We say that such problems are Σ-*repeatable* (see Definition 12). For a Σ-repeatable online problem, an adversary can draw r input sequences independently at random according to some hard input distribution. This gives rise to an r-round input distribution. By our direct product theorem, an online algorithm needs linear advice (in the length of the input) to do better against this r-round input distribution than an online algorithm without advice. Thus, for Σ-repeatable online problems, we get that it is possible to translate lower bounds for randomized algorithms without advice into lower bounds for algorithms with sublinear advice. More precisely, we obtain the following theorem.

▶ **Theorem 1.** *Let* P *be a* Σ*-repeatable online minimization problem and let* c *be a constant. Suppose that for every* $\varepsilon > 0$ *and every* α*, there exists an input distribution* $p_{\alpha,\varepsilon} : I \to [0,1]$ *with finite support such that* $\mathbb{E}_{p_{\alpha,\varepsilon}}[D(\sigma)] \geq (c-\varepsilon)\,\mathbb{E}_{p_{\alpha,\varepsilon}}[OPT(\sigma)] + \alpha$ *for every deterministic algorithm* D *without advice. Then, every randomized algorithm reading at most* $o(n)$ *bits of advice on inputs of length* n *has a competitive ratio of at least* c*.*

Much research has been devoted to obtaining lower bounds for randomized algorithms without advice. Theorem 1 makes it possible to translate many of these lower bounds into advice complexity lower bounds, often resulting in a significant improvement over the previous best lower bounds (see Table 1).

For a Σ-repeatable problem, the total cost has to be the sum of costs incurred in each individual round. It is also possible to consider another kind of repeatable problems, where the total cost is the *maximum cost* incurred in a single round. We call such problems $\vee$-repeatable. Many online coloring problems are $\vee$-repeatable. For $\vee$-repeatable problems, we show in the full version of the paper [42] that under certain conditions, a constant lower bound on the competitive ratio of *deterministic* algorithms without advice carries over to randomized algorithms, even if the randomized algorithms have advice complexity $o(n)$. The proof of this result is straightforward and does not rely on our direct product theorems for Σ-repeatable problems. However, the result improves or simplifies a number of previously known advice complexity lower bounds for $\vee$-repeatable problems.

In Table 1, we have listed most of the repeatable online problems for which lower bounds on algorithms with sublinear advice existed prior to our work, and compared those previous lower bounds with the lower bounds that we obtain in this paper. We have also included two examples of repeatable online problems for which Theorem 1 provides the first known advice complexity lower bounds.

It is evident from Table 1 that there are many repeatable online problems. On the other hand, let us mention that e.g. bin packing and makespan minimization are examples of problems which are provably not repeatable.

Table 1 New and previously best lower bounds on the competitive ratio for algorithms reading $o(n)$ bits of advice on inputs of length n. For the Σ-repeatable problems, the lower bounds in this table are obtained by combining Theorem 1 with known lower bounds for randomized online algorithms without advice. For the $\vee$-repeatable problems, the lower bounds are obtained by combining our general result on $\vee$-repeatable problems with known lower bounds for *deterministic* algorithms without advice. In both cases, references to these previously known lower bounds for online algorithms without advice are provided in the second column of the table. We refer to the full paper [42] for a more detailed explanation of the entries in Table 1, and for a comparison with the current upper bounds.
Bipartite matching and Max-SAT are maximization problems, and hence the lower bound is obtained via the maximization version of Theorem 1. For paging and reordering buffer management, k denotes the cache/buffer size. For metrical task systems, N is the number of states. For the metrical task system problem and the k-server problem, the bounds are for a worst-case metric. It is also possible to use Theorem 1 together with known lower bounds for specific metric spaces. The lower bound for unit clustering is for the one-dimensional case.

	Lower bound for algorithms with advice complexity $o(n)$			
Σ-repeatable problem	**This work**		Previous best	
Paging	$\Omega(\log k)$	[19]	5/4	[17]
k-Server	$\Omega(\log k)$	[19]	3/2	[46]
2-Server	$1 + e^{-1/2}$	[25]	3/2	[46]
List Update	3/2	[47]	15/14	[22]
Metrical Task Systems	$\Omega(\log N)$	[19]	$\Omega(\log N)$	[29]
Bipartite Matching	$e/(e-1)$	[37]	$1+\varepsilon$	[43]
Reordering Buffer Management	$\Omega(\log\log k)$	[1]	$1+\varepsilon$	[2]
2-Sleep States Management	$e/(e-1)$	[36]	7/6	[13]
Unit clustering	3/2	[30]	–	
Max-SAT	3/2	[7]	–	
$\vee$-repeatable problem	**This work**		Previous best	
Edge Coloring	2	[8]	2	[41]
$L(2,1)$-Coloring on Paths	3/2	[11]	3/2	[11]
2-Vertex-Coloring	$\omega(1)$	[9]	2	[10]

Compact online problems. Note that when translating a lower bound on randomized algorithms without advice to a lower bound on algorithms with $o(n)$ bits of advice via Theorem 1, we had to make some assumptions on the lower bound. This begs the following question: Are there online problems where every lower bound (after suitable modifications) satisfies these assumptions? In order to formally answer this question, we make the following definition.

▶ **Definition 2.** Let P be a minimization problem and let $c > 1$ be a constant such that the expected competitive ratio of every randomized P-algorithm is at least c. We say that P is *compact* if for every $\varepsilon > 0$ and every $\alpha \geq 0$, there exists an input distribution $p_{\alpha,\varepsilon}$ with finite support such that if $\mathtt{D}$ is a deterministic online algorithm (without advice), then $\mathbb{E}_{p_{\alpha,\varepsilon}}[\mathtt{D}(\sigma)] \geq (c-\varepsilon) \cdot \mathbb{E}_{p_{\alpha,\varepsilon}}[\mathtt{OPT}(\sigma)] + \alpha$.

If an online problem is both Σ-repeatable and compact, we get the following result: Let c be a constant and let $\varepsilon > 0$. If c is a lower bound on the competitive ratio of randomized online algorithms without advice, then $c - \varepsilon$ is also a lower bound on the competitive ratio of online algorithms with sublinear advice. Equivalently, by contraposition, the existence of a c-competitive algorithm with sublinear advice implies the existence of a $(c + \varepsilon)$-competitive randomized algorithm without advice. Combining this with existing derandomization results [16] (see also [42]) yields the following complexity theoretic equivalence between randomization and sublinear advice (previously, only the forward implication was known [16]):

▶ **Theorem 3.** *Let* P *be a compact and* Σ*-repeatable minimization problem with at most* $2^{n^{O(1)}}$ *inputs of length* n*, and let* c *be a constant independent of* n*. The following are equivalent:*

1. *For every* $\varepsilon > 0$*, there exists a randomized* $(c + \varepsilon)$*-competitive* P*-algorithm without advice.*
2. *For every* $\varepsilon > 0$*, there exists a deterministic* $(c + \varepsilon)$*-competitive* P*-algorithm with advice complexity* $o(n)$*.*

In this paper, we use a technique due to Ambühl [4] and Mömke [44] to show that the class of compact and Σ-repeatable problems contains all problems which can be modeled as a metrical task system (MTS) with a finite number of states and tasks. This means that e.g. the k-server, list update, paging, and dynamic binary search tree problem are all compact and Σ-repeatable, assuming that the underlying metric space or node set is finite. For all these problems, it is known that it is possible to achieve a constant competitive ratio with respect to the length of the input [42]. Also, the number of inputs of length n for each of these problems is at most $2^{n^{O(1)}}$ (this bound holds since when we apply Theorem 3 to e.g. the k-server problem, the metric space will be fixed and not a part of the input). Thus, for each of these problems, Theorem 3 applies. Furthermore, for all the problems just mentioned (except paging), determining the best possible competitive ratio of a randomized algorithm without advice is regarded as important open problems [18, 27]. For example, the randomized k-server conjecture says that there exists a randomized $O(\log k)$-competitive k-server algorithm [39]. Theorem 3 shows that this conjecture is equivalent to the conjecture that there exists a deterministic $O(\log k)$-competitive k-server algorithm with advice complexity $o(n)$. Currently, it is only known how to achieve a competitive ratio of $O(\log k)$ using $2n$ bits of advice [16, 45].

We also show that there are compact and Σ-repeatable problems which cannot be modeled as a MTS. One such example is the metric matching problem (on finite metric spaces) [35].

Pessimistically, Theorem 3 may be seen as a barrier result which says that (for compact and Σ-repeatable online problems) designing an algorithm with sublinear advice complexity and a better competitive ratio than the currently best randomized algorithm without advice might be very difficult. Optimistically, one could hope that this equivalence might be useful in trying to narrow the gap between upper and lower bounds on the best possible competitive ratio of randomized algorithms without advice. In all cases, Theorem 3 shows that understanding better the exact power of (sublinear) advice in online computation would be very useful.

2.1 Other applications of our direct product theorem

The previously mentioned applications of our direct product theorem treat the hard input distribution as a black-box. However, it is also possible to apply our direct product theorem to explicit input distributions. Doing so yields some interesting lower bounds which cannot be obtained via Theorem 1. In what follows, we will state and briefly discuss three such lower bounds. We refer to the full paper [42] for details and for the proofs of Theorems 4 to 6.

Repeated matrix games. Let $q \in \mathbb{N}$ and let $A \in \mathbb{R}_+^{q \times q}$ be a matrix defining a two-player zero-sum game. Let V denote the value of the game defined by A. The *repeated matrix game (RMG)* with cost matrix A is an online problem where the algorithm and adversary repeatedly plays the game defined by A. The adversary is the row-player, the algorithm is the column-player, and the matrix A specifies the cost incurred by the online algorithm in each round. This generalizes the string guessing problem [15] and the generalized matching pennies problem [29] (both of these essentially corresponds to the RMG with a $q \times q$ matrix A where $A(i,j) = 1$ if $i \neq j$ and $A(i,i) = 0$). Using our direct product theorem, we easily get that for every $\varepsilon > 0$, an online algorithm which on inputs of length n is guaranteed to incur a cost of at most $(V - \varepsilon)n$ must read $\Omega(n)$ bits of advice.

▶ **Theorem 4.** *Let* `ALG` *be an algorithm for the RMG with cost matrix A. Furthermore, let V be the value of the (one-shot) two-person zero-sum game defined by A and let $0 < \varepsilon \leq V$ be a constant. If $\mathbb{E}[\texttt{ALG}(\sigma)] \leq (V - \varepsilon)n$ for every input σ of length n, then* `ALG` *must read at least*

$$b \geq \frac{\varepsilon^2}{2\ln(2) \cdot \|A\|_\infty^2} n = \Omega(n) \tag{1}$$

bits of advice.

Furthermore, we also show how a more careful application of our direct product theorem to some particular repeated matrix games yields good trade-off results for the exact amount of advice needed to ensure a cost of at most αn for $0 < \alpha < V$.

A better bin packing lower bound via repeated matrix games. We use our results on repeated matrix games to prove the following advice complexity lower bound for bin packing:

▶ **Theorem 5.** *Let $c < 4 - 2\sqrt{2}$ be a constant. A randomized c-competitive bin packing algorithm must read at least $\Omega(n)$ bits of advice.*

Previously, Angelopoulos et al. showed that a bin packing algorithm with a competitive ratio of $c < 7/6$ had to use $\Omega(n)$ bits of advice by a reduction from the binary string guessing problem [5, 23]. From our results on repeated matrix games, we obtain a lower bound for *weighted* binary string guessing. Using the same reduction as in [5], but reducing from weighted binary string guessing instead, we improve the lower bound for bin packing algorithms with sublinear advice to $4 - 2\sqrt{2}$. Thus, even though bin packing itself is not repeatable, we can obtain a better lower bound via a reduction from a repeated matrix game.

Superlinear lower bounds for graph coloring. We obtain the following superlinear lower bound for online graph coloring by applying our direct product theorem to an ingenious hard input distribution due to Halldórsson and Szegedy [33] (they show that a randomized graph coloring algorithm *without* advice must have a competitive ratio of at least $\Omega(n/\log^2 n)$):

▶ **Theorem 6.** *Let $\varepsilon > 0$ be a constant. A randomized $O(n^{1-\varepsilon})$-competitive online graph coloring algorithm must read at least $\Omega(n \log n)$ bits of advice.*

Previously, it was only known that $\Omega(n \log n)$ bits of advice were necessary to be 1-competitive [32]. Note that $O(n \log n)$ bits of advice trivially suffice to achieve optimality for graph coloring. Furthermore, it is not hard to prove that for every $c = n^{1-o(1)}$, there exists a c-competitive graph coloring algorithm reading $o(n \log n)$ bits of advice. Thus, our lower bound saying that $\Omega(n \log n)$ bits are needed to be $O(n^{1-\varepsilon})$-competitive for every constant $\varepsilon > 0$ is essentially tight.

3 Relation to machine learning

Theorem 3 is very closely related to previous work on combining online algorithms [12, 31, 6]. Let $A_1, \ldots, A_m$ be m algorithms for a MTS of finite diameter Δ. Based on a variant of the celebrated machine learning algorithm Randomized Weighted Majority (RWM) [40], Blum and Burch obtained the following result [12, 24]: For every $\varepsilon > 0$, it is possible to combine the m algorithms into a single randomized MTS-algorithm, R, such that

$$\mathbb{E}[R(\sigma)] = (1 + 2\varepsilon) \cdot \min_{1 \leq i \leq m} A_i(\sigma) + \left(\frac{7}{6} + \frac{1}{\varepsilon}\right) \Delta \ln m, \tag{2}$$

for every input σ. An algorithm with b bits of advice corresponds to an algorithm which runs $m = 2^b$ algorithms in parallel (and selects the best one at the end). Thus, equation (2) immediately implies that given a c-competitive MTS-algorithm with advice complexity $b = O(1)$, we can convert it to a randomized $(c + \varepsilon)$-competitive algorithm without advice.

Theorem 3 improves on the result of Blum and Burch in two ways. First of all, it allows us to convert algorithms with sublinear instead of only constant advice complexity. Furthermore, Theorem 3 applies to all compact and Σ-repeatable online problems, not just those which can be modeled as a MTS.

It is natural to ask if it is possible to use the technique of Blum and Burch in order to obtain a constructive proof of Theorem 3. To this end, we remark that the result of Blum and Burch relies fundamentally on the fact that the cost incurred by RWM when switching from the state of algorithm A_i to the state of algorithm A_j for $i \neq j$ is bounded by a constant. Thus, it does not seem possible to extend the result to those compact and Σ-repeatable problems which does not satisfy this requirement (such as the metric matching problem). On the other hand, in the full paper [42], we show that by combining the result of Blum and Burch with the ideas that we use to prove that the MTS problem is compact, it is possible to use a variant of RWM to algorithmically convert a c-competitive MTS-algorithm with advice complexity $o(n)$ (instead of just $O(1)$) into a randomized $(c + \varepsilon)$-competitive algorithm without advice. This yields a constructive version of Theorem 3 for problems that can be modeled as a MTS.

Finally, we note that very shortly after and independently of our work, Böckenhauer et al. considered applications of machine learning algorithms to the advice complexity model [14]. In [14], the authors present and analyze an algorithm called Shrinking Dartboard which is similar to the algorithm of Blum and Burch (both algorithms are based on RWM).

4 The computational model

We start by formally defining competitive analysis and advice complexity. Since we are very much interested in sublinear advice, we will use the *advice-on-tape model* [17, 34]. In this model, the algorithm is allowed to read an arbitrary number of advice bits from an advice tape. There is an alternative model, the *advice-with-request model* [29], where a fixed number of advice bits is provided along with each request.

▶ **Definition 7** (Advice complexity [17, 34] and competitive ratio [18])**.** The input to an online problem, P, is a sequence $\sigma = (s, x_1, \ldots, x_n)$. We say that s is the *initial state* and $x_1, \ldots, x_n$ are the *requests*. A *deterministic online algorithm with advice*, `ALG`, computes the output $\gamma = (y_1, \ldots, y_n)$, under the constraint that y_i is computed from $\varphi, s, x_1, \ldots, x_i$, where φ is the content of the advice tape. The *advice complexity*, $b(n)$, of `ALG` is the largest number of bits of φ read by `ALG` over all possible inputs of length at most n.

For an input σ, $\texttt{ALG}(\sigma)$ ($\texttt{OPT}(\sigma)$) denotes the non-negative real *cost* of the output computed by $\texttt{ALG}$ ($\texttt{OPT}$) when serving σ. We say that $\texttt{ALG}$ is *c-competitive* if there exists a constant α such that $\texttt{ALG}(\sigma) \leq c \cdot \texttt{OPT}(\sigma) + \alpha$ for all inputs σ.

A *randomized online algorithm* $\texttt{R}$ *with advice complexity* $b(n)$ is a probability distribution over deterministic online algorithms with advice complexity at most $b(n)$. We say that $\texttt{R}$ is c-competitive if there exists a constant α such that $\mathbb{E}[\texttt{R}(\sigma)] \leq c \cdot \texttt{OPT}(\sigma) + \alpha$ for all inputs σ. △

5 An information theoretical direct product theorem

In this section, we formally state (and sketch how to prove) our direct product theorem upon which all of our results rely. Given P-inputs $\sigma_1, \ldots, \sigma_r$, we define $\sigma = \sigma_1 \ldots \sigma_r$ to be the P-input obtained by concatenating the requests of the r inputs and using the same initial state as σ_1. For example, if $\sigma_1 = (s, x_1, \ldots, x_n)$ and $\sigma_2 = (s', x'_1, \ldots, x'_{n'})$, then $\sigma_1\sigma_2 = (s, x_1, \ldots, x_n, x'_1, \ldots, x'_{n'})$.

▶ **Definition 8** (r-round input distribution)**.** Let P be a minimization problem and let $r \in \mathbb{N}$. For each $1 \leq i \leq r$, let I_i be a finite set of P-inputs such that the following holds: If $\sigma_1, \ldots, \sigma_r$ are such that $\sigma_i \in I_i$ for $1 \leq i \leq r$, then $\sigma = \sigma_1\sigma_2 \ldots \sigma_r$ is a valid P-input. Furthermore, let $I^r = I_1 \times \cdots \times I_r = \{\sigma_1 \ldots \sigma_i \ldots \sigma_r \mid \sigma_i \in I_i \text{ for } 1 \leq i \leq r\}$.

For each $1 \leq i \leq r$, let cost_i be a function which maps an output γ computed for an input $\sigma \in I^r$ to a non-negative real number $\text{cost}_i(\gamma, \sigma)$. We say that cost_i is the ith round *cost function.*

Let $p_i : I_i \to [0, 1]$ be a probability distribution over I_i and let $p^r : I^r \to [0, 1]$ be the (product) probability distribution which maps $\sigma_1\sigma_2 \ldots \sigma_r \in I^r$ into $p_1(\sigma_1)p_2(\sigma_2) \cdots p_r(\sigma_r)$. We say that p^r (together with the associated cost functions cost_i) is an *r-round input distribution.* For $1 \leq i \leq r$, we say that p_i is the ith round input distribution of p^r. △

▶ **Definition 9.** Let P be a minimization problem and let $r \in \mathbb{N}$. Let $p^r : I^r \to [0, 1]$ be an r-round input distribution with associated cost functions cost_i and with ith round input distributions p_i. Let $\texttt{ALG}$ be a deterministic P-algorithm with advice. We define the following random variables: Let X be the entire input and Y the output computed by $\texttt{ALG}$. For $1 \leq i \leq r$, let X_i be the requests revealed in round i, and let $\text{cost}_i(\texttt{ALG}) = \text{cost}_i(Y, X)$ be the ith round cost function applied to the output computed by $\texttt{ALG}$. Also, let $\text{cost}(\texttt{ALG}) = \sum_{i=1}^{r} \text{cost}_i(\texttt{ALG})$, let B be the advice bits read by $\texttt{ALG}$, and let $W_i = (X_1, \ldots, X_{i-1}, B)$. Random variables are always denoted by capital letters and their support by the calligraphic version of that letter[2].

Finally, for every $1 \leq i \leq r$ and every $w \in \mathcal{W}_i$, we define the conditional ith round input distribution $p_{i|w} : I_i \to [0, 1]$ as follows:

$$p_{i|w}(x) = p_i(x|W_i = w) = \frac{\Pr(X_i = x, W_i = w)}{\Pr(W_i = w)}.$$

△

We prove our information theoretical direct product theorem by extending an entropy based lower bound technique due to Emek et al. [29]. The statement and proof of the theorem relies on basic notions and results from information theory (see [42] or [26]). In particular, given two distributions $\mu, \nu : \Omega \to [0, 1]$ such that $\text{supp}(\mu) \subseteq \text{supp}(\nu)$, the *Kullback-Leibler*

[2] For example, the support of W_i is denoted $\mathcal{W}_i$, where $\mathcal{W}_i = \{w\colon \Pr[W_i = w] > 0\}$.

divergence $D_{KL}(\mu\|\nu)$ between μ and ν is defined as $D_{KL}(\mu\|\nu) = \sum_{\omega\in\Omega} \mu(\omega)\log(\mu(\omega)/\nu(\omega))$. Also, we heavily use the notation of Definitions 8 and 9.

▶ **Theorem 10.** *Let* P *be a minimization problem and let* p^r *be an* r*-round input distribution with associated cost-functions* cost_i*. Furthermore, let* ALG *be a deterministic algorithm reading at most* b *bits of advice on every input in the support of* p^r*. Assume that there exists a convex and decreasing function* $f : [0,\infty] \to \mathbb{R}$ *such that for every* $1 \leq i \leq r$ *and* $w \in \mathcal{W}_i$*, the following holds:*

$$\mathbb{E}[\text{cost}_i(\texttt{ALG})|W_i = w] \geq f\left(D_{KL}(p_{i|w}\|p_i)\right). \tag{3}$$

Then, $\mathbb{E}[\text{cost}(\texttt{ALG})] \geq rf(b/r)$.

Before sketching the proof of Theorem 10, we informally discuss the theorem. Recall that W_i is the information available to the algorithm when round i begins (the advice read and the history of previous requests). Without any advice or knowledge of the history, the probability of $x \in \mathcal{X}_i$ being selected as the round i request sequence is $p_i(x)$. However, this probability may change given that the algorithm knows W_i (in the most extreme case, the advice could specify exactly the request sequence in round i). For any fixed $w \in \mathcal{W}_i$, the probability of x being selected in round i given that $W_i = w$ is denoted $p_{i|w}(x)$. Assumption (3) informally means that the closer $p_{i|w}$ and p_i are to each other, the better a lower bound we must have on the expected cost incurred by ALG in round i given that ALG knows w. We remark that the convexity assumption on f is automatically satisfied in most applications. The conclusion of Theorem 10 essentially says that under these assumptions, an algorithm with b bits of advice for the entire input can do no better than an algorithm with b/r bits of advice for each individual round. In particular, this will allow us to conclude that $\Omega(r)$ bits of advice are needed to get a non-trivial improvement over having no advice at all. The complete proof of Theorem 10 can be found in the full version of the paper [42].

Proof Sketch (of Theorem 10). Fix i such that $1 \leq i \leq r$. Using first the law of total expectation and then combining assumption (3) with Jensen's inequality, we get that

$$\mathbb{E}[\text{cost}_i(\texttt{ALG})] = \mathbb{E}_w[\mathbb{E}[\text{cost}_i(\texttt{ALG})|W_i = w]] \geq f\left(\mathbb{E}_w[D_{\text{KL}}(p_{i|w}\|p_i)]\right). \tag{4}$$

A simple calculation shows that the expected Kullback-Leibler divergence $\mathbb{E}_w[D_{\text{KL}}(p_{i|w}\|p_i)]$ equals the mutual information $I(X_i;W_i)$ between X_i and W_i. Thus,

$$\mathbb{E}[\text{cost}(\texttt{ALG})] = \sum_{i=1}^{r} \mathbb{E}[\text{cost}_i(\texttt{ALG})] \geq \sum_{i=1}^{r} f(I(X_i;W_i))$$

The mutual information $I(X;B)$ between the input X and the advice B is at most b simply because the entropy $H(B)$ of B is at most b (since ALG reads at most b bits of advice). On the other hand, using that the r rounds are independent and the chain rule of conditional entropy, we get that $I(X;B) = \sum_{i=1}^{r} I(X_i;W_i)$. Combining these two observations, we see that $I(X_1;W_1),\ldots,I(X_r,W_r)$ are non-negative real numbers which sums to at most b. Since f is convex and decreasing, Jensen's inequality then implies that $\sum_{i=1}^{r} f(I(X_i;W_i)) \geq rf(b/r)$. ◀

6 Application: Repeatable online problems

In this section we will present the main ingredients of the proof of Theorem 1. Before we begin, we need to formally define the repeated version, P^*_Σ, of an online problem P, and

we need to define precisely what it means to be Σ-repeatable. To this end, given P-inputs $\sigma_1, \ldots, \sigma_r$ with the same initial state s, we define $(\sigma_1; \ldots; \sigma_r)$ to be a sequence with the requests of the r inputs concatenated and such that the initial state s arrives together with the first request of each σ_i. For example, if $\sigma_1 = (s, x_1, \ldots, x_n)$ and $\sigma_2 = (s, x'_1, \ldots, x'_{n'})$, then $(\sigma_1; \sigma_2) = (s, \{s, x_1\}, x_2, \ldots, x_n, \{s, x'_1\}, x'_2, \ldots, x'_{n'})$.

▶ **Definition 11.** Let P be an online problem, let S be the set of initial states for P, and let I (I_s) be the set of all possible request sequences for P (with initial state s). Define P^*_Σ to be the online problem with inputs $I^* = \{\sigma^* = (\sigma_1; \sigma_2; \ldots; \sigma_r) \mid r \geq 1, s \in S, \sigma_i \in I_s\}$. An algorithm for P^*_Σ must produce an output $\gamma^* = (\gamma_1, \ldots, \gamma_r)$ where $\gamma_i = (y_1, \ldots, y_{n_i})$ is a valid sequence of answers for the P-input $\sigma_i = (s, x_1, \ldots, x_{n_i}) \in I_s$. The score of the output γ^* is $\mathrm{score}(\gamma^*, \sigma^*) = \sum_{i=1}^{r} \mathrm{score}_P(\gamma_i, \sigma_i)$, where $\mathrm{score}_P(\gamma_i, \sigma_i)$ is the score of the P-output γ_i with respect to the P-input σ_i. The optimal offline algorithm for P^*_Σ is denoted $\mathtt{OPT}^*_\Sigma$. △

$\mathrm{P}^*_\vee$ is defined similarly, except that $\mathrm{score}(\sigma^*, \gamma^*) = \max\{\mathrm{score}_P(\sigma_1, \gamma_1), \ldots, \mathrm{score}_P(\sigma_r, \gamma_r)\}$.

In order to better understand the definition of P^*_Σ, it is useful to imagine that after serving the last request of round $i-1$ but before serving the first request of round i, the current state is changed to the initial state s (note that the algorithm knows when this happens since in $(\sigma_1; \ldots; \sigma_r)$, the first request of each σ_i is special). For instance, if P is the k-server problem, then an initial state s is a placement of the k servers in the metric space. Thus, in P^*_Σ, after serving the last request of round $i-1$ and before serving the first request of round i, the k servers automatically return to their initial position specified by s. Note that when P is the k-server problem, we can concatenate P-inputs $\sigma_1, \sigma_2, \ldots, \sigma_r$ into one long P-input $\sigma = \sigma_1\sigma_2 \ldots \sigma_r$. The only difference between serving the P-input σ and serving the P^*_Σ-input $\sigma^* = (\sigma_1; \ldots; \sigma_r)$ is that for the P-input σ, the k servers are not returned to the initial state s when a round ends. However, if the underlying metric space has finite diameter, this difference in the placement of servers when a new round begins cannot invalidate a lower bound. In fact, it turns out that for many online problems, there is a natural reduction from P^*_Σ to P that essentially preserves all lower bounds. This is formalized in Definition 12.

▶ **Definition 12.** Let P be an online minimization problem such that for every fixed P-input, there is only a finite number of valid outputs. Furthermore, let $k_1, k_2, k_3 \geq 0$. We say that P is Σ-*repeatable with parameters* (k_1, k_2, k_3) if there exists a mapping $g : I^* \to I$ with the following properties:

Σ1. $|g(\sigma^*)| \leq |\sigma^*| + k_1 r$ for every $\sigma^* \in I^*$ consisting of r rounds.

Σ2. For every deterministic P-algorithm ALG, there exists a deterministic P^*_Σ-algorithm $\mathtt{ALG}^*$ such that $\mathtt{ALG}^*(\sigma^*) \leq \mathtt{ALG}(g(\sigma^*)) + k_2 r$ for every $\sigma^* \in I^*$ consisting of r rounds.

Σ3. $\mathtt{OPT}^*_\Sigma(\sigma^*) \geq \mathtt{OPT}(g(\sigma^*)) - k_3 r$ for every $\sigma^* \in I^*$ consisting of r rounds. △

The definition of $\vee$-repeatable is identical to that of Σ-repeatable, except that P^*_Σ and $\mathtt{OPT}^*_\Sigma$ are replaced by $\mathrm{P}^*_\vee$ and $\mathtt{OPT}^*_\vee$. The k-server problem on a metric space of diameter Δ is Σ-repeatable with parameters $(0, k\Delta, k\Delta)$.

We will now sketch the proof of Theorem 1. The complete proof is in the full paper [42].

Proof Sketch (of Theorem 1). We prove the desired lower bound for P^*_Σ. Let ε and α be arbitrary. Without loss of generality, we may assume that all inputs in the support of $p_{\alpha,\varepsilon}$ have the same initial state. Draw r inputs independently from $p_{\alpha,\varepsilon}$. This gives rise to an r-round input distribution $p^r_{\alpha,\varepsilon}$ for P^*_Σ. Note that the ith round input distribution, p_i, of $p^r_{\alpha,\varepsilon}$ is simply $p_i = p_{\alpha,\varepsilon}$, and that a round of $p^r_{\alpha,\varepsilon}$ corresponds to a round of P^*_Σ.

Let $\texttt{ALG}^*$ be a deterministic P^*_Σ-algorithm reading at most b bits of advice on inputs in $\mathrm{supp}(p^r_{\alpha,\varepsilon})$. Fix a round $1 \leq i \leq r$ and let $w \in \mathcal{W}_i$. We need to give a lower bound on the expected cost $\mathbb{E}[\mathrm{cost}_i(\texttt{ALG}^*)|W_i = w]$ in terms of the Kullback-Leibler divergence between $p_{i|w}$ and p_i. To this end, we use *Pinsker's inequality*. Pinsker's inequality ([26, Lemma 11.6.1]) says this if the Kullback-Leibler divergence between $p_{i|w}$ and p_i is small, then $p_{i|w}$ and p_i must be close in L_1-norm: $\|p_{i|w} - p_i\|_1 \leq \sqrt{\ln 4 \cdot D_{\mathrm{KL}}(p_{i|w}\|p_i)}$. Thus, it suffices to bound $\mathbb{E}[\mathrm{cost}_i(\texttt{ALG}^*)|W_i = w]$ in terms of the L_1-distance between $p_{i|w}$ and p_i.

To this end, we construct a P-algorithm, $\texttt{ALG}_w$, without advice by hard-wiring w (i.e., the advice and the requests in rounds 1 to $i-1$) into the P^*_Σ-algorithm $\texttt{ALG}^*$. That is, for an input sequence $\sigma \in \mathrm{supp}(p_{i|w})$, the new algorithm $\texttt{ALG}_w$ simulates the computation of $\texttt{ALG}^*$ on σ when $W_i = w$ and $\texttt{ALG}^*$ is given the requests σ in round i. Note that this is possible since w is fixed, and hence the output of $\texttt{ALG}^*$ in round i given σ as input in this round is completely determined. For all other input sequences, $\texttt{ALG}_w$ behaves arbitrarily (but does compute some valid output). It follows that $\texttt{ALG}_w$ is well-defined for all input sequences in $\mathrm{supp}(p_i)$. Thus, $\texttt{ALG}_w$ defines a mapping $\sigma \mapsto \texttt{ALG}_w(\sigma)$ on $\mathrm{supp}(p_i)$ such that $\|\texttt{ALG}_w\|_\infty = \max_{\sigma \in \mathrm{supp}(p_i)} |\texttt{ALG}_w(\sigma)| \leq M$ and such that if $\sigma \in \mathrm{supp}(p_{i|w}) \subseteq \mathrm{supp}(p_i)$, then $\texttt{ALG}_w(\sigma)$ is equal to the cost incurred by $\texttt{ALG}^*$ if $W_i = w$ and σ is given as input in round i. It follows that

$$\begin{aligned}\mathbb{E}[\mathrm{cost}_i(\texttt{ALG})|W_i = w] &= \mathbb{E}_{\sigma \sim p_{i|w}}[\texttt{ALG}_w(\sigma)] \geq \mathbb{E}_{\sigma \sim p_i}[\texttt{ALG}_w(\sigma)] - \|\texttt{ALG}_w\|_\infty \cdot \|p_{i|w} - p_i\|_1 \\ &\geq (c-\varepsilon)\,\mathbb{E}_{\sigma \sim p_i}[\texttt{OPT}(\sigma)] + \alpha - M \cdot \sqrt{\ln 4 \cdot D_{\mathrm{KL}}(p_{i|w}\|p_i)}.\end{aligned}$$

Let $f(d) = (c-\varepsilon)\,\mathbb{E}_{\sigma \sim p_i}[\texttt{OPT}(\sigma)] + \alpha - M \cdot \sqrt{\ln 4 \cdot d}$. Note that f is convex. Our direct product theorem (Theorem 10) therefore yields (remember that $p_i = p_{\alpha,\varepsilon}$).

$$\mathbb{E}[\mathrm{cost}(\texttt{ALG}^*)] \geq r f(b/r) \geq r\left((c-\varepsilon)\,\mathbb{E}_{\sigma \sim p_{\alpha,\varepsilon}}[\texttt{OPT}(\sigma)] + \alpha - 2M\sqrt{\frac{b}{r}}\right). \tag{5}$$

Assume that $\texttt{ALG}^*$ uses sublinear advice, i.e., $b = o(n)$. Note that $n = \Theta(r)$ since $\mathrm{supp}(p_{\alpha,\varepsilon})$ is finite. Thus, we can make $2M\sqrt{b/r}$ arbitrarily small by choosing the number of rounds r sufficiently large. Furthermore, by linearity of expectation, $r\,\mathbb{E}_{\sigma \sim p_{\alpha,\varepsilon}}[\texttt{OPT}(\sigma)] = \mathbb{E}[\mathrm{cost}(\texttt{OPT}^*_\Sigma)]$. Thus, since ε and α was arbitrary, it follows from (5) and Yao's principle that a randomized P^*_Σ-algorithm with sublinear advice must have a competitive ratio of at least c. Using that P is Σ-repeatable, it is possible to convert this lower bound into a lower bound for P. ◀

7 Application: Compact online problems

Recall (Definition 2) that we defined compact online problems to be those for which Theorem 1 could be used to obtain tight lower bounds. Theorem 3 follows trivially by combining Definition 2 with Theorem 1 (and using previously known derandomization results). In this section, we will sketch how to prove that several important Σ-repeatable online problems are compact. Interestingly, our proof will rely on the "upper bound part" of Yao's principle [48] which is (much) less frequently used than the lower bound part.

Fix a Σ-repeatable problem P such that for every n (and every fixed initial state), the number of inputs of length at most n is finite[3]. Let $c > 1$ be a constant such that the expected

[3] Since P is Σ-repeatable, this assumption automatically implies that there is only finitely many P-algorithms for inputs of length at most n.

competitive ratio of every randomized P-algorithm is at least c. What does it mean for P to *not* be compact? It means that there exists an $\varepsilon > 0$ and $\alpha \geq 0$ such that the following holds: For every $n' \in \mathbb{N}$ and for every probability distribution p over P-inputs of length at most n', there exists a deterministic algorithm $\mathtt{D}$ such that $\mathbb{E}_{\sigma\sim p}[\mathtt{D}(\sigma)] < (c-\varepsilon)\,\mathbb{E}_{\sigma\sim p}[\mathtt{OPT}(\sigma)] + \alpha$. Recall that, by assumption, there is only a finite number of inputs and outputs for P of length at most n'. This makes it possible to view the problem as a *finite* two-player zero-sum game between an algorithm and adversary. Thus (see the full paper [42] for a formal proof), by Yao's principle, we get that there exists a randomized P-algorithm $\mathtt{R}_{n'}$ such that $\mathbb{E}[\mathtt{R}_{n'}(\sigma)] < (c-\varepsilon)\mathtt{OPT}(\sigma)+\alpha$ for every P-input of length at most n'. Now, if we can somehow show that it is possible to use the algorithms $\mathtt{R}_1, \mathtt{R}_2, \ldots$ to obtain a single algorithm $\mathtt{R}$ which is better than c-competitive (on all possible inputs), then the problem at hand must, by contradiction, be compact.

From the above discussion, it is easy to see that the metric matching problem is compact (on finite metric spaces). Indeed, in this problem, there are k servers placed in a metric space. Each server can be matched to at most one request (and vice versa). But this means that for every fixed metric space and fixed set of k servers, the length of the input never exceeds k. This allow us to construct a single algorithm $\mathtt{R}$ which on inputs with k servers run the appropriate algorithm $\mathtt{R}_k$. The algorithm $\mathtt{R}$ will be $(c-\varepsilon)$-competitive. This argument extends to all Σ-repeatable problems for which an online algorithm knows a priori some upper bound on the number of requests.

Another important example is the k-server problem on a finite metric space. In this problem, we also have a metric space and a set of k servers. However, for the k-server problem, there is no bound on the number of times we may use a single server. Thus, the length of the input is unbounded, even if we fix the metric space and the set of servers. This means that we cannot just trivially run the appropriate algorithm $\mathtt{R}_{n'}$, since we do not know an upper bound n' on the length of the input. However, what we can do is the following (see the full paper and [4, 44]): Choose n' to be some sufficiently large number. If the number of requests exceeds n', we *restart* $\mathtt{R}_{n'}$. That is, we use a new instantiation of $\mathtt{R}_{n'}$ which pretends that the $(n'+1)$'th request is in fact the very first request of the input sequence. By appropriately handling some technical issues, we can make sure that the price of performing these restarts is sufficiently small compared to the cost of an optimal solution. This gives a single algorithm $\mathtt{R}$ which is better than c-competitive on inputs of arbitrary length. Thus, the k-server problem is compact. This technique works for all problems that can be modeled as a MTS. We refer to the full version of the paper for more details.

8 Further work and open problems

We have attempted to make the results and techniques introduced in this paper as easy as possible to apply and build on. Komm et al. have used our results on repeated matrix games to prove lower bounds for certain online hereditary graph problems with preemption [38]. Also, in Boyar et al. [21], we have applied Theorem 1 to online weighted matching.

Currently, the equivalence between advice and randomization stated in Theorem 3 is mainly being used to obtain knowledge about algorithms with advice using techniques and results for randomized algorithms. It is an interesting open problem to what extent the equivalence is also useful in the other direction.

Acknowledgment. The author thanks Joan Boyar, Magnus Find, Lene Favrholdt, and the anonymous ICALP reviewers for helpful comments on this work and its presentation.

References

1. Anna Adamaszek, Artur Czumaj, Matthias Englert, and Harald Räcke. Almost tight bounds for reordering buffer management. In *STOC*, 2011.
2. Anna Adamaszek, Marc P. Renault, Adi Rosén, and Rob van Stee. Reordering buffer management with advice. In *WAOA*, 2013.
3. Susanne Albers and Matthias Hellwig. Online makespan minimization with parallel schedules. In *SWAT*, 2014.
4. Christoph Ambühl. *On the list update problem.* PhD thesis, ETH Zürich, 2002.
5. Spyros Angelopoulos, Christoph Dürr, Shahin Kamali, Marc P. Renault, and Adi Rosén. Online bin packing with advice of small size. In *WADS*, 2015.
6. Yossi Azar, Andrei Z. Broder, and Mark S. Manasse. On-line choice of on-line algorithms. In *SODA*, 1993.
7. Yossi Azar, Iftah Gamzu, and Ran Roth. Submodular max-sat. In *ESA*, 2011.
8. Amotz Bar-Noy, Rajeev Motwani, and Joseph (Seffi) Naor. The greedy algorithm is optimal for on-line edge coloring. *Inf. Process. Lett.*, 44(5):251–253, 1992.
9. Dwight R. Bean. Effective coloration. *J. Symb. Log.*, 41(2):469–480, 1976.
10. Maria Paola Bianchi, Hans-Joachim Böckenhauer, Juraj Hromkovič, and Lucia Keller. Online coloring of bipartite graphs with and without advice. *Algorithmica*, 70(1):92–111, 2014.
11. Maria Paola Bianchi, Hans-Joachim Böckenhauer, Juraj Hromkovič, Sacha Krug, and Björn Steffen. On the advice complexity of the online $L(2,1)$-coloring problem on paths and cycles. *Theor. Comput. Sci.*, 554:22–39, 2014.
12. Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, 2000.
13. Hans-Joachim Böckenhauer, Richard Dobson, Sacha Krug, and Kathleen Steinhöfel. On energy-efficient computations with advice. In *COCOON*, 2015.
14. Hans-Joachim Böckenhauer, Sascha Geulen, Dennis Komm, and Walter Unger. Constructing randomized online algorithms from algorithms with advice. *ETH-Zürich*, 2015.
15. Hans-Joachim Böckenhauer, Juraj Hromkovič, Dennis Komm, Sacha Krug, Jasmin Smula, and Andreas Sprock. The string guessing problem as a method to prove lower bounds on the advice complexity. *Theor. Comput. Sci.*, 554:95–108, 2014.
16. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, and Richard Královič. On the advice complexity of the k-server problem. In *ICALP (1)*, 2011.
17. Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, and Tobias Mömke. On the advice complexity of online problems. In *ISAAC*, 2009.
18. Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.
19. Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.
20. Joan Boyar, Lene M. Favrholdt, Christian Kudahl, and Jesper W. Mikkelsen. Advice complexity for a class of online problems. In *STACS*, 2015.
21. Joan Boyar, Lene M. Favrholdt, Christian Kudahl, and Jesper W. Mikkelsen. Weighted online problems with advice. In submission, 2016.
22. Joan Boyar, Shahin Kamali, Kim S. Larsen, and Alejandro López-Ortiz. On the list update problem with advice. In *LATA*, 2014.
23. Joan Boyar, Shahin Kamali, Kim S. Larsen, and Alejandro López-Ortiz. Online bin packing with advice. In *STACS*, 2014. Full paper to appear in *Algorithmica*.
24. Carl Burch. *Machine learning in metrical task systems and other on-line problems.* PhD thesis, Carnegie Mellon University, 2000. http://cburch.com/pub/thesis.ps.gz.

25 Marek Chrobak, Lawrence L. Larmore, Carsten Lund, and Nick Reingold. A better lower bound on the competitive ratio of the randomized 2-server problem. *Inf. Process. Lett.*, 63(2):79–83, 1997.

26 Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley, 2006.

27 Erik D. Demaine, Dion Harmon, John Iacono, and Mihai Patrascu. Dynamic optimality – almost. *SIAM J. Comput.*, 37(1):240–251, 2007.

28 Stefan Dobrev, Rastislav Královič, and Dana Pardubská. Measuring the problem-relevant information in input. *RAIRO – Theor. Inf. Appl.*, 43(3):585–613, 2009.

29 Yuval Emek, Pierre Fraigniaud, Amos Korman, and Adi Rosén. Online computation with advice. *Theor. Comput. Sci.*, 412(24):2642–2656, 2011.

30 Leah Epstein and Rob van Stee. On the online unit clustering problem. *ACM Transactions on Algorithms*, 7(1):1–18, 2010.

31 Amos Fiat, Dean P. Foster, Howard J. Karloff, Yuval Rabani, Yiftach Ravid, and Sundar Vishwanathan. Competitive algorithms for layered graph traversal. *SIAM J. Comput.*, 28(2):447–462, 1998.

32 Michal Forišek, Lucia Keller, and Monika Steinová. Advice complexity of online graph coloring. Unpublished manuscript, 2012.

33 Magnús M. Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring. *Theor. Comput. Sci.*, 130(1):163–174, 1994.

34 Juraj Hromkovič, Rastislav Královič, and Richard Královič. Information complexity of online problems. In *MFCS*, 2010.

35 Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993.

36 Anna R. Karlin, Mark S. Manasse, Lyle A. McGeoch, and Susan S. Owicki. Competitive randomized algorithms for non-uniform problems. In *SODA*, 1990.

37 Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, 1990.

38 Dennis Komm, Rastislav Královic, Richard Královic, and Christian Kudahl. Advice complexity of the online induced subgraph problem. *CoRR*, abs/1512.05996, 2015.

39 Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2):105–118, 2009.

40 Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.

41 Jesper W. Mikkelsen. Optimal online edge coloring of planar graphs with advice. In *CIAC*, 2015.

42 Jesper W. Mikkelsen. Randomization can be as helpful as a glimpse of the future in online computation. *CoRR*, abs/1511.05886, November 2015.

43 Shuichi Miyazaki. On the advice complexity of online bipartite matching and online stable marriage. *Inf. Process. Lett.*, 114(12):714–717, 2014.

44 Tobias Mömke. A competitive ratio approximation scheme for the k-server problem in fixed finite metrics. *CoRR*, abs/1303.2963, 2013.

45 Marc P. Renault and Adi Rosén. On online algorithms with advice for the k-server problem. *Theory Comput. Syst.*, 56(1):3–21, 2015.

46 Jasmin Smula. *Information Content of Online Problems, Advice versus Determinism and Randomization*. PhD thesis, ETH Zürich, 2015.

47 Boris Teia. A lower bound for randomized list update algorithms. *Inf. Process. Lett.*, 47(1):5–9, 1993.

48 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *FOCS*, 1977.

Online Semidefinite Programming*

Noa Elad[1], Satyen Kale[2], and Joseph (Seffi) Naor[3]

1 **Computer Science Dept., Technion, Haifa, Israel**
noako@cs.technion.ac.il
2 **Yahoo Research, New York, NY, USA**
satyen@yahoo-inc.com
3 **Computer Science Dept., Technion, Haifa, Israel**
naor@cs.technion.ac.il

Abstract

We consider semidefinite programming through the lens of online algorithms – what happens if not all input is given at once, but rather iteratively? In what way does it make sense for a semidefinite program to be revealed? We answer these questions by defining a model for online semidefinite programming. This model can be viewed as a generalization of online covering-packing linear programs, and it also captures interesting problems from quantum information theory. We design an online algorithm for semidefinite programming, utilizing the online primal-dual method, achieving a competitive ratio of $O(\log n)$, where n is the number of matrices in the primal semidefinite program. We also design an algorithm for semidefinite programming with box constraints, achieving a competitive ratio of $O(\log F^*)$, where F^* is a sparsity measure of the semidefinite program. We conclude with an online randomized rounding procedure.

1998 ACM Subject Classification G.1.6 Optimization

Keywords and phrases online algorithms, semidefinite programming, primal-dual

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.40

1 Introduction

The study of online algorithms is a major theme in computer science. In contrast to an algorithm that receives all its input at once, an online algorithm receives its input in an iterative manner, and must make irrevocable decisions after receiving each part of the input. An online algorithm is evaluated based on the ratio between its cost and the optimal offline cost, that is, the cost which could have been achieved had the entire input sequence been known in advance. The worst-case bound on this ratio is called the *competitive ratio* of the algorithm. Competitive analysis of online algorithms is a very active area of research and the last twenty five years have witnessed many exciting new results. For a broad study of online algorithms, see [2, 3].

Online set cover is a classical online problem; elements arrive one by one, specifying which sets they belong to, and each element must be covered upon arrival. Once a set is chosen to the cover it cannot be removed later on, and its cost is accumulated into the final cost. The objective is to minimize the total cost of the chosen sets. A natural extension is online fractional set cover, where sets are associated with fractional values, and for each element the sum of the fractions of the sets containing it is at least 1. Irrevocability is expressed by requiring the fractional values to be monotonically non-decreasing during the online steps.

* This work was partially supported by ISF grant 1585/15 and BSF grant 2014414.

Set cover further generalizes into general *covering* problems. In this setting, an element is not simply contained in a set, but rather a non-negative weight $a_{e,S}$ is associated with each element-set pair. This is actually equivalent to a linear program where the entries of the constraint matrix are all non-negative. The objective function is also non-negative. Covering problems arise naturally in many settings, e.g., graph optimization problems, facility location and paging. The dual problem of covering is *packing* which also captures an important family of combinatorial problems, e.g., flow, routing, matchings, and combinatorial auctions.

The online primal-dual method is a powerful algorithmic technique that has proved to be extremely useful for a wide variety of problems in the area of online algorithms, serving as an important unifying design methodology. In general, suppose that an online problem can be formulated as a linear program, such that requests correspond to linear constraints. The online nature of the problem translates into the constraint matrix being exposed row by row, while the dual program is updated by adding a new dual variable. The idea behind the primal-dual method is to use both primal and dual programs and construct (simultaneously) feasible solutions for both of them, while maintaining some relationship between their corresponding objective functions. Then, using weak duality, it is easy to bound the competitive factor of the primal solution.

For online covering-packing linear programs, the non-negativity of the constraint matrix facilitates the design of elegant online primal-dual algorithms. This approach has been amazingly successful in both unifying previously known results, as well as resolving several important open questions in competitive analysis. This includes, among others, the classic ski rental problem, the online set-cover problem, paging and weighted paging, graph optimization problems, the dynamic TCP-acknowledgement problem, various routing problems, load balancing, machine scheduling, ad auctions, and more. Please refer for more details to a survey on the covering-packing approach to online algorithms [3].

We explore in this work semidefinite programming in the context of online problems. A symmetric matrix $A \in \mathbb{R}^{m \times m}$ is said to be positive semidefinite, i.e., $A \succcurlyeq 0$, if for every vector $v \in \mathbb{R}^m$, it holds that $v^t A v \geq 0$. A semidefinite program has a constraint requiring that a matrix of variables $X = (x_{i,j})$ is positive semidefinite. Such constraints correspond to an unbounded number of linear constraints, since $X \succcurlyeq 0$ is equivalent to a family of linear constraints: $\forall v \in \mathbb{R}^n \; v^t X v = \sum v_i v_j x_{ij} \geq 0$. This greatly extends our ability to express complex problems, since semidefinite programs can be tailored more specifically, thus decreasing integrality gaps of linear relaxations of combinatorial problems and yielding tighter approximation factors. In their seminal work, Goemans and Williamson [4] used a semidefinite programming relaxation for max cut in undirected graphs, obtaining an approximation factor of 0.878, dramatically improving over the straightforward factor of 0.5.

1.1 Results

We study here further extensions of the successful online primal-dual method and linear programming relaxations for online problems to the realm of semidefinite programming. We define the problem of semidefinite covering-packing, which is a semidefinite program in which all coefficients are non-negative. For matrix coefficients, such as those in constraints of the form $A \bullet X \leq c$, the coefficient matrix A is positive-semidefinite. This problem arises in several natural settings, e.g., quantum hypergraph covering, studied by [1], is a semidefinite covering problem with all matrices restricted to the range $[0, I]$. Our work extends semidefinite covering-packing to an online setting. For example, our extension captures the online covering problem as a special case, in which all matrices are diagonal.

We design an online primal-dual algorithm with competitive ratio $O(\log n)$ for the semidefinite covering-packing setting, where n denotes the number of matrices (e.g., corresponding

to sets in the case of set cover). This is called our *general algorithm* (Section 3). We then consider in Section 4 a class of semidefinite covering-packing problems with *box constraints* on the variables. We design a primal-dual algorithm with an improved bound of $O(\log F^*)$, where F^* is a sparsity parameter coinciding with the maximum row-sparsity D of the constraint matrix in the linear case. This bound can be compared to the $O(\log D)$-competitive algorithm known for covering in the linear case [3, 5].

In Section 5 we design an online randomized rounding procedure, which is applicable to both of our algorithms, while adding a factor of $O(R(\log m + \log n))$ to the solution, where m is the dimension of the matrices and R is a bound on the largest eigenvalue of each matrix. Our randomized rounding uses a general matrix Chernoff bound due to Tropp [6], which is based on and improves slightly the novel matrix Chernoff bound developed by Ahlswede and Winter [1]. In a way, our algorithm is an online variation to Wigderson and Xiaos's [7] randomized rounding algorithm for integer semidefinite programming.

Techniques. Our algorithms and proofs strongly utilize the online primal-dual technique, thus taking advantage of semidefinite weak duality and the fact that semidefinite programming is defined over a cone. Additionally, we use the fact that a semidefinite constraint of the form $A \succcurlyeq B$ can be expressed as an infinite number of linear constraints, each corresponding to a projection onto a vector v satisfying $v^T A v \geq v^T B v$.

The heart of our general algorithm is an update step: when considering a primal semidefinite constraint, it is either already satisfied (and then we are done), or there exists a matrix (in fact, many) which is a witness to the violation of the primal constraint, also inducing a linear constraint which is violated. The witness matrix is then used for updating the primal constraint, and for the direction in which we are going to make dual progress. Once the witness matrix is determined, our update rule becomes quite similar to the update rule in the linear case [3]. An important difference though is that, while only a finite number of (linear) constraints needs to be satisfied in the linear case, now we need to satisfy a semidefinite constraint, equivalent to infinitely many linear constraints. We address this issue by over-satisfying constraints, so as to ensure that we make enough progress for our choice of witness matrix, thus bounding the number of steps needed. Without this over-satisfaction, we might only make infinitesimal progress, and it is not clear if the algorithm ever terminates.

For box constraints, we define the *sparsity* of an online semidefinite program, a measure capturing the potential to overshoot when solving a sub-problem of a semidefinite program. While the definition also coincides with the notion of row-sparsity for linear programs, it is not a simple extension, but rather one which is tailored to a very specific observation about our update rule; each subset of the variables defines such a subproblem when those variables are saturated (i.e. their value equals the upper bound). In this situation we want to choose a good progress direction, so that we do not over-satisfy the semidefinite constraint. The progress direction turns out to have the property that, when projecting the problem onto that direction, minimizes the ratio between the uncovered coefficients and the remainder which is left to cover. This proves to be very useful in bounding the primal-dual ratio, and also in allowing the algorithm to make meaningful progress.

2 Our Model and Definitions

Throughout this paper, all matrices are square, symmetric, real, and have dimension m. Given two matrices A, B, their *Frobenius product*, denoted $A \bullet B$, is defined as $A \bullet B = \sum_{i,j} A_{i,j} B_{i,j} = tr\left(A^T B\right)$. This is equivalent to standard inner product if the matrices are

treated as vectors in $\mathbb{R}^{m^2}$. For any vector $v \in R^n$, it is easy to check that the following useful identity holds: $A \bullet \left(vv^T\right) = v^T A v = \sum_{i,j} v_i v_j A_{i,j}$.

The eigenvalues of a matrix A are $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_m(A)$. For a matrix A and vector v, the *Rayleigh quotient* is defined as $\frac{v^T A v}{v^T v}$ and has the property $\lambda_m(A) \leq \frac{v^T A v}{v^T v} \leq \lambda_1(A)$. A symmetric matrix A is said to be *positive-semidefinite* (p.s.d.), denoted $A \succcurlyeq 0$, if all its eigenvalues are non-negative. This induces a partial order over $\mathbb{R}^{m \times m}$, where $A \succcurlyeq B$ if and only if $A - B \succcurlyeq 0$.

A*semidefinite* minimization program in standard form and its dual program are:

$$\begin{array}{c} \min \sum_{i=1}^n c_i x_i \\ s.t.\ \sum_{i=1}^n A_i x_i \succcurlyeq B \\ \forall i \in [n]\ \ x_i \geq 0 \end{array} \qquad \begin{array}{c} \max B \bullet Y \\ s.t.\ \forall i \in [n]\ A_i \bullet Y \leq c_i \\ Y \succcurlyeq 0 \end{array} .$$

This pair of programs satisfies weak duality, e.g. every feasible solution of the primal problem is an upper bound on the dual problem and vice versa. This can readily be seen as follows. Let x and Y be feasible solutions for the primal and dual problems, respectively. Then

$$\sum_{i=1}^n c_i x_i \geq \sum_{i=1}^n (A_i \bullet Y) x_i = \left(\sum_{i=1}^n A_i x_i\right) \bullet Y \geq B \bullet Y.$$

The first inequality follows since $x_i \geq 0$ for all $i \in [n]$, and $A_i \bullet Y \leq c_i$. The second inequality is due to $\sum_{i=1}^n A_i x_i \succcurlyeq B$ and $Y \succcurlyeq 0$.

We sometimes use the shorthand $\mathcal{A}x$ to denote the matrix $\sum_{i=1}^n A_i x_i$, that is, $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^{m \times m}$ is a linear function defined as $\mathcal{A}(x) = \sum_{i=1}^n A_i x_i$.

2.1 Our Model

We define a *semidefinite covering problem* as a semidefinite program in which all scalar coefficients are non-negative ($c_i \geq 0$), and all matrix coefficients are positive semidefinite ($A_i \succcurlyeq 0$, $B \succcurlyeq 0$). We can view the matrix B as being "covered", and the matrices A_i as the "covering" matrices. The variable x_i specifies "how much" of A_i is used to cover B.

We further define as *online semidefinite covering problem* as a semidefinite covering problem in which the covered matrix B is revealed in an online fashion, i.e., in each online step t a new matrix B_t is revealed as the matrix which must be covered, and the following relation holds: $B_t \succcurlyeq B_{t-1}$. The irrevocability property is expressed by the requirement that each variable x_i cannot be decreased at any point of time, and is only allowed to increase (or stay unchanged). In each step t the semidefinite constraint $\mathcal{A}x \succcurlyeq B_t$ must be satisfied.

In the dual *online semidefinite packing problem*, the objective function $B \bullet Y$ is revealed online, while the constraints $A_i \bullet Y \leq c_i$ are known in advance. The monotonicity is captured by allowing the variable matrix Y to only increase, e.g. $Y_t \succcurlyeq Y_{t-1}$.

Set Cover and Linear Covering-Packing as a Special Case. When all matrices A_i and B_t are diagonal, the constraint $\sum_{i=1}^n A_i x_i \succcurlyeq B_t$ defines m linear constraints – one for every diagonal element $\sum_{i=1}^n (A_i)_{j,j} x_i \geq (B_t)_{j,j}$. This is because a diagonal matrix D is p.s.d. if and only if all of its diagonal entries are non-negative. Thus, revealing parts of the matrix B is equivalent to revealing new linear constraints. We assume without loss of generality that the linear constraints are revealed one row at a time, meaning that we reduced the primal problem to an instance of online fractional covering. Specifically, if all the diagonal entries are either 0 or 1, the problem then becomes online set cover. In the dual problem, since Y is only multiplied by diagonal matrices, only its diagonal is taking a part in the program, therefore the dual problem also becomes equivalent to online fractional packing.

3 The General Algorithm

In this section we provide an algorithm for the problem of *online semidefinite covering*, as defined in the previous section:

$$\begin{array}{cc} \min \sum_{i=1}^{n} c_i x_i & \max B_t \bullet Y \\ s.t.\ \sum_{i=1}^{n} A_i x_i \succcurlyeq B_t & s.t.\ \forall i \in [n]\ A_i \bullet Y \leq c_i \\ \forall i \in [n]\ \ x_i \geq 0 & Y \succcurlyeq 0 \end{array} .$$

In each online step t a new matrix B_t is revealed, satisfying $B_t \succcurlyeq B_{t-1}$. Our algorithm must then increase the variables x_i, incurring an additional cost of $c_i \Delta x_i$, such that the updated covering constraint $\sum_{i=1}^{n} A_i x_i \succcurlyeq B_t$ is satisfied.

3.1 Intuition

Our algorithm performs a sort of binary search on the optimal value of the primal problem. For a certain guess α, we either find a primal solution whose value does not exceed α, or find a dual solution whose value is at least $\alpha / O(\log n)$. By doubling our guess each time, we are able to mitigate the cost of failed guesses and only lose a factor of 2 in total. Each guess α defines a phase, which may extend over many online steps. During a phase, the primal solution's value is always less than α, and when the phase ends, the dual solution's value is at least $\alpha / O(\log n)$.

We maintain monotonicity within each phase by only increasing the primal and dual variables. Monotonicity is maintained between phases by setting each variable to be the sum (or the maximum) of its values in the previous phases. We note that the optimal value of the primal problem in each online step can only increase (or at least not change).

The idea behind the algorithm's update step is the following. Either the primal constraint $\mathcal{A}x \succcurlyeq B_t$ is already satisfied (and then we are done), or there exists a matrix $V \succcurlyeq 0$ such that $\mathcal{A}x \bullet V < B_t \bullet V$ (by observing that $C \succcurlyeq 0$ if and only if for every matrix $V \succcurlyeq 0$, $C \bullet V \geq 0$). The requirement that $trV = 1$ is a simple scaling constraint, which can be achieved by setting $V' = V / tr(V)$. We can think of the matrix V as a witness to the violation of the primal constraint; it induces a linear constraint $\mathcal{A}x \bullet V \geq B_t \bullet V$ which is violated. We use the witness as the direction in which we make dual progress, and increase the primal according to the relation we wish to maintain between the primal and dual solutions, until the linear constraint $\mathcal{A}x \bullet V \geq B_t \bullet V$ is satisfied. Specifically, the rate at which x_i is increased is proportional to $A_i \bullet V$, which makes sense, since this is the coefficient of x_i in the linear constraint $\mathcal{A}x \bullet V = \sum_{i=1}^{n} A_i \bullet V x_i \geq B_t \bullet V$.

We note that for technical reasons that help simplify the analysis we actually over-satisfy the above linear constraint, i.e. we continue increasing until $\mathcal{A}x \bullet V \geq \mathbf{2} B_t \bullet V$. This is necessary to ensure that we make enough progress for each choice of V, thus bounding the number of steps needed. Without this requirement, we might make infinitesimal progress for each V, and it is not clear if the algorithm will finish its execution, since there are infinitely many choice of V possible. It turns our that this over-satisfaction only contributes a factor of 2 to the competitive ratio.

There are many possible choices of progress direction V. One natural choice is $V = vv^T$, where v is a unit eigenvector corresponding to the smallest eigenvalue of $(\mathcal{A}x - B_t)$. Since $\mathcal{A}x - B_t \not\succcurlyeq 0$, its smallest eigenvalue λ_n is negative, therefore $(\mathcal{A}x - B_t) \bullet V = v^T (\mathcal{A}x - B_t) v = \lambda_n < 0$ as required.

3.2 Algorithm Description

In each phase r we search for a primal solution with cost at most $\alpha(r)$. Whenever the primal cost exceeds $\alpha(r)$, a new phase starts and $\alpha(r+1) \leftarrow 2\alpha(r)$ is doubled. In each new phase we "forget" about the values of the primal and dual variables from the previous phase, but we do account for their cost in the analysis. That is, in each phase r, new variables $x_{r,i}$, Y_r are introduced; however, since the variables are required to be monotonically non-decreasing, each variable x_i is actually set to $\sum_r x_{r,i}$, and the dual variable is also set to $\sum_r Y_r$.

Let OPT_t be the optimal value of the primal problem in phase t. After the first constraint matrix B_1 is introduced, we set the first lower-bound: $\alpha(1) \leftarrow \min_{i=1}^{n} \frac{c_i tr B_1}{tr A_i}$. Note that $\alpha(1) \leq OPT_1 \leq OPT_t$, for all t, because the matrix $Y_0 = \left(\min_{i=1}^{n} \frac{c_i}{tr A_i}\right) I$ is a feasible solution for the dual problem and its cost is $Y_0 \bullet B_1 = \alpha(1)$. In the beginning of each phase, we initialize $Y = 0$, $x_i = \frac{\alpha(r)}{2nc_i}$. Algorithm 2 describes the phase scheme. Algorithm 1 describes the execution during a single online step t, within a single phase r. Algorithm 1 uses an auxiliary variable δ, the measure of progress by which we update all other variables.

Algorithm 1 Primal-Dual Algorithm for step t within phase r

Input: current solutions x, Y, current limit α. Output: updated values for x, Y.
While $\mathcal{A}x \not\succcurlyeq B_t$:

1. Let V be a density matrix ($V \succcurlyeq 0$, $trV = 1$) such that $\mathcal{A}x \bullet V < B_t \bullet V$.
2. While $\mathcal{A}x \bullet V < 2B_t \bullet V$ and $\sum_{i=1}^{n} c_i x_i < \alpha$:
   a. Set $\delta = 0$ initially, and increase it in a continuous manner.
   b. Increase Y continuously by adding $V\delta$ to it.
   c. Increase x continuously by setting:

$$x_i = \frac{\alpha}{2nc_i} \exp\left(\frac{\log 2n}{c_i} A_i \bullet Y\right).$$

Algorithm 2 Phase Scheme

Initialize $r = 1$, $\alpha(0) = \min_{i=1}^{n} \frac{c_i tr B_1}{2tr A_i}$.
For $t = 1, 2, \ldots$:

1. Let $\alpha(r) = 2\alpha(r-1)$, $Y_r = 0$, $x_{r,i} = \frac{\alpha(r)}{2nc_i}$ for $i = 1, \ldots, n$.
2. Run Algorithm 1 on x_r, Y_r, $\alpha(r)$.
3. If $\sum c_i x_{r,i} \geq \alpha(r)$, then update $r \leftarrow r+1$ and go to step 1.
4. Return solutions $\sum_r x_{r,i}$, $\sum_r Y_r$.

▶ **Theorem 1.**

- *The scheme generates a feasible primal solution with a competitive ratio of* $O(\log n)$.
- *The scheme generates a dual solution with competitive ratio of* $O(\log n)$*, and each constraint is violated by a factor of at most* $O\left(\log\log n + \log \frac{OPT_t}{\alpha(1)}\right)$.
- *The scheme terminates, and its runtime is* $O\left(n\left(\log\log n + \log \frac{OPT_t}{\alpha(1)}\right)\right)$

Before we prove the theorem, we establish some assisting claims. Let $X(r)$ and $Y(r)$ be the values of the primal and dual solutions, respectively, generated in the r-th phase. We say that the r-th phase is *finished* if the condition in step 3 of Algorithm 2 holds, i.e. $\sum c_i x_{r,i} \geq \alpha(r)$.

▶ **Lemma 2.** *For each* finished *phase* r, $Y(r) \geq \alpha(r)/4\log 2n$.

Proof. In the beginning of phase r, we have $x_i = \frac{\alpha(r)}{2nc_i}$, therefore $X(r) = \frac{\alpha(r)}{2}$. The dual cost is zero. When the phase ends, $X(r) \geq \alpha(r)$. The proof is completed by showing that at every point in Algorithm 1, we have $\frac{\partial X(r)}{\partial \delta} \leq 2\log 2n \frac{\partial Y(r)}{\partial \delta}$. To show this, we have

$$\begin{aligned}
\frac{\partial X(r)}{\partial \delta} &= \frac{\partial}{\partial \delta}\left(\sum_{i=1}^{n} c_i x_i\right) = \sum_{i=1}^{n} c_i \frac{\partial x_i}{\partial \delta} = \sum_{i=1}^{n} c_i \frac{\partial}{\partial \delta}\left(\frac{\alpha(r)}{2nc_i}\exp\left(\frac{\log 2n}{c_i} A_i \bullet (Y+\delta V)\right)\right) \\
&= \sum_{i=1}^{n} c_i \frac{\log 2n}{c_i}(A_i \bullet V)\frac{\alpha(r)}{2nc_i}\exp\left(\frac{\log 2n}{c_i} A_i \bullet (Y+\delta V)\right) \\
&= \sum_{i=1}^{n} \log 2n\,(A_i \bullet V)\,x_i = \log 2n\left(\sum_{i=1}^{n} A_i x_i\right) \bullet V \\
&= \log 2n \mathcal{A}x \bullet V \leq 2\log 2n B_t \bullet V = 2\log 2n \frac{\partial Y(r)}{\partial \delta}.
\end{aligned}$$

The last inequality holds since we only increase δ while $\mathcal{A}x \bullet V < 2B_t \bullet V$. ◀

▶ **Lemma 3.** *The dual solution generated during each phase is feasible.*

Proof. Consider the i-th dual constraint of the form $A_i \bullet Y \leq c_i$. Since we are in the r-th phase, the current primal solution's value is at most $\alpha(r)$, therefore the value of x_i can be at most $\frac{\alpha(r)}{c_i}$. Thus:

$$\frac{\alpha(r)}{2nc_i}\exp\left(\frac{\log 2n}{c_i} A_i \bullet Y\right) = x_i \leq \frac{\alpha(r)}{c_i}.$$

Simplifying, we get that $A_i \bullet Y \leq c_i$. The final dual constraint $Y \succcurlyeq 0$ is satisfied since Y is the sum of p.s.d matrices δV. ◀

▶ **Lemma 4.** *The total cost of the primal solutions generated from the first phase until the r-th phase is less than $2\alpha(r)$.*

Proof. We bound the total cost paid by the online algorithm. The total primal cost in the r-th phase is at most $\alpha(r)$. Since the ratio between $\alpha(k)$ and $\alpha(k-1)$ is 2, we get that the total cost until the r-th phase is at most $\sum_{k=1}^{r}\alpha(k) = \alpha(r)\sum_{k=1}^{r}\frac{1}{2^{k-1}} \leq 2\alpha(r)$. ◀

▶ **Lemma 5.** *If the algorithm stops during a certain phase, then x is feasible.*

Proof. The algorithm stops only when $\mathcal{A}x \succcurlyeq B_t$. Also, $x \geq 0$ throughout the entire run of the algorithm, since we only increase each x_i from an initially positive value $\frac{\alpha(r)}{2nc_i} > 0$. ◀

▶ **Lemma 6.** *The number of iterations (= choices of V) in each phase is at most $n+1$.*

Proof. Consider any iteration in Algorithm 1 for which the while loop terminates because $\mathcal{A}x \bullet V \geq 2B_t \bullet V$. In the beginning of the while loop, we have $\mathcal{A}x \bullet V < B_t \bullet V$. This implies that at least one x_i was doubled during the iteration. Now, $x_i \geq \frac{\alpha(r)}{2nc_i}$, therefore each iteration increases the primal value by at least $c_i x_i \geq \frac{\alpha(r)}{2n}$. Since the primal value is $\frac{\alpha(r)}{2}$ in the beginning of the phase, after n such choices of V it must reach $\alpha(r)$ and the phase will be finished. Accounting for the possibly one extra iteration when the while loop terminates because the condition $\sum_i c_i x_i \geq \alpha$ is satisfied, the number of iterations is bounded by $n+1$. ◀

▶ **Lemma 7.** *The number of phases reached by step t is bounded by* $O\left(\log\log n + \log\frac{OPT_t}{\alpha(1)}\right)$.

Proof. Let $\mathcal{R} > 1$ be the current phase. Then

$$\alpha(1)\,2^{\mathcal{R}-2} = \alpha(\mathcal{R}-1) \le 4\log(2n)\,Y(\mathcal{R}-1) \le 4\log(2n)\,OPT_t$$

$$\Rightarrow \mathcal{R} \le 2 + \log\left(2\log(2n)\,\frac{OPT_t}{\alpha(1)}\right) = O\left(\log\log n + \log\frac{OPT_t}{\alpha(1)}\right).$$

The first inequality above follows by Lemma 2, and the second because $Y(\mathcal{R}-1)$ is a feasible solution by Lemma 3. ◀

Proof of Theorem 1. Suppose the online scheme terminates within $\mathcal{R}$ phases. Note that if $\mathcal{R} = 1$ then since $X(1) \le \alpha(1) \le OPT_1$ and x is feasible (from Lemma 5)), we must have exactly reached the end of the phase, i.e. $X = \alpha(1) = OPT_1$, $Y \ge \frac{OPT_1}{\log 2n}$, and Y is also feasible (from Lemma 3). From now on assume $\mathcal{R} > 1$.

- By Lemma 5, the primal solution is feasible. The total primal cost is bounded by:

$$\text{total primal cost} \ \le 2\alpha(\mathcal{R}) = 4\alpha(\mathcal{R}-1) \le 16\log 2nY(\mathcal{R}-1) \le 16\log(2n)\,OPT_t.$$

  The first inequality is by Lemma 4, the second by Lemma 2, and the last one by Lemma 3.
- By Lemma 3, the dual solution in each phase is feasible. Summing up over $\mathcal{R}$ phases and using the bound for $\mathcal{R}$ from Lemma 7, we get that the dual solution is violated up to $\mathcal{R} \le O\left(\log\log n + \log\frac{OPT_t}{\alpha(1)}\right)$. The total dual cost is bounded by:

$$\text{total dual cost} = \sum_{r=1}^{\mathcal{R}} Y(r) \ge Y(\mathcal{R}-1) \ge \frac{\alpha(\mathcal{R}-1)}{4\log 2n} = \frac{\alpha(\mathcal{R})}{8\log 2n} \ge \frac{X(\mathcal{R})}{8\log 2n} \ge \frac{OPT_t}{8\log 2n}.$$

  The second inequality is by Lemma 2 and the last one by Lemma 5.
- The runtime and termination of the scheme follow immediately from Lemmas 6 and 7.

◀

4 Box Constraints

In this section we introduce box constraints to our program, i.e. every variable x_i is now limited to a bounded range $0 \le x_i \le u_i$. Surprisingly, these bounds allows us to achieve approximation factors which do not depend on n, the number of matrices in the primal problem, but rather on a natural property of the program which we call the *sparsity*. Without loss of generality, we can assume $u_i = 1$ by replacing A_i with $A'_i = u_i A_i$, and c_i with $c'_i = u_i c_i$. We note that this assumption does alter the sparsity of the program. Our primal-dual problems with box constraints are the following:

$$\begin{array}{cc} \min \sum_{i=1}^n c_i x_i & \max B_t \bullet Y - \sum_{i=1}^n z_i \\ s.t.\ \sum_{i=1}^n A_i x_i \succcurlyeq B_t & s.t.\ \forall i \in [n]\ A_i \bullet Y - z_i \le c_i \\ \forall i \in [n]\ \ 0 \le x_i \le 1 & Y \succcurlyeq 0,\ z \ge 0 \end{array}.$$

4.1 Sparsity

The sparsity of a semidefinite program is defined as:

$$F^* = \max_t \max_{S \subseteq [n]} \min_{V \succcurlyeq 0:\ \left(B_t - \sum_{i \in S} A_i\right) \bullet V > 0} \frac{\sum_{i \notin S} A_i \bullet V}{\left(B_t - \sum_{i \in S} A_i\right) \bullet V}.$$

This quantity can be viewed as follows. If we saturate all variables in S, then $\sum_{i\in S} A_i$ is the coverage we get from the saturated variables, and $B_t - \sum_{i\in S} A_i$ is the remainder that we still need to cover. $\sum_{i\notin S} A_i$ is the total coverage potential of the unsaturated variables. We project both of these matrices, $\sum_{i\notin S} A_i$ and $B_t - \sum_{i\in S} A_i$, onto some witness V, and take the ratio $\frac{\sum_{i\notin S} A_i \bullet V}{\left(B_t - \sum_{i\in S} A_i\right)\bullet V}$. This is the ratio by which we "over-cover" the remainder $B_t - \sum_{i\in S} A_i$ if we use up all non-saturated variables. It turns out that $F^* \geq 1$. We actually use a relaxed version of F^* which may yield an even better result:

$$F = \max_{t,S \text{ occuring in the algorithm}} \min_{V\succcurlyeq 0:\ \left(B_t - \sum_{i\in S} A_i\right)\bullet V > 0} \frac{\sum_{i\notin S} A_i \bullet V}{\left(B_t - \sum_{i\in S} A_i\right)\bullet V}.$$

Here, the maximum is taken only over each online step t and the corresponding set of saturated variables S at step t. For fixed t, S, we can compute the minimum value of this ratio (and a minimizer V) very efficiently, see Lemma 8 below for details. Although we show that $F \leq F^*$, we still phrase our results using F^*, since F^* only depends on the given program, while F also depends on the specific choices made by the algorithm. We note that it is not necessarily the case that $F^* \leq n$, thus it would only make sense to use this parameter if the specifics of the problem can guarantee a better bound on F^* than n.

To further illustrate the meaning of F^*, it is useful to consider the set cover problem. As discussed earlier, set cover is obtained when all matrices A_i and B_t are diagonal, and having only entries from $\{0, 1\}$ on the diagonal. To analyze F^* in this case, it is sufficient to observe witness matrices V of the form $E_{j,j}$ (any other witness is simply a convex combination of these). Then $\sum_{i\notin S} A_i \bullet V$ and $\sum_{i\in S} A_i \bullet V$ are equal to the number of non-saturated and saturated sets which contain element j respectively; and $B_t \bullet V$ is 1 if element j needs to be covered and 0 otherwise. Clearly the only way in which $\left(B_t - \sum_{i\in S} A_i\right) \bullet V > 0$, is if element j needs to be covered and none of the sets containing it are saturated. Then, the expression $\frac{\sum_{i\notin S} A_i \bullet V}{\left(B_t - \sum_{i\in S} A_i\right)\bullet V}$ is exactly the number of sets containing j. Thus, F^* in this case is exactly the maximum number of sets that an element is contained in (taken over the elements which the algorithm is required to cover). This matches the notion of *row sparsity* of a linear program, and clearly in this case $F^* \leq n$.

▶ **Lemma 8.** *Let N be a positive semidefinite matrix and D be a symmetric matrix. Then the minimum value of the ratio $\frac{N\bullet V}{D\bullet V}$ such that $V \succeq 0$ and $D \bullet V > 0$ is equal to the smallest non-negative root of the polynomial equation $\det(N - \lambda D) = 0$.*

Proof. Let V be an optimal solution. Since $V \succeq 0$, we can express it as $V = \sum_\ell v_\ell v_\ell^T$ for some vectors v_ℓ. Since $N \succeq 0$, we have $N \bullet v_\ell v_\ell^T \geq 0$. Then we claim that $D \bullet v_\ell v_\ell^T \geq 0$; otherwise, we could drop $v_\ell v_\ell^T$ from the sum forming V and decrease the ratio. Next, note that $\frac{N\bullet V}{D\bullet V} = \frac{\sum_\ell N\bullet v_\ell v_\ell^T}{\sum_\ell D\bullet v_\ell v_\ell^T} \geq \min_\ell \left\{\frac{N\bullet v_\ell v_\ell^T}{D\bullet v_\ell v_\ell^T}\right\}$, and so we conclude that there is a rank 1 minimizer. Let this minimizer be $V = vv^T$ for some vector v. The minimum ratio is then obtained by minimizing $v^T N v$ subject to $v^T D v = 1$. The Karush-Kuhn-Tucker conditions imply that the optimal vector v satisfies the generalized eigenvalue equation, $Nv = \lambda Dv$ for some constant λ (the generalized eigenvalue). Note that $\frac{N\bullet vv^T}{D\bullet vv^T} = \lambda$. Since we require $v^T D v = 1$, and $v^T N v \geq 0$, only non-negative values of λ are admissible. Since the generalized eigenvalues λ are roots of the polynomial equation $\det(N - \lambda D) = 0$, and we conclude that the minimum value of the ratio is the smallest non-negative root of this polynomial equation. ◀

4.2 Algorithm Description

We now describe an algorithm for semidefinite covering-packing with box constraints. We denote by S the set of indices i of *saturated* primal variables x_i, i.e. variables which have reached their upper bound: $S = \{i | x_i = 1\}$. We use the following notations for simplicity: $\mathcal{A}_{\bar{S}} x = \sum_{i \notin S} A_i x_i$, and $\mathcal{A}_S x = \sum_{i \in S} A_i$ (note that in the second notation, we omit the x_i's because they are all saturated, so are equal to 1). We use an auxiliary variable δ, which is the measure of progress by which we update all other variables.

Algorithm 3 Primal-Dual Algorithm For Box Constraints

Initialize $x = 0$, $z = 0$, $Y = 0$.

Upon arrival of a new constraint matrix B_t, while $\mathcal{A}x \not\succcurlyeq B_t$:

1. Let V be a density matrix ($V \succcurlyeq 0$, $trV = 1$) such that $\mathcal{A}x \bullet V < B_t \bullet V$, minimizing the ratio $\frac{\sum_{i \notin S} A_i \bullet V}{B_t \bullet V - \sum_{i \in S} A_i \bullet V}$.
2. Update $F = \max\left\{F, \frac{\sum_{i \notin S} A_i \bullet V}{B_t \bullet V - \sum_{i \in S} A_i \bullet V}\right\}$.
3. While $\mathcal{A}_{\bar{S}} x \bullet V < 2\left[B_t \bullet V - \mathcal{A}_S x \bullet V\right]$:
   a. Set $\delta = 0$ initially, and increase it in a continuous manner.
   b. Increase Y continuously by adding $V\delta$ to it.
   c. For every i such that $x_i = 1$, increase z_i continuously at rate $A_i \bullet V\delta$.
   d. Update x continuously by setting:

$$x_i = \max\left\{x_i, \frac{1}{F}\left(\exp\left(\frac{\log(F+1)}{c_i}\left(A_i \bullet Y - z_i\right)\right) - 1\right)\right\}.$$

▶ **Theorem 9.** *The algorithm produces a feasible primal solution with a competitive ratio of* $O(\log F^*)$*, and a feasible dual solution with a competitive ratio of* $O(\log F^*)$*.*

The proof is omitted due to space constraints.

5 Rounding

In this section, we discuss the integer version of online semidefinite programming, and how our results in the previous sections can be rounded to an integer solution.

5.1 Changing the Framework

An *integer semidefinite program* is the following:

$$\begin{array}{c} \min \sum_{i=1}^{n} c_i \hat{x}_i \\ s.t. \ \sum_{i=1}^{n} A_i \hat{x}_i \succcurlyeq I \\ \forall i \in [n] \ \ \hat{x}_i \in \mathbb{N}^n \end{array}.$$

Note the different notation for integer variables $\hat{x}_i$ as opposed to real-valued variables x_i of the previous sections. Also note that we now restrict the covered matrix to be the identity matrix I. This is a necessity for our rounding scheme. To justify why this is not any different from the general setting, we invoke some linear algebra. First, assume that B is of full rank (and, of course, p.s.d.). Then B is congruent to I, by some invertible matrix X such that

$X^T BX = I$. Now $\sum_{i=1}^n A_i \hat{x}_i \succcurlyeq B$ if and only if $\sum_{i=1}^n X^T A_i X \hat{x}_i \succcurlyeq X^T BX = I$. We can change the matrices $A'_i = X^T A_i X$ so as to arrive at the formulation above.

However, restricting B to be of full rank throughout the entire online algorithm does not make much sense. To overcome this, we change the framework to allow the dimension m of the matrices to increase over time. Let m_t be the dimension of the online problem at step t. Then, in step $t+1$, the dimension of the matrices may increase to $m_t + 1$ (or may stay m_t). The covering matrices $A_i^{(t+1)}$ must contain $A_i^{(t)}$ as their top-left submatrix, and they must be p.s.d., i.e. $A_i^{(t+1)} \succcurlyeq 0$. The covered matrix $B^{(t+1)}$ must also be p.s.d., and its top-left submatrix of dimension m_t must be greater than or equal to $B^{(t)}$ in the p.s.d. partial ordering. The *online integer semidefinite program* is therefore:

$$\begin{array}{c} \min \sum_{i=1}^n c_i \hat{x}_i \\ s.t. \ \sum_{i=1}^n A_i^{(t)} \hat{x}_i \succcurlyeq B^{(t)} \\ \forall i \in [n] \ \ \hat{x}_i \in \mathbb{N}^n \end{array},$$

where $A_i^{(t)}, B^{(t)} \in \mathbb{R}^{m_t \times m_t}$, and $B^{(t)}$ is of full rank for all t.

Under these assumptions, it can be verified that our algorithms from Sections 3 and 4 still work (for real-valued solutions $x_i \in \mathbb{R}^n$). For Section 3, it is important to note that OPT_t indeed increases with t. The proof of the next lemma is omitted due to space constraints.

▶ **Lemma 10.** *With the above assumptions, $OPT_t \leq OPT_{t+1}$.*

Another consideration when discussing rounding is scale – if $A_1 = (M)$ and $B = (1)$ (matrices of dimension 1), then the optimum fractional solution is $x_1 = \frac{1}{M}$ but the optimum integer solution is 1, which costs M times the fractional solution. Therefore we must assume some kind of bound on the covering matrices (we use $\lambda_1(A_i) \leq R$), and we will be paying for this bound in our competitive ratio. As demonstrated by this example, this is unavoidable.

5.2 A Randomized Rounding Algorithm

We use a simple randomized rounding scheme. We run the algorithm described in Section 3[1] to maintain a fractional solution $x^{(t)}$, and we round that solution to an integer solution $\hat{x}^{(t)}$, so that $\mathbb{E}\hat{x}^{(t)} = \rho_t x^{(t)}$, where ρ_t is an approximation factor (see algorithm for exact definition), which we adjust as the dimension of the matrices grows. We then use the feasibility of $x^{(t)}$ to show that w.h.p. $\hat{x}^{(t)}$ is also feasible. The rounding is done as follows: whenever a variable x_i increases from $x_i^{(t-1)}$ to $x_i^{(t)}$, we add 1 to $\hat{x}_i$ with probability $\rho_t x_i^{(t)} - \rho_{t-1} x_i^{(t-1)}$. Since m_t and $x^{(t)}$ can only increase with t, so can ρ_t and thus $\rho_t x_i^{(t)} - \rho_{t-1} x_i^{(t-1)}$ is non-negative. To make sure that this value is always ≤ 1, we can simply break up the increase in x_i into smaller steps. This is expressed formally in algorithm 4.

▶ **Theorem 11.** *The resulting solution's expected cost is at most $O(R(\log m_t + \log n))$ times the cost of the fractional solution.*

The proof is omitted due to space constraints.

[1] If box constraints are given, we can run the algorithm from Section 4 instead. The only modification to the rounding is that each variable is increased at most once. The same analysis holds for both cases.

Algorithm 4 Randomized rounding

- Let $R > 0$ be such that $\lambda_1(A_i) \leq R$ for all i.
- Initialize $\hat{x}^{(0)} = 0$
- For step $t = 1, 2, \ldots$:
  - Let $\rho_t = 1 + \max\{8R(\log 2m_t + \log n), 2\}$.
  - Use Algorithms (2) or (3) to compute the fractional solution $x^{(t)}$.
  - For $i = 1, \ldots, n$:
    - Let $d_{i,t} = \left\lfloor \rho_t x_i^{(t)} - \rho_{t-1} x_i^{(t-1)} \right\rfloor$, and $\delta_{i,t} = \rho_t x_i^{(t)} - \rho_{t-1} x_i^{(t-1)} - d_{i,t}$
    - Let $\hat{x}_{i,1}^{(t)}, \ldots, \hat{x}_{i,d_{i,t}}^{(t)} = 1$.
    - Let $\hat{x}_{i,d_{i,t}+1}^{(t)} = \begin{cases} 1 & \text{with probability } \delta_{i,t} \\ 0 & \text{otherwise} \end{cases}$.
    - return $\hat{x}_i^{(t)} = \hat{x}_i^{(t-1)} + \hat{x}_{i,1}^{(t)} + \cdots + \hat{x}_{i,d_{i,t}+1}^{(t)}$.

5.3 An Application – Quantum Hypergraph Covering

Quantum hypergraphs originate from the field of quantum information theory. A hypergraph is a pair (V, E) where $E \subseteq 2^V$. The hypergraph covering problem (finding a collection of edges covering all vertices) is equivalent to set cover. Let each edge $e \in E$ be represented by a $\{0,1\}$ diagonal matrix A_e, where the i-th diagonal entry is 1 if and only if the i-th element in V belongs to e. The hypergraph covering problem is to find a collection of edges such that $\sum_{e \in cover} A_e \succcurlyeq I$. A *quantum hypergraph* is a pair (V, E), where each $e \in E$ corresponds to a symmetric matrix A_e of dimension $|V|$, such that $0 \preccurlyeq A_e \preccurlyeq I$. The difference is that A_e need not be diagonal or only have $\{0,1\}$ eigenvalues. The *quantum hypergraph covering* problem is thus the problem of finding a minimum collection of edges in E that satisfy $\sum_{e \in cover} A_e \succcurlyeq I$. The *fractional quantum hypergraph covering* problem is the problem of assigning non-negative weights x_e to each $e \in E$ such that $\sum_{e \in E} x_e A_e \succcurlyeq I_{|V|}$, while minimizing $\sum_{e \in E} x_e$.

Ahlswede and Winter [1] used a novel approach to develop a matrix-valued generalization of the Chernoff inequality. They applied this inequality in the analysis of their randomized rounding scheme, which takes a fractional quantum hypergraph cover and rounds it to a quantum hypergraph cover. They showed that this rounding finds a cover which is at most $O(\log m)$ times larger than the minimum cover, where m is the dimension of the matrices involved. Wigderson and Xiao [7] derandomized this result to provide a deterministic algorithm for quantum hypergraph covering. Their result also produces an $O(\log m)$-approximation cover. In addition to the quantum hypergraph covering problem, Wigderson and Xiao also provided a more general algorithm for integer semidefinite programming.

It is easy to see that the fractional quantum hypergraph covering problem is exactly the semidefinite covering problem, with the added requirement that all the covering matrices A_e fall in the range $[0, I]$, and the covered matrix is the identity matrix $B = I$. When translating this into an online setting, the most natural approach is to look at the setting presented in Section 5, where the dimension of the matrices can increase but the covered matrix must have full rank. The *online quantum hypergraph covering problem* is thus the problem of maintaining a monotonically increasing cover while the edges are revealed one dimension at a time. In this setting, combining algorithms 2 and 4 gives an $O(\log n \log m)$-competitive solution to online quantum hypergraph programming ($R = 1$ since $0 \preccurlyeq A_e \preccurlyeq I$).

References

1 Rudolf Ahlswede and Andreas Winter. Strong converse for identification via quantum channels. *Information Theory, IEEE Transactions on*, 48(3):569–579, 2002.

2 Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

3 Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Mathematics of Operations Research*, 34(2):270–286, 2009.

4 Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

5 Anupam Gupta and Viswanath Nagarajan. Approximating sparse covering integer programs online. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 436–448, 2012.

6 Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, 2012.

7 Avi Wigderson and David Xiao. Derandomizing the AW matrix-valued Chernoff bound using pessimistic estimators and applications, 2006.

Beating the Harmonic Lower Bound for Online Bin Packing*

Sandy Heydrich[1] and Rob van Stee[2]

1 **Max Planck Institute for Informatics, Saarbrücken, Germany; and Graduate School of Computer Science, Saarbrücken, Germany**
heydrich@mpi-inf.mpg.de

2 **Department of Computer Science, University of Leicester, Leicester, UK**
rvs4@le.ac.uk

Abstract

In the online bin packing problem, items of sizes in $(0, 1]$ arrive online to be packed into bins of size 1. The goal is to minimize the number of used bins. Harmonic++ achieves a competitive ratio of 1.58889 and belongs to the Super Harmonic framework [Seiden, J. ACM, 2002]; a lower bound of Ramanan et al. shows that within this framework, no competitive ratio below 1.58333 can be achieved [Ramanan et al., J. Algorithms, 1989]. In this paper, we present an online bin packing algorithm with asymptotic performance ratio of 1.5815, which constitutes the first improvement in fifteen years and reduces the gap to the lower bound by roughly 15%.

We make two crucial changes to the Super Harmonic framework. First, some of the decisions of the algorithm will depend on *exact sizes* of items, instead of only their types. In particular, for item pairs where the size of one item is in $(1/3, 1/2]$ and the other is larger than $1/2$ (a *large* item), when deciding whether to pack such a pair together in one bin, our algorithm does not consider their types, but only checks whether their total size is at most 1.

Second, for items with sizes in $(1/3, 1/2]$ (*medium* items), we try to pack the larger items of every type in pairs, while combining the smallest items with large items whenever possible. To do this, we *postpone* the coloring of medium items (i.e., the decision which items to pack in pairs and which to pack alone) where possible, and later select the smallest ones to be reserved for combining with large items. Additionally, in case such large items arrive early, we pack medium items with them whenever possible. This is a highly unusual idea in the context of Harmonic-like algorithms, which initially seems to preclude analysis (the ratio of items combined with large items is no longer a fixed constant).

For the analysis, we carefully mark medium items depending on how they end up packed, enabling us to add crucial constraints to the linear program used by Seiden. We consider the dual, eliminate all but one variable and then solve it with the ellipsoid method using a separation oracle. Our implementation uses additional algorithmic ideas to determine previously hand set parameters automatically and gives certificates for easy verification of the results.

We give a lower bound of 1.5766 for algorithms like ours. This shows that fundamentally different ideas will be required to make further improvements.

1998 ACM Subject Classification G.2.1 Combinatorial Algorithms

Keywords and phrases Bin packing, online algorithms, harmonic algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.41

* A full version of this paper can be found at http://arxiv.org/abs/1511.00876.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 41; pp. 41:1–41:14

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

1 Introduction

In the online bin packing problem, a sequence of *items* with sizes in the interval $(0, 1]$ arrive one by one and need to be packed into *bins*, so that each bin contains items of total size at most 1. Each item must be irrevocably assigned to a bin before the next item becomes available. The algorithm has no knowledge about future items. There is an unlimited supply of bins available, and the goal is to minimize the total number of used bins (bins that receive at least one item). Bin packing is a classical and well-studied problem in combinatorial optimization. Extensive research has gone into developing approximation algorithms for this problem, e.g. [5, 7, 6, 12, 17, 9]. Such algorithms have provably good performance for any possible input and work in polynomial time. In fact, the bin packing problem was one of the first for which approximation algorithms were designed [10].

For bin packing, we are typically interested in the long-term behavior of algorithms: how good is the algorithm for large inputs? If we simply compare to the optimal solution, the worst ratio is often determined by some very small inputs. To avoid such pathological instances, the *asymptotic performance ratio* was introduced: For a given input sequence σ, let $\mathcal{A}(\sigma)$ be the number of bins used by algorithm $\mathcal{A}$ on σ. The *asymptotic performance ratio* for an algorithm $\mathcal{A}$ is defined to be

$$\mathcal{R}_{\mathcal{A}}^{\infty} = \limsup_{n\to\infty} \sup_{\sigma} \left\{ \frac{\mathcal{A}(\sigma)}{\mathrm{OPT}(\sigma)} \,\middle|\, \mathrm{OPT}(\sigma) = n \right\}. \tag{1}$$

From now on, we only consider the asymptotic competitive ratio unless otherwise stated.

Lee and Lee [13] presented an algorithm called Harmonic, which partitions the interval $(0, 1]$ into $m > 1$ intervals $(1/2, 1], (1/3, 1/2], \ldots, (0, 1/m]$. The type of an item is defined as the index of the interval which contains its size. Each type of items is packed into separate bins (i items per bin for type i). For any $\varepsilon > 0$, there is a number m such that the Harmonic algorithm that uses m types has a performance ratio of at most $(1 + \varepsilon)\Pi_\infty$ [13], where $\Pi_\infty \approx 1.69103$.

If we consider the bins packed by Harmonic, then it is apparent that in bins with type 1 items, nearly half the space can remain unused. It is better to use this space for items of other types. After a sequence of papers which used this idea to develop ever better algorithms [13, 15, 16], Seiden [18] presented a general framework called Super Harmonic which captures all of these algorithms. Super Harmonic algorithms classify items based on an interval partition of $(0, 1]$ and give each item a color as it arrives, red or blue. For each type of items j, the fraction of red items is some constant denoted by α_j. Blue items are packed as in Harmonic, i.e., for each item type j, every bin with blue items contains a maximal number of blue items. (This may leave some space for smaller red items of different types.) Red items are packed in bins which are only partially filled. The idea is that hopefully, later blue items of other types will arrive that can be placed into the bins with red items. Seiden [18] showed that the Super Harmonic algorithm Harmonic++, which uses 70 intervals for its classification and has about 40 manually set parameters, achieves a performance ratio of at most 1.58889.

Ramanan et al. [15] gave a lower bound of $19/12 \approx 1.58333$ for this type of algorithm. It is based on inputs like the one shown in Figure 1, which contains a *medium* item (size in $(1/3, 1/2]$) and a *large* item (size in $(1/2, 1]$). Both of these items arrive N times for some large number N, and although they fit pairwise into bins, the algorithm never combines them like this. No matter how fine the item classification of an algorithm, pairs of items such as these, that the algorithm does not pack together into one bin, can always be found. (To complete the lower bound construction, we also need to consider inputs containing the

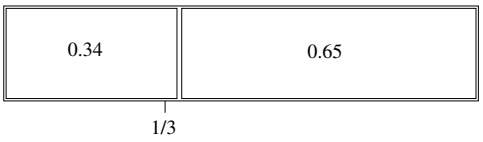
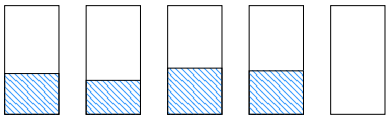
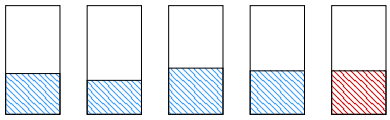


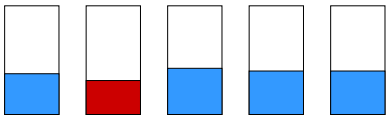
Figure 1 Part of the lower bound construction from Ramanan et al. [15]. The figure shows how one bin is packed in the *optimal* solution. Both of these items arrive many times.



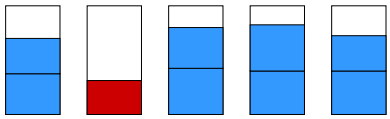
(a) Pack items one per bin with provisional coloring.



(b) A provisional red item arrives.



(c) We fix the colors. The smallest item becomes red.



(d) Additional blue items of the same type are added.

Figure 2 Illustration of the coloring in Extreme Harmonic. In this example, $\alpha = 1/9$. Note that the ratio of 1/9 does not hold (for the bins shown) at the time that the colors are fixed: 1/5 of the items are red at this point. The ratio 1/9 is achieved when all bins with blue items contain two blue items.

sizes $1/3+\varepsilon$, $1/2+\varepsilon$, which can be combined into a single bin, and the input consisting only of items of size $1/3+\varepsilon$.)

We avoid this lower bound construction by defining the algorithm so that it simply combines medium and large items *whenever* they fit together in a single bin. Essentially, we use Any Fit to combine such items into bins (under certain conditions specified below). This is a generalization of the well-known algorithms First Fit and Best Fit [19, 7], which have been used in similar contexts before [2, 1]. Proving formally that this helps to improve the asymptotic performance ratio requires a surprising amount of additional technical modifications to the algorithm and the proof, in particular setting up a complete marking scheme (see below).

As in the Super Harmonic framework, medium items that are packed in pairs are colored blue, and the ones that are packed alone into bins (possibly together with items of other types) are colored red. At this point it is important to note that medium items of any given type are not all exactly the same size, since the type only specifies an interval. This means that the items of any given type could arrive in such an order that all of the red items are slightly larger than the blue ones. Then, when large items arrive later, it could be that they are too large to fit in bins with red medium items, so the online algorithm is forced to pack them into new bins.

In order to benefit from using Any Fit, it is crucial to ensure that for each medium type, as much as possible, *it is the smallest items that are colored red.* We will do this by initially packing each medium item alone into a bin and giving it a provisional color. After several items of the same type have arrived, we will color the smallest one red and start packing additional medium items of the same type together with the other items, that are now colored blue. (See Figure 2.) In this way, we can ensure that at least *half* of the blue

items (namely, the ones that had already arrived at the time when we select the smallest to be red) are at least as large as the smallest red items. The point of this is that if those red items are still alone in bins at the end of the input, OPT cannot pack too many bins as shown in Figure 1, because this can only happen with large items that do not fit with the red items that remain alone in bins (Lemma 6).

We do not postpone the coloring decisions in the following two cases.

1. If a bin with suitable small red items is available, we will pack p into that bin and color it blue, regardless of the precise size of p.[1] In this case, in our analysis we will exploit the fact that these small items exist in the input, meaning that not *all* optimal bins are packed as shown in Figure 1: the small items must be packed somewhere (Lemma 7).
2. If bins with a *large* item are available, and p fits into such a bin, we will pack p in one such bin. This is the best case overall, since finding combinations like this was exactly our goal! However, there is a technical problem with this, which we discuss below.

Overall, we have three different cases: medium items are packed alone initially (in which case we have a guarantee about the sizes of some of the blue items), medium items are combined with smaller red items (so these red items exist and must be packed: Lemma 7), or medium items are combined with larger blue items (which is exactly our goal). The main technical challenge is to quantify these different advantages into one overall analysis. In order to do this (i.e., to prove Lemmas 6 and 7), we introduce - in addition to and separate from the coloring – a marking of the medium items, which we now describe.

$\mathcal{R}$ For any medium type j, a fraction α_j of the items marked $\mathcal{R}$ are red, and all of these **red** items are packed into mixed bins (i.e., together with a large item).

$\mathcal{B}$ For any medium type j, a fraction α_j of the items marked $\mathcal{B}$ are red, and the **blue** items are packed into mixed bins (i.e., together with red items of other (smaller) types)

$\mathcal{N}$ For any medium type j, a fraction α_j of the items marked $\mathcal{N}$ are red, and **none** of the red and blue items marked $\mathcal{N}$ are packed into mixed bins.

Our marking is illustrated in Figure 3. Maintaining the fraction α_j of red items for all marks separately is crucial for the analysis. However, we note here immediately that the fraction α_j of red items is **not** actually maintained continuously throughout the execution for all marks. This can be seen clearly for the items marked $\mathcal{R}$, where the ratio only becomes equal to α_j (ignoring rounding) after all the bins with single blue items in them receive additional blue items (see Figure 2).

Seemingly more problematically, it could happen that many large items arrive first, leading to more than an α_j fraction of the items of type j and mark $\mathcal{B}$ being packed with the large items and colored red. (Potentially, this could even happen to all of them.) While this is in principle exactly what we want to achieve, there is no guarantee that later in the input, sufficiently many additional items of type j will arrive to restore the correct ratio α_j. This is a problem for our analysis, which assumes the ratio α_j is maintained exactly. However, if we insist on maintaining this ratio throughout, i.e., if we color some of these items blue and pack them in pairs even though they could fit with existing large items, we end up with the same worst case instances as for Super Harmonic. We deal with this case by *modifying the input (for the analysis) after it has been packed.* By this and some additional postprocessing, we ensure that for each mark $\mathcal{R}, \mathcal{B}, \mathcal{N}$, an α_j fraction of the medium items of type j are indeed colored red in the end (ignoring rounding) as required. The postprocessing also ensures that

[1] Unless we already have sufficiently many blue items of the type of p, in which case we pack p into a separate bin and color it red to maintain the correct fraction of red items.

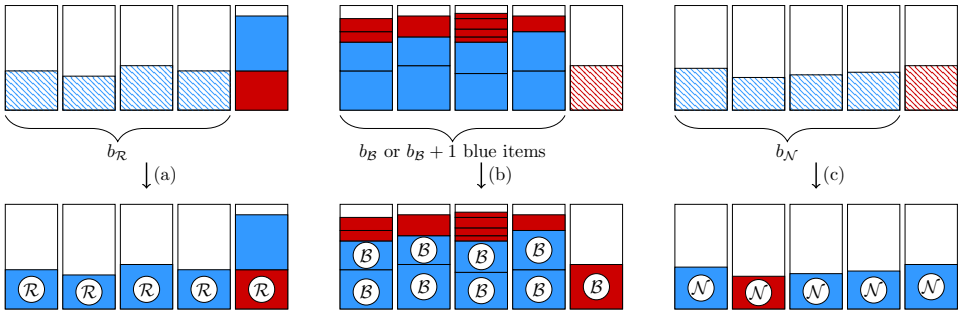


Figure 3 Illustrating the marking of the items. Again we take $\alpha_i = 1/9$. **(a)** Items get mark $\mathcal{R}$: provisionally blue items and a red item in a mixed bin. Bins with blue $\mathcal{R}$-items will receive a second blue item of the same type before a new bin is opened for this type. **(b)** Items get mark $\mathcal{B}$: a provisionally red item and blue items (in pairs) in mixed bins. **(c)** Items get mark $\mathcal{N}$: provisionally blue and provisionally red items. Note that in this step, the colors of items might be fixed to a different color than their provisional color. Bins with blue items will receive a second blue item of the same type before a new bin is opened for this type. See Fig. 2.

the marks are all correct. For instance, if a red item is packed with a blue item marked $\mathcal{N}$, the mark of that blue item gets adjusted in the end.

Like Seiden [18] and many other authors [19, 13, 15], we use weighting functions to analyze our algorithm. A weighting function defines a weight for each item type. By analyzing these, Seiden ended up with a set of mathematical programs that upper bounded the asymptotic performance ratio of Super Harmonic algorithms. These represented a kind of knapsack problems where each item has two different weights. Seiden used heuristics to get exact upper bounds for the solutions of these mathematical programs.

We use a different approach for the Extreme Harmonic framework. First of all we split each mathematical program into two standard linear programs, where both linear programs have a constraint that states its objective value should be smaller than that of the other one (representing for each one that the minimum is achieved for the set of weights it considers). To each linear program, we add two constraints that are based on the marking of the medium items. These constraints essentially state that in the optimal solution for a given input, there cannot be too many bins that are packed as shown in Figure 1 (unless the online algorithm also packs the items like this). This is the key to our improvement of the asymptotic performance ratio. However, after adding these constraints, the heuristic approach by Seiden can no longer be applied. Since each linear program has a very large number of variables but only four constraints, we take the dual and apply the ellipsoid method to solve it. To do this, we construct a separation oracle. This separation oracle solves a standard knapsack problem, making the results much easier to verify.

In order to apply the ellipsoid method, we write the dual in terms of just one variable, by eliminating two variables and assuming a third one to be given. This means that we can now do a straightforward binary search for the final remaining variable. We implemented a computer program which solves the knapsack problems and also does the other necessary work, including the automated setting of many parameters like item sizes and α values. As a result, our algorithm Son Of Harmonic requires far less manual settings than Harmonic++.

Our program uses an exact representation of fractions with arbitrary precision in order to avoid rounding errors. For our final calculations we have set the bound such that every dual LP is feasible; this means that our results do not rely on the correctness of any infeasibility claims (which are generally harder to prove). We provide a certificate and a

verifier program, and we also output the final set of knapsack problems directly to allow independent verification.

Our second main contribution is a new lower bound for all algorithms of this kind. The fundamental property of all these algorithms is that they color a fixed fraction of all items red (for each type). We show that no such algorithm can be better than 1.5766-competitive. Due to space constraints, this result is deferred to the full version.

1.1 Previous Results

The online bin packing problem was first investigated by Ullman [19]. He showed that the First Fit algorithm has performance ratio $\frac{17}{10}$. This result was then published in [7]. Johnson [11] showed that the Next Fit algorithm has performance ratio 2. Yao showed that Revised First Fit has performance ratio $\frac{5}{3}$, and further showed that no online algorithm has performance ratio less than $\frac{3}{2}$ [21]. Brown and Liang independently improved this lower bound to 1.53635 [4, 14]. The lower bound stood for a long time at 1.54014, due to van Vliet [20], until it was improved to $\frac{248}{161} = 1.54037$ by Balogh et al. [3].

The *offline* version, where all the items are given in advance, is well-known to be NP-hard [8]. This version has also received a great deal of attention, for a survey see [5].

2 The Super Harmonic framework [18]

The fundamental idea is to first classify items by size, and then pack an item according to its type. We use numbers $t_1 \geq t_2 \geq \cdots \geq t_N$ to partition the interval $(0, 1]$ into subintervals (N is a parameter). We define $I_j = (t_{j+1}, t_j]$ for $i = 1, \ldots, N$ and $I_{N+1} = (0, t_{N+1}]$. An item of size s has type j if $s \in I_j$. A type j item has size at most t_j.

For each type j, a fraction $\alpha_j \in [0, 1]$ of items are colored red when they arrive, the rest are colored blue. Blue items are packed using Next Fit: we use each bin until exactly $\text{bluefit}_j := \lfloor 1/t_j \rfloor$ items are packed into it. Red items are also packed using Next Fit, but using only some fixed amount of the available space in a bin. This space is not necessarily exactly some value $1 - \text{bluefit}_j t_j$; for any given type j, there may be several other types that the algorithm will potentially pack into a bin together with items of type j. For each type of items that have size at most $1/3$, the algorithm chooses in advance an upper bound for the space that red items may occupy from a fixed set $\mathcal{D} = \{\Delta_i\}_{i=1}^{K}$ of spaces, where $\Delta_1 \leq \cdots \leq \Delta_K$. For *medium* items (i.e., items whose size is in $(1/3, 1/2]$), red items are packed one per bin. The number of red items of type i that are packed in one bin is denoted by redfit_i. In the space not used by blue (resp. red) items, the algorithm may pack red (resp. blue) items. Each bin will contain items of at most two different types.

A Super Harmonic algorithm uses a function $b : \{1, \ldots, N\} \to \{0, \ldots, K\}$ to map each item type to an index of a space in $\mathcal{D}$, indicating how much space for red items it leaves unused in bins with blue items of this type. Here $b(j) = 0$ means that no space is left for red items. The algorithm also uses a function $r : \{1, \ldots, N\} \to \{1, \ldots, K\}$ to map how much space (given by an index of $\mathcal{D}$) red items of each type require.

We say that the *class* of an item of type j is $b(j)$, if it is blue, and $r(j)$ if it is red.[2] Thus, the class of a blue item reflects how much space is left (at least) in a bin with blue items of this type, and the class of a red item indicates how much space red items of this type require (at most) in a bin. There are four kinds of bins.

[2] Seiden used the notation $\phi(j)$ and $\varphi(j)$ for these functions.

- Pure blue: $\{i|b(i) = 0, 1 \leq i \leq N\}$. No red items are ever packed into such bins.
- Unmixed blue: $\{(i, ?)|b(i) \neq 0, 1 \leq i \leq N\}$. There is at least one blue item in the bin, and red items might still be packed into it (in the free space of size $\Delta_{b(i)}$).
- Unmixed red: $\{(?, j)|\alpha_i \neq 0, 1 \leq i \leq N\}$. There is at least one red item in the bin and no blue items, but blue items might still be packed in it (in the free space of size $1 - \Delta_{r(i)}$).
- Mixed bins: $\{(i, j)|b_i \neq 0, \alpha_j \neq 0, t_j \leq \Delta_{b(i)}, 1 \leq i \leq N, 1 \leq j \leq N\}$. There are items of both colors.

An unmixed blue bin is *compatible* with a red item of type i if the bin is in a group $(j, ?)$ and $b(j) \geq r(i)$. An unmixed red bin is compatible with a blue item of type i if the bin is in a group $(?, j)$ and $b(i) \geq r(j)$. In both cases, the condition means that the blue items and the red items together would use at most 1 space in the bin (the blue items leave enough space for the red items).

3 Marking the items and the Extreme Harmonic framework

The heart of our improvement over the Super Harmonic framework is marking the medium items. It enables us to keep track of how they are packed, allowing us to prove the crucial Lemmas 6 and 7 later, which bound how often "bad" patterns of the form shown in Figure 1 (which have weight > 1.5815) can be used in the optimal solution. Mark Items divides the medium items into three sets $\mathcal{N}, \mathcal{B}$ and $\mathcal{R}$ (see Figure 3). Every time an item arrives, after it is packed using the new framework below, Mark Items performs one of the three steps in Figure 3 if possible. This is done to keep the number of provisionally colored items small (a constant). We define Mark Items formally in the full version.

▶ **Theorem 1.** *At all times, there are at most* $5/\alpha_i$ *provisionally colored items of type* i*.*

Once assigned, an item remains in a set until the end of the input. This holds even if e.g. a blue item is packed with a red $\mathcal{N}$-item, meaning that a more appropriate mark for the red item is $\mathcal{B}$. We change marks where needed only after all items have been packed.

Let n^i count the total number of items of type i, and n^i_r count the number of red items of type i. For a given type i and set X, denote the number of red items in set X by $n^i_r(X)$, the number of blue items by $n^i_b(X)$, and the total number of items by $n^i(X)$. After all items have arrived and after some postprocessing, we will have

$$n^i_r(X) = \lfloor \alpha_i n^i(X) \rfloor \text{ for } X \in \{\mathcal{N}, \mathcal{B}, \mathcal{R}\} \text{ and each medium type } i. \tag{2}$$

▶ **Definition 2.** An unmixed bin is *red-compatible* with a newly arriving item if (1) the bin contains (provisionally) blue items of type i, the new item is of type j and will be colored red, and $b(i) \geq r(j)$, *or* (2) the bin contains a large item of size s, the new item is medium and has size at most $1 - s$. The definition for unmixed bins being blue-compatible to new items is completely analogous.

We say that a (mixed or unmixed) bin is *red-open* if it contains some non-provisionally red items but can still receive additional red items. We define *blue-open* analogously.

Like Super Harmonic algorithms, an Extreme Harmonic algorithm first tries to pack a red (blue) item into a red-open (blue-open) bin with items of the same type and color; then it tries to find a unmixed compatible bin; if all else fails, it opens a new bin. Of course, the definition of compatible has been extended compared to Super Harmonic (where this concept was not defined explicitly). Note that the choice of bin depends on the actions of Mark Items, since that algorithm fixes the colors of some items and bins.

1 $n^i \leftarrow n^i + 1$
2 **if** *p is medium, $\alpha_i > 0$, and there exists a red-compatible bin B with a large item* **then**
3 Place p in B and label it as bonus item. `// special case: existing bin`
4 $n^i \leftarrow n^i - 1$ `// we do not count this item for type` i
5 **else**
6 **if** $n_r^i < \lfloor \alpha_i n^i \rfloor$ **then** `// pack a (provisionally) red item`
7 **if** *there is a bin B with a bonus item of type i or there is a bin B with a bonus item of type j and $r(i) \leq b(j)$* **then**
8 Label the medium item in B as type i and color it red. It is no longer a bonus item.
9 $n^i \leftarrow n^i + \text{redfit}_i$ `// count medium item as type` i `item(s)`
10 $n_r^i \leftarrow n_r^i + \text{redfit}_i$
11 Pack(p, blue) `// since we now have` $n_r^i \geq \lfloor \alpha_i n^i \rfloor$ `again`
12 **else** Pack(p, red)
13 **else** `// pack a (provisionally) blue item`
14 **if** $b(i) = 0$ then pack p using Next Fit into pure bins of type i and color p blue.
15 **else** Pack(p, blue)
16 Update the marks and colors using Mark Items.

Algorithm 1: How the Extreme Harmonic framework packs a single item p of type $i < n$. At the beginning, we set $n_r^i \leftarrow 0$ and $n^i \leftarrow 0$ for $1 \leq i \leq n$.

1 Try the following types of bins to place p with (provisional) color c in this order:
2 - a mixed or unmixed c-open bin with items of type i and **definite** color c
3 - a c-compatible unmixed bin (the bin becomes mixed, its items' colors are fixed)
4 - a new unmixed bin
5 If p was packed into a new bin, p is medium and $\alpha_i > 0$, give p provisionally the color c, else give it the definite color c. If p got the definite color red, $n_r^i \leftarrow n_r^i + 1$.

Algorithm 2: The algorithm Pack(p, c) for packing an item p of type i with color $c \in \{\text{blue, red}\}$.

The new framework is formally described in Algorithm 1 and 2. We require $\alpha_i < 1/3$ for all types i. We discuss the changes from Super Harmonic one by one. All the changes stem from our much more careful packing of medium items.

As can be seen in Algorithm 2 (lines 2, 4 and 5), medium items that are packed into new bins are initially packed **one** per bin and given a provisional color. The goal of having provisionally colored items is to try and make sure that the *smallest* items of each type become red in the end. Thus, we wait until some number of these items have arrived, and then color the smallest one red (Figure 2).

When an item arrives, in many cases, we cannot postpone assigning it a color, since a c-open or c-compatible bin is already available (see lines 2–3 of Pack(p, c)). Additionally, we need to check right at the start whether a suitable large item has already arrived. We deal with this case in lines 2–4 of Algorithm 1. In this special case, we *ignore* the value α_i. We pack the medium item with the large item as if it was a red item, but we *do not count* it towards the total number of existing items of its type; instead we label it a **bonus item**. Bonus items do not have a color or mark, at least initially.

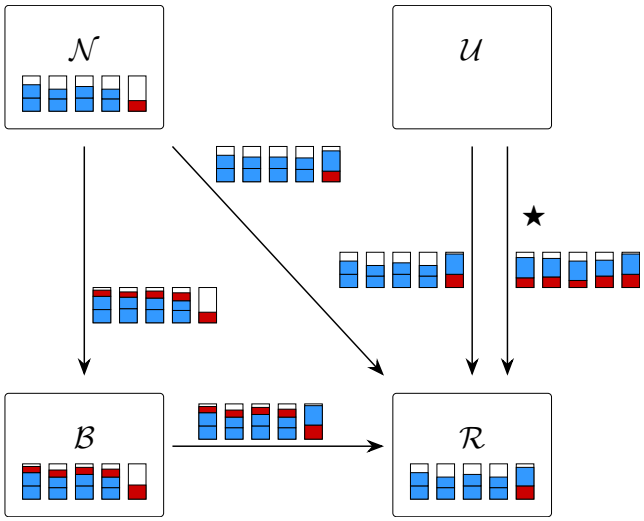


Figure 4 Reassigning marks after the input is complete. Items are sorted into their correct sets whenever possible, updating the marks that they received while the algorithm was running. Some item sizes are reduced. The bins next to the arrows indicate what sets of bins are being reassigned. The step marked with ★ takes place at the end of the postprocessing, after all other steps.

This means that we have (possibly temporarily) too many items of this type that are packed as red items (we do not count them towards the quantities n^i and n^i_r, but we do record that they exist). There are several ways that this can be fixed later on. Either, additional blue items of type i arrive and we can restore the correct ratio of red items. Or, some item of type j and size at most $1/3$ arrives that should be colored red and is compatible with blue items of type i. In this case, for our accounting, we replace the bonus item with redfit_j red items of type j, adjust the counts accordingly in lines 9–10, and color the new type j item blue.[3] Finally, it could also happen that some bonus items remain until the end; in this case we use careful post-processing so that each item does have a type and color at the end, and the ratio α_i is maintained. Note that we only modify item sizes for the analysis, and we only make items smaller, so the value of the optimal solution can only decrease and the implied competitive ratio can only increase as a result. Also note that allowing bonus items (i.e., occasionally packing too many items as red items) is essential to achieve a better competitive ratio; without this, we would get the same lower bound instances as before.

It can be seen that blue items of size at most $1/3$ are packed as in Super Harmonic. For red items of size at most $1/3$, we need to deal with existing bonus items in lines 9–10, and in line 3 of Pack(p, c), the provisional color of an existing item may be made permanent. Otherwise, the packing proceeds as in Super Harmonic. By the order in which existing bins are tried for packing new items, c-open bins always take precedence over other bins.

4 Postprocessing

After the algorithm has packed all items, we perform some postprocessing. For an overview of our changes of marks and sizes, see Figure 4. A formal version is given in the full paper.

[3] Note that the meanings of i and j are switched in the description of the algorithm for reasons of presentation.

▶ **Theorem 3.** *After postprocessing, (2) holds. Each blue item in $\mathcal{N}, \mathcal{R}$ and $\mathcal{B}$ is packed in a bin that contains two blue items. No bins with items in $\mathcal{N}$ or* red *items in $\mathcal{B}$ are mixed.*

In line 3 of Extreme Harmonic, bonus items are created. These are medium items which are packed together with a large blue item. Some of them may still be bonus when the algorithm has finished. Also, some of them may be labeled with a different type than the type they belong to according to their size. We call such items reduced items. In an additional postprocessing step, we split up reduced items into (possibly several) red items of the type with which we labeled the item. If any bonus items remain, we modify the packing that the algorithm outputs (for the analysis) by replacing some number of bins with a large blue item and a red medium item by the same number of bins with two blue medium items. Note that we only make items smaller, so all items still fit in their bins in both the optimal packing and the online packing. We finally achieve the following result.

▶ **Theorem 4.** *For each type i, we have $n_r^i \in [\lfloor \alpha_i n^i \rfloor - 3, \lfloor \alpha_i n^i \rfloor]$.*

5 Analysis using weights

Let $\mathcal{A}$ be an Extreme Harmonic algorithm. For analyzing the asymptotic performance ratio of $\mathcal{A}$, we will use the well-known technique of weighting functions: We assign weights to each item such that the number of bins that our algorithm uses in order to pack a specific input is equal to the sum of the weights of all items in this input. Then, we determine the maximum weight that can be packed in a single bin. Clearly, the offline algorithm cannot pack more weight than this in any of its bins, thus this maximum weight for a single bin gives us an upper bound on the competitive ratio.

Recall that the class of a red item of type i is $r(i)$ and the class of a blue item of type i is $b(i)$. Let $\mathfrak{r}$ be the smallest red item in a bin that has no blue items. Let the type of $\mathfrak{r}$ be ℓ, and $k = r(\ell)$. The weights of a non-large item p will depend on its class relative to k, and on its mark in case its class is k. The value of k (and the marks) become clear by running the algorithm. Note that the algorithm including the postprocessing does not depend on the weight functions in any way. There are $2K$ weighting functions in total, where $K = |\mathcal{D}|$ is the number of different spaces used for red items. For each k, w_k counts all the blue items, and v_k counts all the red items. The two weight functions of an item p of type i and mark m are given by the following table. The marks are only relevant for items of class k.

$w_k(p) = w_k(i,m)$		$v_k(p) = v_k(i,m)$	
$\frac{1-\alpha_i}{\text{bluefit}_i} + \frac{\alpha_i}{\text{redfit}_i}$	if $r(i) > k$	$\frac{1-\alpha_i}{\text{bluefit}_i} + \frac{\alpha_i}{\text{redfit}_i}$	if $b(i) < k$
$\frac{1-\alpha_i}{\text{bluefit}_i} + \frac{\alpha_i}{\text{redfit}_i}$	if $r(i) = k, m \in \{\mathcal{N}, \mathcal{B}, \emptyset\}$	$\frac{\alpha_i}{\text{redfit}_i}$	if $b(i) \geq k$
$\frac{1-\alpha_i}{\text{bluefit}_i}$	if $r(i) = k, m = \mathcal{R}$		
$\frac{1-\alpha_i}{\text{bluefit}_i}$	if $r(i) < k$		

▶ **Theorem 5.** *We have $\mathcal{A}(\sigma) \leq \max_{1 \leq k \leq K+1} \min\{\sum_{i=1}^n w_k(p_i), \sum_{i=1}^n v_k(p_i)\} + O(1)$ for any Extreme Harmonic algorithm $\mathcal{A}$ and any input σ.*

A *pattern* is a tuple $q = \{q_1, \ldots, q_m\}$ such that $\sum_{i=1}^m q_i t_{i+1} < 1$. Intuitively, a pattern describes the contents of a bin in the optimal offline solution. For a given weight function w, the weight of pattern q is $w(q) = w\left(1 - \sum_{i=1}^m q_i t_{i+1}\right) + \sum q_i w(t_i)$.

Denote the (finite) set of patterns by $\mathcal{Q}$. We can define an offline algorithm for a given input by a distribution χ over the patterns, where $\chi(q)$ indicates which fraction of the bins

are packed using pattern q. To show that a given Extreme Harmonic algorithm has performance ratio at most 1.5815 for input sequences with $\mathfrak{r}$ having class k, we must show

$$\frac{\min\{\sum_{i=1}^n w_k(p_i), \sum_{i=1}^n v_k(p_i)\}}{OPT(\sigma)} = \min\left\{\frac{\sum_{i=1}^n w_k(p_i)}{OPT(\sigma)}, \frac{\sum_{i=1}^n v_k(p_i)}{OPT(\sigma)}\right\}$$
$$\leq \min\left\{\sum_{q\in\mathcal{Q}} \chi(q) w_k(q), \sum_{q\in\mathcal{Q}} \chi(q) v_k(q)\right\} \leq 1.5815 \quad (3)$$

for all such inputs σ. As can be seen from this bound, the question now becomes: what is the distribution χ (the mix of patterns) that maximizes the minimum in (3)?

For this χ, the following constraints hold. Consider an input where $\mathfrak{r} > 1/3$. Let $m(q)$ be the number of $\mathcal{N}$-items of type ℓ in pattern q. Let q_1 be the pattern with an $\mathcal{N}$-item of type ℓ and an item of type i where $b(i) = k-1$. (Such an item is larger than $1-\mathfrak{r}$.) The parameters of the algorithm, in particular the type boundaries, must be such that this pattern is unique (i.e., no non-sand item can be added); it is easy to ensure this holds by setting an appropriate upper bound for the sand.

▶ **Lemma 6.** *If $\mathfrak{r} > 1/3$ and the type of $\mathfrak{r}$ is ℓ, then $m(q) \in \{0,1,2\}$ for all q, and $\chi(q_1) \leq \frac{1-\alpha_\ell}{1+\alpha_\ell} \sum_{q\neq q_1} \chi(q) m(q)$.*

For any j and q, let $n_j(q)$ be the number of items of type j in pattern q. Let q_2 be the pattern with a $\mathcal{B}$-item of the type of $\mathfrak{r}$ and an item larger than $1-\mathfrak{r}$. Like q_1, q_2 should be unique (this is easy to guarantee and check). Note that the patterns q_1 and q_2 are versions of the pattern shown in Figure 1.

▶ **Lemma 7.** *If $\mathfrak{r} > 1/3$, and ℓ is the type of $\mathfrak{r}$, $\frac{1-\alpha_\ell}{2}\chi(q_2) \leq \sum_{r(j)\leq b(\ell)} \sum_q \frac{\alpha_j}{\text{redfit}_j} \chi(q) n_j(q)$.*

Maximizing the minimum in (3) is the same as maximizing the first term under the condition that it is not larger than the second term—except that this condition might not be satisfiable, in which case we need to maximize the second term. We are led to consider two linear programs, which we will call LP_w^k and LP_v^k. Let $\mathcal{Q} = \{q_1, \ldots, q_{|Q|}\}$ and let $\chi_i = \chi(q_i), w_{ik} = w_k(q_i), n_{ij} = n_j(q_i), m_i = m(q_i)$. LP_w^k is the following linear program.

$$\begin{aligned}
\max \quad & \sum_{i=1}^{|\mathcal{Q}|} \chi_i w_{ik} && \text{// First term in (3)} && (4)\\
s.t. \quad & \chi_1 - \tfrac{1-\alpha_\ell}{2} \sum_{i=2}^{|\mathcal{Q}|} \chi_i m_i \leq 0 && \text{// Lemma 6} && (5)\\
& \tfrac{1-\alpha_\ell}{2}\chi_2 - \sum_{j:r(j)\leq b(\ell)} \sum_{i=3}^{|\mathcal{Q}|} \tfrac{\alpha_j}{\text{redfit}_j} \chi_i n_{ij} \leq 0 && \text{// Lemma 7} && (6)\\
& \sum_{i=3}^{|\mathcal{Q}|} \chi_i (w_{ik} - v_{ik}) \leq 0 && \text{// Bound on first term} && (7)\\
& \sum_{i=1}^{|\mathcal{Q}|} \chi_i \leq 1 && \text{// } \chi \text{ is a distribution} && (8)\\
& \chi_i \geq 0, 1 \leq i \leq |\mathcal{Q}| && \text{// } \chi \text{ is a distribution} && (9)
\end{aligned}$$

LP_w^k has a very large number of variables but only four constraints (apart from the non-negativity constraints). In (7) we use the following proposition.

▶ **Proposition 8.** $w_{1k} = v_{1k}, w_{2k} = v_{2k}$.

The dual DP_w^k is the following.

$$\begin{aligned}
\min \quad & y_4 && && (10)\\
s.t. \quad & y_1 + y_4 \geq w_{1k} && && (11)\\
& \frac{1-\alpha_\ell}{2} y_2 + y_4 \geq w_{2k} && && (12)\\
& -\tfrac{1-\alpha_\ell}{2} m_i y_1 - y_2 \sum_{j:r(j)\leq b(\ell)} \tfrac{\alpha_j}{\text{redfit}_j} n_{ij} + (w_{ik} - v_{ik}) y_3 + y_4 \geq w_{ik} && i = 3, \ldots, |\mathcal{Q}| && (13)\\
& y_i \geq 0 && i = 1, \ldots, 4 && (14)
\end{aligned}$$

If the objective value of DP_w^k as well as that of DP_v^k is at most some value y_4^* (or if one is infeasible), then y_4^* upper bounds the asymptotic performance ratio of our algorithm for this value of k by duality and by (3). It is easy to see that if for some feasible y^*, constraint (11) or (12) is not tight, then we can decrease y_1^* or y_2^* and still have a feasible solution. We therefore restrict our search to solutions for which (11) and (12) are tight. Given y_4^*, we then know the values of y_1^* and y_2^*. If the constraint (13) does not hold for pattern q_i and a given dual solution y^*, we have the following:

$$(1-y_3^*)w_{ik} + y_3^* v_{ik} + \frac{1-\alpha_\ell}{2} m_i y_1^* + y_2^* \sum_{j:r(j)\le b(\ell)} \alpha_j n_j > y_4^* \tag{15}$$

Note that we get exactly the same condition by considering DP_v^k due to symmetry.

Recall that w_{ik} and v_{ik} are just the sums of the respective weights of all the non-sand items in pattern q_i. Based on (15), we define a new weighting function $\omega(p)$ as follows.

$$\omega(p) = \begin{cases} (1-y_3^*)w_k(p) + y_3^* v_k(p) + \frac{1-\alpha_\ell}{2} y_1^* & \text{type of } p \text{ is } \ell \ (= \text{ type of } e) \\ (1-y_3^*)w_k(p) + y_3^* v_k(p) + y_2^* \alpha_j & \text{type of } p \text{ is } j, r(j) \le b(\ell) \\ (1-y_3^*)w_k(p) + y_3^* v_k(p) & \text{else} \end{cases}$$

The inequality (15) then turns into $\omega(q_i) > y_4^*$. For given y_4^*, we can therefore determine feasibility of (11)–(13) by using the ellipsoid method, fortunately for only one dimension: that is, we do a binary search for $y_3^* \in [0,1]$. For every value y_3^* that we consider, we solve a simple knapsack problem to determine $W = \max_{q\in\mathcal{Q}} \omega(q)$ using a dynamic program.

Summarizing the above discussion, proving that an algorithm is c-competitive can be done by running the described binary search for $k = 1, \ldots, K$ using $y_4^* = c$. Note that for $\mathfrak{r} \le 1/3$, we do not have conditions (5) and (6), and we can define $\omega(p) = (1-y_3^*)w_k(p) + y_3^* v_k(p)$ for all items.

For our algorithm Son Of Harmonic we have set initial values as follows. The last three columns contain item sizes and corresponding α_i values that were set manually, separated by semicolons. Numbers of the form $1/i$ until the value t_N are added automatically by our program if they are not listed below, but only up to 1/50; for very small items, we (automatically) merge some consecutive classes without loss of performance to speed up the binary search.

$c = \frac{15815}{10000}$	Item bounds and α values:	1/4;106/1000	3/20;0
$t_N = \frac{1}{2^{100}}$	33345/100000;0	8/39;8/100	29/200;0
$\gamma = \frac{2}{7}$	33340/100000;0	1/5;93/1000	1/7;16/100
(starting from $\frac{1}{14}$)	5/18;2/100	3/17;3/100	1/13;1/8
Last type before small type generation: $\frac{1}{50}$	7/27;105/1000	1/6;8/100	1/14;1/13

The remaining values α_i are set automatically using heuristics designed to speed up the search and minimize the resulting upper bound. In the range $(1/3, 1/2]$, we automatically generate item sizes (with corresponding α values and Δ_i values) that are less than t_N apart to ensure uniqueness of q_1 and q_2. The value γ specifies how much room is used by red items of size at most 1/14; larger items ($\le 1/3$) use at most 1/3 room. Our computer program and more information is available at `http://people.mpi-inf.mpg.de/~heydrich/extremeHarmonic/index.html`.

Acknowledgements. We thank the anonymous referees for their useful comments.

References

1 Luitpold Babel, Bo Chen, Hans Kellerer, and Vladimir Kotov. Algorithms for on-line bin-packing problems with cardinality constraints. *Discrete Applied Mathematics*, 143(1-3):238–251, 2004. doi:10.1016/j.dam.2003.05.006.

2 János Balogh, József Békési, György Dósa, Jirí Sgall, and Rob van Stee. The optimal absolute ratio for online bin packing. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1425–1438. SIAM, 2015. doi:10.1137/1.9781611973730.94.

3 János Balogh, József Békési, and Gábor Galambos. New lower bounds for certain classes of bin packing algorithms. In Klaus Jansen and Roberto Solis-Oba, editors, *Approximation and Online Algorithms – 8th International Workshop, WAOA 2010, Liverpool, UK, September 9-10, 2010. Revised Papers*, volume 6534 of *Lecture Notes in Computer Science*, pages 25–36. Springer, 2010. doi:10.1007/978-3-642-18318-8_3.

4 Donna J. Brown. A lower bound for on-line one-dimensional bin packing algorithms. Technical Report R-864, Coordinated Sci. Lab., Urbana, Illinois, 1979.

5 Edward G. Coffman, Michael R. Garey, and David S. Johnson. Approximation algorithms for bin packing: A survey. In D. Hochbaum, editor, *Approximation algorithms*. PWS Publishing Company, 1997.

6 Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1:349–355, 1981.

7 Michael R. Garey, Ronald L. Graham, and Jeffrey D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*, pages 143–150. ACM, 1972.

8 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the theory of of NP-Completeness*. Freeman and Company, San Francisco, 1979.

9 Michel X. Goemans and Thomas Rothvoß. Polynomiality for bin packing with a constant number of item types. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 830–839. SIAM, 2014. doi:10.1137/1.9781611973402.61.

10 David S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, MIT, Cambridge, MA, 1973.

11 David S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8:272–314, 1974.

12 Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 312–320, 1982.

13 Chung-Chieh Lee and D. T. Lee. A simple online bin packing algorithm. *Journal of the ACM*, 32:562–572, 1985.

14 Frank M. Liang. A lower bound for online bin packing. *Information Processing Letters*, 10:76–79, 1980.

15 Prakash V. Ramanan, Donna J. Brown, Chung-Chieh Lee, and D. T. Lee. Online bin packing in linear time. *Journal of Algorithms*, 10:305–326, 1989.

16 Michael B. Richey. Improved bounds for harmonic-based bin packing algorithms. *Discrete Applied Mathematics*, 34:203–227, 1991.

17 Thomas Rothvoß. Approximating bin packing within o(log OPT * log log OPT) bins. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 20–29. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.11.

18 Steve S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002.

19 Jeffrey D. Ullman. The performance of a memory allocation algorithm. Technical Report 100, Princeton University, Princeton, NJ, 1971.

20 André van Vliet. An improved lower bound for online bin packing algorithms. *Information Processing Letters*, 43:277–284, 1992.

21 Andrew C. C. Yao. New algorithms for bin packing. *Journal of the ACM*, 27:207–227, 1980.

Online Weighted Degree-Bounded Steiner Networks via Novel Online Mixed Packing/Covering*

Sina Dehghani[1], Soheil Ehsani[2], Mohammad Hajiaghayi[3], Vahid Liaghat[4], Harald Räcke[5], and Saeed Seddighin[6]

1 **University of Maryland, College Park, USA**
dehghani@cs.umd.edu
2 **University of Maryland, College Park, USA**
ehsani@cs.umd.edu
3 **University of Maryland, College Park, USA**
hajiagha@cs.umd.edu
4 **Stanford University, Palo Alto, USA**
vliaghat@stanford.edu
5 **Technische Universität München, München, Germany**
raecke@in.tum.de
6 **University of Maryland, College Park, USA**
saeedrez@cs.umd.edu

Abstract

We design the first online algorithm with poly-logarithmic competitive ratio for the *edge-weighted degree-bounded Steiner forest* (EW-DB-SF) problem and its generalized variant. We obtain our result by demonstrating a new generic approach for solving mixed packing/covering integer programs in the online paradigm. In EW-DB-SF, we are given an edge-weighted graph with a degree bound for every vertex. Given a root vertex in advance, we receive a sequence of terminal vertices in an online manner. Upon the arrival of a terminal, we need to augment our solution subgraph to connect the new terminal to the root. The goal is to minimize the total weight of the solution while respecting the degree bounds on the vertices. In the offline setting, *edge-weighted degree-bounded Steiner tree* (EW-DB-ST) and its many variations have been extensively studied since early eighties. Unfortunately, the recent advancements in the online network design problems are inherently difficult to adapt for degree-bounded problems. In particular, it is not known whether the fractional solution obtained by standard primal-dual techniques for mixed packing/covering LPs can be rounded online. In contrast, in this paper we obtain our result by using structural properties of the optimal solution, and reducing the EW-DB-SF problem to an exponential-size mixed packing/covering integer program in which every variable appears only once in covering constraints. We then design a generic *integral* algorithm for solving this restricted family of IPs.

As mentioned above, we demonstrate a new technique for solving mixed packing/covering integer programs. Define the *covering frequency* k of a program as the maximum number of covering constraints in which a variable can participate. Let m denote the number of packing constraints. We design an online *deterministic integral algorithm* with competitive ratio of $O(k \log m)$ for the mixed packing/covering integer programs. We prove the tightness of our result by providing a matching lower bound for any randomized algorithm. We note that our solution solely depends on m and k. Indeed, there can be exponentially many variables. Furthermore,

* Supported in part by NSF AF:Small grant CCF-1216698, NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHS/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 42; pp. 42:1–42:14

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

our algorithm directly provides an integral solution, even if the integrality gap of the program is unbounded. We believe this technique can be used as an interesting alternative for the standard primal-dual techniques in solving online problems.

1998 ACM Subject Classification I.1.2 Algorithms, G.2.2 [Graph Theory] Network problems

Keywords and phrases Online, Steiner Tree, Approximation, Competitive ratio

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.42

1 Introduction

Degree-bounded network design problems comprise an important family of network design problems since the eighties. Aside from various real-world applications such as vehicle routing and communication networks [6, 32, 38], the family of degree-bounded problems has been a testbed for developing new ideas and techniques. The problem of *degree-bounded spanning tree*, introduced in Garey and Johnson's *Black Book* of NP-Completeness [29], was first investigated in the pioneering work of Fürer and Raghavachari [15]. In this problem, we are required to find a spanning tree of a given graph with the goal of minimizing the maximum degree of the vertices in the tree. Let b^* denote the maximum degree in the optimal spanning tree. Fürer and Raghavachari give a parallel approximation algorithm which produces a spanning tree of degree at most $O(\log(n)b^*)$. This result was later generalized by Agrawal, Klein, and Ravi [1] to the case of degree-bounded Steiner tree (DB-ST) and degree bounded Steiner forest (DB-SF) problem. In DB-ST, given a set of terminal vertices, we need to find a subgraph of minimum maximum degree that connects the terminals. In the more generalized DB-SF problem, we are given pairs of terminals and the output subgraph should contain a path connecting each pair. Fürer and Raghavachari [16] significantly improved the result for DB-SF by presenting an algorithm which produces a Steiner forest with maximum degree at most $b^* + 1$.

The study of DB-ST and DB-SF was the starting point of a very popular line of work on various degree-bounded network design problems; e.g. [28, 31, 27, 22, 13] and more recently [14, 13]. One particular variant that has been extensively studied was initiated by Marathe et al. [28]: In the *edge-weighted degree-bounded spanning tree* problem, given a weight function over the edges and a degree bound b, the goal is to find a minimum-weight spanning tree with maximum degree at most b. The initial results for the problem generated much interest in obtaining approximation algorithms for the edge-weighted degree-bounded spanning tree problem [11, 10, 17, 23, 24, 25, 26, 34, 35, 36]. The groundbreaking results obtained by Goemans [18] and Singh and Lau [37] settle the problem by giving an algorithm that computes a minimum-weight spanning tree with degree at most $b+1$. Singh and Lau [27] generalize their result for the *edge-weighted Steiner tree* (EW-DB-ST) and *edge-weighted Steiner forest* (EW-DB-SF) variants. They design an algorithm that finds a Steiner forest with cost at most twice the cost of the optimal solution while violating the degree constraints by at most three.

Despite these achievements in the offline setting, it was not known whether degree-bounded problems are tractable in the *online setting*. The online counterparts of the aforementioned Steiner problems can be defined as follows. The underlying graph and degree bounds are known in advance. The demands arrive one by one in an online manner. At the arrival of a demand, we need to augment the solution subgraph such that the new demand is satisfied. The goal is to be competitive against an offline optimum that knows the demands in advance.

Recently, Dehghani et al. [12] explore the tractability of the Online DB-SF problem by showing that a natural greedy algorithm produces a solution in which the degree bounds are violated by at most a factor of $O(\log n)$, which is asymptotically *tight*. They analyze their algorithm using a dual fitting approach based on the combinatorial structures of the graph such as the toughness[1] factor. Unfortunately, they can also show that greedy methods are not competitive for the edge-weighted variant of the problem. Hence, it seems unlikely that the approach of [12] can be generalized to EW-DB-SF.

The *online edge-weighted Steiner connectivity* problems (with no bound on the degrees) have been extensively studied in the last decades. Imase and Waxman [21] use a dual-fitting argument to show that the greedy algorithm has a competitive ratio of $O(\log n)$, which is also asymptotically tight. Later the result was generalized to the EW SF variant by Awerbuch et al. [4] and Berman and Coulston [7]. In the past few years, various primal-dual techniques have been developed to solve the more general node-weighted variants [2, 30, 19], prize-collecting variants [33, 20], and multicommodity buy-at-bulk [9]. These results are obtained by developing various primal-dual techniques [2, 19] while generalizing the application of combinatorial properties to the online setting [30, 20, 9]. In this paper however, we develop a primal approach for solving *bounded-frequency mixed packing/covering integer programs*. We believe this framework would be proven useful in attacking other online packing and covering problems.

1.1 Our Results and Techniques

In this paper, we consider the online Steiner tree and Steiner forest problems at the presence of both edge weights and degree bounds. In the Online EW-DB-SF problem, we are given a graph $G = (V, E)$ with n vertices, edge-weight function w, degree bound b_v for every $v \in V$, and an online sequence of connectivity demands (s_i, t_i). Let w_{opt} denote the minimum weight subgraph which satisfies the degree bounds and connects all demands. Let $\rho = \frac{\max_e w(e)}{\min_{e:w(e)>0} w(e)}$.

▶ **Theorem 1.** *There exists an online deterministic algorithm which finds a subgraph with total weight at most $O(\log^3 n)w_{opt}$ while the degree bound of a vertex is violated by at most a factor of $O(\log^3(n)\log(n\rho))$.*

If one favors the degree bounds over total weight, one can find a subgraph with degree-bound violation $O(\log^3(n)\frac{\log(n\rho))}{\log\log(n\rho)})$ and total cost $O(\log^3(n)\frac{\log(n\rho))}{\log\log(n\rho)})w_{opt}$.

We note that the logarithmic dependency on ρ is indeed necessary. It follows from the result of [12] that the competitive ratio of any algorithm is either $\Omega(n)$ or $\Omega(\log \rho)$.

Our technical contribution for solving the EW-DB-SF problem is twofold. First by exploiting a structural result and massaging the optimal solution, we show a formulation of the problem that falls in the restricted family of *bounded-frequency mixed packing/cover IPs*, while losing only logarithmic factors in the competitive ratio. We then design a generic online algorithm with a logarithmic competitive ratio that can solve any instance of the bounded-frequency packing/covering IPs. In what follows, we describe our results in detail.

1.1.1 Massaging the optimal solution

Initiated by work of Alon et al. [2] on online set cover, Buchbinder and Naor developed a strong framework for solving packing/covering LPs *fractionally* online. For the applications

[1] The toughness of a graph is defined as $\min_{X \subseteq V} \frac{|X|}{|\text{CC}(G \setminus X)|}$; where for a graph H, $\text{CC}(H)$ denotes the collection of connected components of H.

of their general framework in solving numerous online problems, we refer the reader to the survey in [8]. Azar et al. [5] generalize this method for the fractional *mixed* packing and covering LPs. The natural linear program relaxation for EW-DB-SF, commonly used in the literature, is a special case of mixed packing/covering LPs: one needs to select an edge from every cut that separates the endpoints of a demand (covering constraints), while for a vertex we cannot choose more than a specific number of its adjacent edges (packing constraints). Indeed, one can use the result of Azar et al. [5] to find an online *fractional* solution with polylogarithmic competitive ratio. However, doing the rounding in an online manner seems very hard.

Offline techniques for solving degree-bounded problems often fall in the category of iterative and dependent rounding methods. Unfortunately, these methods are inherently difficult to adapt for an online settings since the underlying fractional solution may change dramatically in between the rounding steps. Indeed, this might be the very reason that despite many advances in the online network design paradigm in the past two decades, the natural family of degree-bounded problems has remained widely open. In this paper, we circumvent this by reducing EW-DB-ST to a novel formulation beyond the scope of standard online packing/covering techniques and solving it using a new online integral approach.

The crux of our IP formulation is the following structural property: Let (s_i, t_i) denote the i^{th} demand. We need to augment the solution Q_{i-1} of previous steps by buying a subgraph that makes s_i and t_i connected. Let G_i denote the graph obtained by contracting the pairs of vertices s_j and t_j for every $j < i$. Note that any $(s_i - t_i)$-path in G_i corresponds to a feasible augmentation for Q_{i-1}. Some edges in G_i might be already in Q_{i-1} and therefore by using them again we can save both on the total weight and the vertex degrees. However, in Section 2 we prove that there always exists a path in G_i such that even without sharing on any of the edges in G_i and therefore paying completely for the increase in the weight and degrees, we can approximate the optimal solution up to a logarithmic factor. This in fact, enables us to have a formulation in which the covering constraints for different demands are *disentangled*. Indeed, we only have one covering constraint for each demand. Unfortunately, this implies that we have exponentially many variables, one for each possible path in G_i. This may look hopeless since the competitive factors obtained by standard fractional packing/covering methods introduced by Buchbinder and Naor [8] and Azar et al. [5], depend on the logarithm of the number of variables. Therefore we come up with a new approach for solving this class of mixed packing/covering integer programs (IP).

1.1.2 Bounded-frequency mixed packing/covering IPs

We derive our result for EW-DB-ST by demonstrating a new technique for solving mixed packing/covering *integer* programs. We believe this approach could be applicable to a broader range of online problems. The integer program $\mathbb{IP}1$ describes a general mixed packing/covering IP with the set of integer variables $\mathbf{x} \in \mathbb{Z}^n_{\geq 0}$ and α. The packing constraints are described by a $m \times n$ non-negative matrix P. Similarly, the $q \times n$ matrix C describes the covering constraints. The *covering frequency* of a variable x_i is defined as the number of covering constraints in which x_i has a positive coefficient. The covering frequency of a mixed packing/covering program is defined as the maximum covering frequency of its variables.

$$\begin{aligned} \text{minimize} \quad & \alpha \ , \\ s.t. \quad & P\mathbf{x} \leq \alpha \ . \\ & C\mathbf{x} \geq 1 \ . \\ & \mathbf{x} \in \mathbb{Z}_{\geq 0}, \alpha \in \mathbb{R}_{>0} \ . \end{aligned} \tag{$\mathbb{IP}1$}$$

In the online variant of mixed packing and covering IP, we are given the packing constraints in advance. However the covering constraints arrive in an online manner. At the arrival of each covering constraint, we should *increase* the solution $\mathbf{x}$ such that it satisfies the new covering constraint. We provide a deterministic algorithm for solving online mixed packing/covering IPs.

▶ **Theorem 2.** *Given an instance of the online mixed packing/covering IP, there exists a deterministic integral algorithm with competitive ratio $O(k \log m)$, where m is the number of packing constraints and k is the covering frequency of the IP.*

We note that the competitive ratio of our algorithm is independent of the number of variables or the number of covering constraints. Indeed, there can be exponentially many variables.

Our result can be thought of as a generalization of the work of Aspnes et al. [3] on virtual circuit routing. Although not explicit, their result can be massaged to solve mixed packing/covering IPs in which all the coefficients are zero or one, and the covering frequency is one. They show that such IPs admit a $O(\log(m))$-competitive algorithms. Theorem 2 generalizes their result to the case with arbitrary non-negative coefficients and any bounded covering frequency.

We complement our result by proving a matching lower bound for the competitive ratio of any *randomized* algorithm. This lower bound holds even if the algorithm is allowed to return fractional solutions.

▶ **Theorem 3.** *Any randomized online algorithm A for integral mixed packing and covering is $\Omega(k \log m)$-competitive, where m denotes the number of packing constraints, and k denotes the covering frequency of the IP. This even holds if A is allowed to return a fractional solution.*

As mentioned before, Azar et al. [5] provide a fractional algorithm for mixed packing/-covering LPs with competitive ratio of $O(\log m \log d)$ where d is the maximum number of variables in a single constraint. They show an almost matching lower bound for deterministic algorithms. We distinguish two advantages of our approach compared to that of Azar et al.:

- The algorithm in [5] outputs a *fractional* competitive solution which then needs to be rounded online. For various problems such as Steiner connectivity problems, rounding a solution online is very challenging, even if offline rounding techniques are known. Moreover, the situation becomes hopeless if the integrality gap is unbounded. However, for bounded-frequency IPs, our algorithm directly produces an integral competitive solution. Thus it does not depend on rounding methods, and is applicable to problems with large integrality gap, or the problems for which it is shown that rounding methods do not preserve any approximation guarantee, and as such, the traditional approach fails.
- Azar et al. find the best competitive ratio with respect to the number of packing constraints and the size of constraints. Although these parameters are shown to be bounded in several problems, in many problems such as connectivity problems and flow problems, formulations with exponentially many variables are very natural. Our techniques provide an alternative solution with a tight competitive ratio, for formulations with bounded covering frequency.

1.2 Preliminaries

Let $G = (V, E)$ be an undirected graph of size n ($|V| = n$). Let $w : E \to \mathbb{Z}_{>0}$ be a function denoting the edge weights. For a subgraph $H \subseteq G$, we define $w(H) := \sum_{e \in E(H)} w(e)$. For every vertex $v \in V$, let $b_v \in \mathbb{Z}_{>0}$ denote the degree bound of v. Let $\deg_H(v)$ denote the degree of vertex v in subgraph H. We define the load $l_H(v)$ of vertex v w.r.t. H as

$\deg_H(v)/b_v$. In DB-SF we are given graph G, degree bounds, and k connectivity demands. Let σ_i denote the i-th demand. The i-th demand is a pair of vertices $\sigma_i = (s_i, t_i)$, where $s_i, t_i \in V$. In DB-SF the goal is to find a subgraph $H \subseteq G$ such that for each demand σ_i, s_i is connected to t_i in H, for every vertex $v \in V$, $l_H(v) \leq 1$, and $w(H)$ is minimized. In this paper without loss of generality we assume the demand endpoints are distinct vertices with degree one in G and degree bound infinity.

In the online variant of the problem, we are given graph G and degree bounds in advance. However the sequence of demands are given one by one. At arrival of demand σ_i, we are asked to provide a subgraph H_i, such that $H_{i-1} \subseteq H_i$ and s_i is connected to t_i in H_i.

The following integer program is a natural mixed packing and covering integer program for EW-DB-SF. Let $\mathcal{S}$ denote the collection of subsets of vertices that separate the endpoints of at least one demand. For a set of vertices S, let $\delta(S)$ denote the set of edges with exactly one endpoint in S. In $\mathbb{SF}\_\mathbb{IP}$, for an edge e, $x_e = 1$ indicates that we include e in the solution while $x_e = 0$ indicates otherwise. The variable α indicates an upper bound on the violation of the load of every vertex and an upper bound on the violation of the weight. The first set of constraints ensures that the load of a vertex is upper bounded by α. The second constraint ensures that the violation for the weight is upper bounded by α. The third set of constraints ensures that the endpoints of every demand are connected. Here we assume w_{opt} is known to the algorithm, although this can be waived by standard doubling techniques.

$$\text{minimize} \quad \alpha \ . \tag{$\mathbb{SF}\_\mathbb{IP}$}$$

$$\forall v \in V \quad \frac{1}{b_v} \sum_{e \in \delta(\{v\})} x_e \leq \alpha \ . \tag{1}$$

$$\frac{1}{w_{\text{opt}}} \sum_{e \in E} w(e) x_e \leq \alpha \ . \tag{2}$$

$$\forall S \subseteq \mathcal{S} \quad \sum_{e \in \delta(S)} x_e \geq 1 \ . \tag{3}$$

$$x_e \in \{0,1\}, \alpha \in \mathbb{Z}_{>0} \ .$$

1.3 Overview of the Paper

We begin Section 2 by providing a bounded frequency IP for EW-DB-SF. The IP is not a proper formulation of the problem, however, we can show that one can map feasible solutions of EW-DB-SF to feasible solutions of the IP without increasing the cost too much. In Section 3 we provide a deterministic algorithm for online bounded frequency mixed packing/covering IPs. In the full version of the paper, we also provide a matching lower bound for the competitive ratio of any randomized algorithm. Finally, in Section 4 we merge our techniques to obtain online polylogarithmic-competitive algorithms for EW-DB-SF.

2 Finding the Right Integer Program

In this section we design an online mixed packing and covering integer program for EW-DB-SF. We show this formulation is near optimal, i.e. any $f-$approximation for this formulation, implies an $O(f \log^2 n)$-approximation for EW-DB-SF. In Section 4 we show there exists an online algorithm that finds an $O(\log n)$-approximation solution for this IP and violates degree bounds by $O(\log^3 n \log w_{\text{opt}})$, where w_{opt} denotes the optimal weight.

First we define some notations. For a sequence of demands $\sigma = \langle (s_1, t_1), \ldots, (s_k, t_k) \rangle$, we define $R_\sigma(i)$ to be a set of i edges, connecting the endpoints of the first i demands.

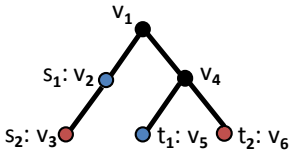


Figure 1 An example where every vertex has degree-bound 3 and every edge has weight 1. The first demand is (v_2, v_5) and the second demand is (v_3, v_6). The optimal solution for $\mathbb{SF}\_\mathbb{IP}$ is a subgraph, say H, with the set of all edges and vertices, i.e. $H = G$. However an optimal solution for $\mathbb{PC}\_\mathbb{IP}$ is: Two subgraphs H_1 for the first request which has edges $\{e(v_1, v_2), e(v_1, v_4), e(v_4, v_5)\}$ and H_2 for the second request which has edges $\{e(v_2, v_3), e(v_4, v_5), e(v_4, v_6)\}$. Note that $w(H) = 5$ and $w(H_1) + w(H_2) = 6$, since we have edge $e(v_4, v_5)$ in both H_1 and H_2. Moreover the number of edges incident with v_4 in the solution of $\mathbb{PC}\_\mathbb{IP}$ is 4, i.e. $\deg_{H_1}(v_4) + \deg_{H_2}(v_4) = 4$.

In particular $R_\sigma(i) := \bigcup_{j=1}^{i} e(s_j, t_j)$, where $e(s_j, t_j)$ denotes a direct edge from s_j to t_j. Moreover, we say subgraph H_i satisfies the connectivity of demand $\sigma_i = (s_i, t_i)$, if s_i and t_i are connected in graph $H_i \cup R_\sigma(i-1)$. Let $\mathcal{H}_i$ denote the set of all subgraphs that satisfy the connectivity of demand σ_i. In $\mathbb{PC}\_\mathbb{IP}$ variable α denotes the violation in the packing constraints. Furthermore for every subgraph $H \subseteq G$ and demand σ_i, there exists a variable $x^i_H \in \{0, 1\}$. $x^i_H = 1$ indicates we add the edges of H to the existing solution, at arrival of demand σ_i. The first set of constraints ensure the degree-bounds are not violated more than α. The second constraint ensures the weight is not violated by more than α. The third set of constraints ensure the endpoints of every demand are connected.

$$\text{minimize} \quad \alpha \ . \tag{$\mathbb{PC}\_\mathbb{IP}$}$$

$$\forall v \in V \qquad \frac{1}{b_v} \sum_{i=1}^{k} \sum_{H \subseteq G} \deg_H(v) x^i_H \leq \alpha \ . \tag{4}$$

$$\frac{1}{w_{\text{opt}}} \sum_{i=1}^{k} \sum_{H \subseteq G} w(H) x^i_H \leq \alpha \ . \tag{5}$$

$$\forall \sigma_i \qquad \sum_{H \in \mathcal{H}_i} x^i_H \geq 1 \ . \tag{6}$$

$$\forall H \subseteq G, 1 \leq i \leq k \qquad \mathbf{x}^i_H \in \{0, 1\} \ .$$

$$\alpha > 0 \ .$$

We are considering the online variant of the mixed packing and covering program. We are given the packing Constraints (4) and (5) in advance. At arrival of demand σ_i, the corresponding covering Constraint (6) is added to the program. We are looking for an online solution which is feasible at every online stage. Moreover the variables x_H should be monotonic, i.e. once an algorithm sets $x_H = 1$ for some H, the value of x_H is 1 during the rest of the algorithm. Figure 1 illustrates an example which indicates the difference between the solutions of $\mathbb{PC}\_\mathbb{IP}$ and $\mathbb{SF}\_\mathbb{IP}$.

Let popt and lopt denote the optimal solutions for $\mathbb{PC}\_\mathbb{IP}$ and $\mathbb{SF}\_\mathbb{IP}$, respectively. Lemma 4 shows that given an online solution for $\mathbb{PC}\_\mathbb{IP}$ we can provide a feasible online solution for $\mathbb{SF}\_\mathbb{IP}$ of cost popt.

▶ **Lemma 4.** *Given a feasible solution $\{\mathbf{x}, \alpha\}$ for $\mathbb{PC}\_\mathbb{IP}$, there exists a feasible solution $\{\mathbf{x}', \alpha\}$ for $\mathbb{SF}\_\mathbb{IP}$.*

In the rest of this section, we show that we do not lose much by changing $\mathbb{SF}\_\mathbb{IP}$ to $\mathbb{PC}\_\mathbb{IP}$. In particular we show $\mathsf{popt} \leq O(\log^2 n)\mathsf{lopt}$.

To this end, we first define the *connective* list of subgraphs for a graph G, a forest F, and a list of demands σ. We then prove an existential lemma for such a list of subgraphs with a desirable property for any $\langle G, F, \sigma\rangle$. With that in hand, we prove $\mathsf{popt} \leq O(\log^2 n)\mathsf{lopt}$. In what follows, we refer the reader to the full version of the paper for detailed proofs.

Given G, a list of demands $\sigma = \langle (s_1, t_1), \ldots, (s_k, t_k)\rangle$, and a forest $F \subseteq G$:

▶ **Definition 5.** Let $Q = \langle Q_1, Q_2, Q_3, \ldots, Q_k\rangle$ be a list of k subgraphs of F. We say Q is a *connective list* of subgraphs for $\langle G, F, \sigma\rangle$ iff for every $1 \leq i \leq k$ there exists no cut disjoint from Q_i that separates s_i from t_i, but does not separate any s_j from t_j for $j < i$.

The intuition behind the definition of connective subgraphs is the following: If Q is a connective list of subgraphs for an instance $\langle G, F, \sigma\rangle$ then for every i we are guaranteed that the union of all subgraphs $\cup_{j=1}^{i} Q_i$ connects s_i to t_i. In Lemma 6 we show for every $\langle G, F, \sigma\rangle$, there exists a connective list of subgraphs for $\langle G, F, \sigma\rangle$, such that each edge of F appears in at most $O(\log^2 n)$ subgraphs of Q.

▶ **Lemma 6.** *Let G be a graph and F be a forest in G. If σ is a collection of k demands $\langle (s_1, t_1), \ldots, (s_k, t_k)\rangle$, then there exists a connective list of subgraphs $Q = \langle Q_1, Q_2, \ldots, Q_k\rangle$ for $\langle G, F, \sigma\rangle$ such that every edge of F appears in at most $3\log^2 |V(F)|$ number of Q_i's.*

Proof. Here we give a sketch of the proof of lemma; we refer the reader to the full version for detailed proofs. We first prove a cost-minimization variant of the lemma. Consider an arbitrary weight vector $\hat{w} : F \to \mathbb{R}^{\geq 0}$. We argue that there is a connective list Q, such that $\sum_i \hat{w}(Q_i) \leq O(\log^2 n)\hat{w}(F)$. Let $\hat{H}_i = (V, F \cup R_\sigma(i), \hat{w}_i)$ denote a weighted graph for which $\hat{w}_i(e) = \hat{w}(e)$ for $e \in F$, and $\hat{w}_i(e) = 0$ for $e \in R_\sigma(i)$. Now we note that there is no cost-sharing among Q_i's in the goal $\sum_i \hat{w}(Q_i)$. Therefore the optimal choice for Q_i corresponds to the minimum-weight (s_i, t_i)-path in $\hat{H}_{i-1}$. Hence, we need to analyze the cost of these greedy choices.

Awerbuch et al. [4] showed that the greedy algorithm is indeed $O(\log^2 n)$-competitive for the edge-weighted Steiner forest problem. The standard greedy algorithm is slightly different from the greedy process we discussed above. In the greedy algorithm of Awerbuch et al., at time step i we choose a minimum-cost (s_i, t_i)-path in a graph in which there is a zero-cost edge between *any* pair of vertices in the same connected component of the current solution; not just the (s_j, t_j) pairs of the previous demands. However, in their analysis they only use the zero-cost edges among the terminals of a previous demand. This is indeed not surprising since we hardly have any control on the greedy choices other than the fact that they satisfy the demands. Therefore the following claim follows from the result of Awebuch et al.[2]:

▶ **Claim 7** (implicitly proven in Theorem 2.1 of [4]). *For any weight function $\hat{w}$ defined over F, there exists a connective list Q for which*

$$\sum_i \hat{w}(Q_i) \leq O(\log^2 n)\hat{w}(F).$$

However, Claim 7 is not enough for us. We need a solution in which every edge is used at most $O(\log^2 n)$ times, not just in an amortized sense. Indeed we can show that since

[2] There is also a lower bound of $\Omega(\log n)$ for the competitive ratio of the greedy algorithm. Closing the gap between this lower bound and the upper bound of $O(\log^2 n)$ for EW Steiner forest is an important open problem.

there is a solution for *every* weight function, we can have a *fractional* connective list Q in which every edge is used (fractionally) at most $O(\log^2 n)$ times. This implies that we have a fractional connective list. Finally, we provide a rounding argument which obtains an integral connective list by losing only a constant factor; which completes the proof of lemma. ◀

Finally, we can leverage Lemma 6 to show $\mathsf{popt} \leq O(\log^2 n)\mathsf{lopt}$. This shows we can use $\mathbb{PC}\_\mathbb{IP}$ as an online mixed packing/covering IP to obtain an online solution for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST losing a factor of $O(\log^2 n)$. In Section 4 we show this formulation is an online bounded frequency mixed packing/covering IP, thus we leverage our technique for such IPs to obtain a polylogarithmic-competitive algorithm for online EW-DB-SF.

3 Online Bounded Frequency Mixed Packing/Covering IPs

In this section we consider bounded frequency online mixed packing and covering integer programs. For every online mixed packing and covering IP with covering frequency k, we provide an online algorithm that violates each packing constraint by at most a factor of $O(k \log m)$, where m is the number of packing constraints. We note that this bound is independent of the number of variables, the number of covering constraints, and the coefficients of the mixed packing and covering program. Moreover the algorithm is for integer programs, which implies obtaining an integer solution does not rely on (online) rounding.

In particular we prove there exists an online $O(k \log m)$-competitive algorithm for any mixed packing and covering IP such that every variable has covering frequency at most k, where the covering frequency of a variable x_r is the number of covering constraints with a non-zero coefficient for x_r.

We assume that all variables are binary. One can see this is without loss of generality as long as we know every variable $x_r \in \{1, 2, 3, \ldots, 2^l\}$. Since we can replace x_r by l variables $y_r^1, \ldots, y_r^l$ denoting the digits of x_r and adjust coefficients accordingly. Furthermore, for now we assume that the optimal solution for the given mixed packing and covering program is 1. In Theorem 10 we prove that we can use a doubling technique to provide an $O(k \log m)$-competitive solution for online bounded frequency mixed packing and covering programs with any optimal solution. The algorithm is as follows. We maintain a family of subsets $\mathcal{S}$. Initially $\mathcal{S} = \emptyset$. Let $\mathcal{S}(j)$ denote $\mathcal{S}$ at arrival of C_{j+1}. For each covering constraint C_{j+1}, we find a subset of variables S_{j+1} and add S_{j+1} to $\mathcal{S}$. We find S_{j+1} in the following way. For each set of variables S, we define a cost function $\tau_S(\mathcal{S}(j))$ according to our current $\mathcal{S}$ at arrival of C_{j+1}. We find a set S_{j+1} that satisfies C_{j+1} and minimizes $\tau_S(\mathcal{S}(j))$. More precisely we say a set of variables S satisfies C_{j+1} if

- $\sum_{x_r \in S} C_{j+1,r} x_r \geq 1$, where $C_{j+1,r}$ denotes the coefficient of C_{j+1} for x_r.
- For each packing constraint P_i, $\sum_{x_r \in S} \frac{1}{k} P_{ir} \leq 1$.

Now we add S_{j+1} to $\mathcal{S}$ and for every $x_r \in S_{j+1}$, we set $x_r = 1$. We note that there always exists a set S that satisfies C_{j+1}, since we assume there exists an optimal solution with value 1. Setting S to be the set of all variables with value one in an optimal solution which have non-zero coefficient in C_{j+1}, satisfies C_{j+1}. It only remains to define $\tau_S(\mathcal{S}(j))$. But before that we need to define $\Delta_i(S)$ and $F_i(\mathcal{S}(j))$. For packing constraint P_i and subset of variables S, we define $\Delta_i(S)$ as $\Delta_i(S) := \sum_{x_r \in S} \frac{1}{k} P_{ir}$. For packing constraint P_i and $\mathcal{S}(j)$, let

$$F_i(\mathcal{S}(j)) := \sum_{S \in \mathcal{S}(j)} \Delta_i(S) \ . \tag{7}$$

Now let $\tau_S(\mathcal{S}(j)) = \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}$, where $\rho > 1$ is a constant to be defined later.

Algorithm 1

Input: Packing constraints P, and an online stream of covering constraints $C_1, C_2, \ldots$.
Output: A feasible solution for online bounded frequency mixed packing/covering.
Offline Process:

1: Initialize $\mathcal{S} \leftarrow \emptyset$.

Online Scheme; assuming a covering constraint C_{j+1} is arrived:

1: $S_{j+1} \leftarrow \arg\min_S\{\tau_S(\mathcal{S}(j)) \mid S \text{ satisfies } C_{j+1}\}$.
2: **for all** $x_r \in S_{j+1}$ **do**
3: $\quad x_r \leftarrow 1$.

Let $\mathbf{x}^*$ be an optimal solution, and $\mathbf{x}^*(j)$ denote its values at online stage j. We define $G_i(j)$ as

$$G_i(j) := \sum_{l=1}^{j} \sum_{r:C_{lr}>0} \frac{1}{k} x_r^* P_{ir} \ . \tag{8}$$

Now we define a potential function Φ_j for online stage j.

$$\Phi_j = \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \ , \tag{9}$$

where $\rho, \gamma > 1$ are constants to be defined later.

▶ **Lemma 8.** *There exist constants ρ and γ, such that Φ_j is non-increasing.*

Proof. We find ρ and γ such that $\Phi_{j+1} - \Phi_j \leq 0$. By the definition of Φ_j,

$$\Phi_{j+1} - \Phi_j = \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j+1))}(\gamma - G_i(j+1)) - \rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \ . \tag{10}$$

By Equation (7), $\rho^{F_i(\mathcal{S}(j+1))} - \rho^{F_i(\mathcal{S}(j))} = \rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}$. Moreover by Equation (8), $(\gamma - G_i(j+1)) - (\gamma - G_i(j)) = -\sum_{r:C_{j+1,r}>0} \frac{1}{k} x_r^* P_{ir}$. For simplicity of notation we define $B_i(j+1) := \sum_{r:C_{j+1,r}>0} \frac{1}{k} x_r^* P_{ir}$. Thus we can write Equation (10) as:

$$\Phi_{j+1} - \Phi_j = \sum_{i=1}^{m} \rho^{F_i(\mathcal{S}(j+1))}(\gamma - G_i(j) - B_i(j+1)) - \rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \tag{11}$$

$$= \sum_{i=1}^{m} (\gamma - G_i(j))(\rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j+1))} B_i(j+1) \quad \text{Since } G_i(j) \geq 0$$

$$\leq \sum_{i=1}^{m} \gamma(\rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j+1))} B_i(j+1) \qquad F_i(\mathcal{S}(j+1)) \geq F_i(\mathcal{S}(j))$$

$$\leq \sum_{i=1}^{m} \gamma(\rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j))} B_i(j+1) \ .$$

Now according to the algorithm for each subset of variables S' such that $\sum_{x_r \in S'} C_{j+1}(x_r) \geq 1$, either $\tau_S(\mathcal{S}(j)) \leq \tau_{S'}(\mathcal{S}(j))$ or there exists a packing constraint P_i such that $\Delta_i(S') > 1$. In

$B_i(j+1)$, we are considering variables x_r such that $x^*_e = 1$, thus for every P_i, $B_i(j+1) \leq 1$. Therefore setting S' to be the set of variables x_r such that $x^*_r = 1$ and $C_{j+1,r} > 0$, we have $\tau_S(\mathcal{S}(j)) \leq \tau_{S'}(\mathcal{S}(j))$. Thus $\sum_{i=1}^m \rho^{F_i(\mathcal{S}(j))+\Delta_i(S)} - \rho^{F_i(\mathcal{S}(j))} \leq \sum_{i=1}^m \rho^{F_i(\mathcal{S}(j))+B_i(j+1)} - \rho^{F_i(\mathcal{S}(j))}$. Therefore we can rewrite Inequality (11) as

$$\begin{aligned}\Phi_{j+1} - \Phi_j &\leq \sum_{i=1}^m \gamma(\rho^{F_i(\mathcal{S}(j))+B_i(j+1)} - \rho^{F_i(\mathcal{S}(j))}) - \rho^{F_i(\mathcal{S}(j))} B_i(j+1) \qquad (12)\\ &= \sum_{i=1}^m \rho^{F_i(\mathcal{S}(j))}(\gamma\rho^{B_i(j+1)} - \gamma - B_i(j+1)) \ .\end{aligned}$$

We would like to find ρ and γ such that Φ_j is non-increasing. We find ρ and γ such that for each packing constraint P_i, $\gamma\rho^{B_i(j+1)} - \gamma - B_i(j+1) \leq 0$. Thus

$$\gamma\rho^{B_i(j+1)} - \gamma \leq B_i(j+1) \qquad \text{Since } 0 \leq B_i(j+1) \leq 1 \qquad (13)$$

$$\gamma\rho B_i(j+1) - \gamma \leq B_i(j+1) \qquad \text{By simplifying} \qquad (14)$$

$$\rho \leq 1 + 1/\gamma \ . \qquad (15)$$

Thus if we set $\rho \leq 1 + 1/\gamma$, Φ_j is non-increasing, as desired. ◀

Now we prove Algorithm 1 obtains a solution of at most $O(k \log m)$.

▶ **Lemma 9.** *Given an online bounded frequency mixed packing covering IP with optimal value 1, there exists a deterministic integral algorithm with competitive ratio $O(k \log m)$, where m is the number of packing constraints and k is the covering frequency of the IP.*

Proof. By Lemma 8 for each stage j, $\Phi_{j+1} \leq \Phi_j$. Therefore $\Phi_j \leq \Phi_0 = \gamma m$. Thus for each packing constraint P_i,

$$\rho^{F_i(\mathcal{S}(j))}(\gamma - G_i(j)) \leq \gamma m \ . \qquad (16)$$

Thus,

$$\rho^{F_i(\mathcal{S}(j))} \leq \frac{\gamma m}{(\gamma - G_i(j))} \leq \frac{\gamma m}{\gamma - 1} \ . \qquad \text{Since } G_i(j) \leq 1 \qquad (17)$$

Thus we can conclude

$$F_i(\mathcal{S}(j)) \in O(\log m) \ . \qquad (18)$$

By definition of $F_i(\mathcal{S}(j))$, $F_i(\mathcal{S}(j)) = \sum_{S\in\mathcal{S}(j)} \Delta_i(S) = \sum_{S\in\mathcal{S}(j)} \sum_{x_r\in S} \frac{1}{k} P_{ir}$. Since each variable x_r is present in at most k sets, $\frac{1}{k} P_i \cdot \mathbf{x}(j) \leq F_i(\mathcal{S}(j))$. Thus by Inequality (18) $P_i\mathbf{x}(j) \in O(k \log m)$, which completes the proof. ◀

Finally we prove there exists an online $O(k \log m)$-competitive algorithm for bounded frequency online mixed packing and covering integer programs with any optimal value.

▶ **Theorem 10.** *Given an instance of the online mixed packing/covering IP, there exists a deterministic integral algorithm with competitive ratio $O(k \log m)$, where m is the number of packing constraints and k is the covering frequency of the IP.*

4 Putting Everything Together

In this section we consider the online mixed packing/covering formulation discussed in Section 2 for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST $\mathbb{PC}\_\mathbb{IP}$. In this section we show this formulation is an online bounded frequency mixed packing/covering IP. Therefore we our techniques discussed in Section 3 to obtain a polylogarithmic-competitive algorithm for ONLINE EDGE-WEIGHTED DEGREE-BOUNDED STEINER FOREST.

First we assume we are given the optimal weight w_{opt} as well as degree bounds. We can obtain the following theorem.

▶ **Theorem 11.** *Given the optimal weight w_{opt}, there exists an online deterministic algorithm which finds a subgraph with total weight at most $O(\log^3 n)w_{opt}$ while the degree bound of a vertex is violated by at most a factor of $O(\log^3 n)$.*

Proof. By Lemma 4, given a feasible online solution for $\mathbb{PC}\_\mathbb{IP}$ with violation α, we can provide an online solution for $\mathbb{SF}\_\mathbb{IP}$ with violation α. Moreover, in Section 2 we show that $\mathsf{popt} \leq O(\log^2 n)\mathsf{lopt}$. Thus given an online solution for $\mathbb{PC}\_\mathbb{IP}$ with competitive ratio f, there exists an $O(f \log n)$-competitive algorithm for ONLINE DEGREE-BOUNDED STEINER FOREST. We note that in $\mathbb{PC}\_\mathbb{IP}$ we know the packing constraints in advance. In addition every variable x_H^i has non-zero coefficient only in the covering constraint corresponding to connectivity of the i-th demand endpoints, i.e. the covering frequency of every variable is 1. Therefore by Theorem 10 there exists an online $O(\log m)$-competitive solution for $\mathbb{PC}\_\mathbb{IP}$, where m is the number of packing constraints, which is $n + 1$. Thus there exists an online $O(\log^3 n)$-competitive algorithm for ONLINE DEGREE-BOUNDED STEINER FOREST. This means the violation for both degree bounds and weight is of $O(\log^3 n)$. ◀

Finally if w_{opt} is not given, we show in the full version of the paper that by applying standard doubling techniques one can prove Theorem 1 using the result shown above.

References

1. Ajit Agrawal, Philip Nathan Klein, and R Ravi. How tough is the minimum-degree steiner tree?: A new approximate min-max equality. *Technical Report CS-91-49, Brown University*, 1991.
2. Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. *SIAM Journal on Computing*, 39(2):361–370, 2009.
3. James Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM (JACM)*, 44(3):486–504, 1997.
4. Baruch Awerbuch, Yossi Azar, and Yair Bartal. On-line generalized steiner problem. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 68–74, 1996.
5. Yossi Azar, Umang Bhaskar, Lisa Fleischer, and Debmalya Panigrahi. Online mixed packing and covering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 85–100. SIAM, 2013.
6. Fred Bauer and Anujan Varma. Degree-constrained multicasting in point-to-point networks. In *INFOCOM'95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE*, pages 369–376. IEEE, 1995.

7 Piotr Berman and Chris Coulston. On-line algorithms for steiner tree problems. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 344–353, 1997.

8 Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal: dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.

9 Deeparnab Chakrabarty, Alina Ene, Ravishankar Krishnaswamy, and Debmalya Panigrahi. Online buy-at-bulk network design. In *FOCS*, 2015.

10 Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. A push-relabel algorithm for approximating degree bounded msts. In *Proceedings of the 33rd international conference on Automata, Languages and Programming-Volume Part I*, pages 191–201. Springer-Verlag, 2006.

11 Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. What would edmonds do? augmenting paths and witnesses for degree-bounded msts. *Algorithmica*, 55(1):157–189, 2009.

12 Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online degree-bounded steiner network design. In *SODA*, 2016. URL: `http://web.stanford.edu/~vliaghat/uploads/notes.pdf`.

13 Alina Ene and Ali Vakilian. Improved approximation algorithms for degree-bounded network design problems with node connectivity requirements. *STOC*, 2014.

14 Takuro Fukunaga and R Ravi. Iterative rounding approximation algorithms for degree-bounded node-connectivity network design. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 263–272. IEEE, 2012.

15 Martin Fürer and Balaji Raghavachari. An NC approximation algorithm for the minimum degree spanning tree problem. In *Allerton Conf. on Communication, Control and Computing*, pages 274–281, 1990.

16 Martin Fürer and Balaji Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, 1994.

17 Michel X Goemans. Minimum bounded degree spanning trees. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 273–282. IEEE, 2006.

18 Michel X Goemans. Minimum bounded degree spanning trees. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 273–282, 2006.

19 Mohammad Taghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Online node-weighted steiner forest and extensions via disk paintings. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 558–567, 2013.

20 MohammadTaghi Hajiaghayi, Vahid Liaghat, and Debmalya Panigrahi. Near-optimal online algorithms for prize-collecting steiner problems. In *Automata, Languages, and Programming*, pages 576–587. Springer, 2014.

21 Makoto Imase and Bernard M Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.

22 Rohit Khandekar, Guy Kortsarz, and Zeev Nutov. On some network design problems with degree constraints. *Journal of Computer and System Sciences*, 79(5):725–736, 2013.

23 Philip N Klein, Radha Krishnan, Balaji Raghavachari, and R Ravi. Approximation algorithms for finding low-degree subgraphs. *Networks*, 44(3):203–215, 2004.

24 Jochen Könemann and R Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 537–546. ACM, 2000.

25 Jochen Könemann and R Ravi. Primal-dual meets local search: approximating mst's with nonuniform degree bounds. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 389–395. ACM, 2003.

26 Lap Chi Lau, Joseph Naor, Mohammad R Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM Journal on Computing*, 39(3):1062–1087, 2009.

27 Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. *SIAM Journal on Computing*, 42(6):2217–2242, 2013.

28 Madhav V Marathe, R Ravi, Ravi Sundaram, SS Ravi, Daniel J Rosenkrantz, and Harry B Hunt III. Bicriteria network design problems. *Journal of Algorithms*, 28(1):142–171, 1998.

29 R Garey Michael and S Johnson David. Computers and intractability: a guide to the theory of np-completeness. *WH Freeman & Co., San Francisco*, 1979.

30 Joseph Naor, Debmalya Panigrahi, and Mohit Singh. Online node-weighted steiner tree and related problems. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 210–219, 2011.

31 Zeev Nutov. Degree-constrained node-connectivity. In *LATIN 2012: Theoretical Informatics*, pages 582–593. Springer, 2012.

32 Carlos AS Oliveira and Panos M Pardalos. A survey of combinatorial optimization problems in multicast routing. *Computers & Operations Research*, 32(8):1953–1981, 2005.

33 Jiawei Qian and David P Williamson. An o (logn)-competitive algorithm for online constrained forest problems. In *Automata, Languages and Programming*, pages 37–48. Springer, 2011.

34 Balaji Raghavachari. Algorithms for finding low degree structures. In *Approximation algorithms for NP-hard problems*, pages 266–295. PWS Publishing Co., 1996.

35 R Ravi, Madhav V Marathe, SS Ravi, Daniel J Rosenkrantz, and Harry B Hunt III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31(1):58–78, 2001.

36 R Ravi and Mohit Singh. Delegate and conquer: An lp-based approximation algorithm for minimum degree msts. In *Automata, Languages and Programming*, pages 169–180. Springer, 2006.

37 Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 661–670, 2007.

38 Stefan Voß. Problems with generalized steiner problems. *Algorithmica*, 7(1):333–335, 1992.

Carpooling in Social Networks

Amos Fiat[1], Anna R. Karlin[*2], Elias Koutsoupias[†3], Claire Mathieu[4], and Rotem Zach[5]

1 **Department of Computer Science, Tel Aviv University, Tel Aviv, Israel**
2 **Department of Computer Science, University of Washington, Seattle, USA**
3 **Department of Computer Science, University of Oxford, Oxford, UK**
4 **Department of Computer Science, CNRS, Ecole Normale Superieure, Paris, France**
5 **Department of Computer Science, Tel Aviv University, Tel Aviv, Israel**

Abstract

We consider the online carpool fairness problem of [Fagin and Williams, 1983] in which an online algorithm is presented with a sequence of pairs drawn from a group of n potential drivers. The online algorithm must select one driver from each pair, with the objective of partitioning the driving burden as fairly as possible for all drivers. The *unfairness* of an online algorithm is a measure of the worst-case deviation between the number of times a person has driven and the number of times they would have driven if life was completely fair.

We introduce a version of the problem in which drivers only carpool with their neighbors in a given social network graph; this is a generalization of the original problem, which corresponds to the social network of the complete graph. We show that for graphs of degree d, the unfairness of deterministic algorithms against adversarial sequences is exactly $d/2$.

For random sequences of edges from planar graph social networks we give a [deterministic] algorithm with logarithmic unfairness (holds more generally for any bounded-genus graph). This does not follow from previous random sequence results in the original model, as we show that restricting the random sequences to sparse social network graphs may increase the unfairness.

A very natural class of randomized online algorithms are so-called static algorithms that preserve the same state distribution over time. Surprisingly, we show that any such algorithm has unfairness $\tilde{\Theta}(\sqrt{d})$ against oblivious adversaries. This shows that the *local random greedy* algorithm of [Ajtai et al, 1996] is close to optimal amongst the class of static algorithms. A natural (non-static) algorithm is global random greedy (which acts greedily and breaks ties at random). We improve the lower bound on the competitive ratio from $\Omega(\log^{1/3}(d))$ to $\Omega(\log d)$. We also show that the competitive ratio of global random greedy against adaptive adversaries is $\Omega(d)$.

1998 ACM Subject Classification F.1.2 Online Computation, F.2.0 Analysis of Algorithms and Problem Complexity

Keywords and phrases Online algorithms, Fairness, Randomized algorithms, Competitive ratio, Carpool problem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.43

"It was then that Hook bit him. Not the pain of this but its unfairness was what dazed Peter."

J. M. Barrie, *Peter Pan* (1911)

* The second author acknowledges the support of NSF grant CCF 1420381.
† The third author acknowledges the support of ERC Advanced Grant 321171 (ALGAME).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 43; pp. 43:1–43:13

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

1 Introduction

In multiple experimental studies involving hundreds of graduate students, Loewenstein, Thompson, and Bazerman [8] give evidence that individuals are strongly averse to outcomes where they are at a disadvantage relative to others. Moreover, albeit significantly less so, the grad students were also averse to outcomes where they have a relative advantage in payoff. Fehr and Schmidt [6] coined the phrase *inequity aversion* to describe this phenomena. Festinger [7] had much earlier introduced the concept of cognitive dissonance, and inequity aversion is modeled as a special case thereof. Supposedly, inequity aversion may lead individuals to make significant changes, including stopping interpersonal relationships where inequities arise.

The *carpool* problem, introduced by Fagin and Williams [5], is a stylized mathematical model in which one can study questions related to minimizing inequity. As described in [5], *"suppose that n people, tired of spending their time and money in gasoline lines, decide to form a carpool. We present a scheduling algorithm for determining which person should drive on any given day. We want a scheduling algorithm that will be perceived as fair by all the members."* A priori, it seems that fairness should not be hard to achieve, but – unfortunately – precise answers as to what extent one can avoid inequity have been sought over two decades with seemingly little progress.

Formally, each day t, a set of people $S_t \subseteq \{1, \ldots, n\}$ form a carpool. The goal is to choose who drives, so that on all days t, the overall driving burden to date has been partitioned fairly: Let $f_i(t)$ be driver i's fair share of the driving on day t, which is $1/|S_t|$ for each $i \in S_t$ and 0 otherwise. Define $F_i(t)$ to be driver i's fair share of the driving on all days up to day t, that is $F_i(t) = \sum_{\tau \le t} f_i(\tau)$, and let $D_i(t)$ be the number of times i has actually driven out of the first t days. For a particular sequence $\{S_t\}_{t=1}^{T}$, and algorithm for deciding who drives, we define

$$\text{the unfairness on day } t \;=\; \max_{\text{driver } i} |D_i(t) - F_i(t)|. \tag{1}$$

A carpool algorithm decides which person in S_t drives on day t; the *maximum unfairness* of the algorithm is

$$\max_{T \ge 1} \max_{\{S_t\}_{t=1}^{T}} [\text{unfairness on day } T]\,.$$

Notice that the definition of unfairness takes into account **all** trips i took, regardless of who i's companions were on that trip, that is, it is a global notion of fairness.

The offline version of the problem, when $\{S_t\}_{t=1}^{T}$ is known in advance, is easy: there is an algorithm that guarantees maximum unfairness of 1 and this is optimal (see, e.g. [10].)

Ajtai, Aspnes, Naor, Rabani, Schulman, and Waarts [1] studied the online version of problem, in which the algorithm must select a driver on day t, based only on the history up to time t. They obtained a number of extremely interesting results. First, they showed that, up to losing a factor of 2, one may assume that all the sets S_t consist of two persons. Thus, one can think of the process as a sequence of edge additions (*requests*), say $S_t = (i, j)$ at time t, to a multigraph on $\{1, \ldots, n\}$ (the people), with the algorithmic decision being one of choosing the orientation of the edge (towards the driver for that carpool). The goal then is to minimize

$$\max_{\text{vertex } i} |\,\text{indegree}(i) - \text{outdegree}(i)\,|\,.$$

Ajtai et al. obtained results for two different online settings: when the requests (carpools) are selected uniformly at random, and when the request sequence is selected by an oblivious

adversary that knows the algorithm, but not the outcome of any random choices the algorithm makes.

The first algorithm they considered was *Global Greedy*: on request (i, j), the driver among i and j with minimum unfairness (as defined in equation (1)) drives,; in case of a tie, the choice is arbitrary. For a uniformly random request sequence, they showed that for each t, *Global Greedy* has expected unfairness on that day of $O(\log \log n)$.

For the adversarial case, Ajtai et al showed that every deterministic algorithm has unfairness $\lfloor n/2 \rfloor$. They also showed that this is tight: *Global Greedy* has unfairness at most $n/2$ for every request sequence. They were able to obtain a better upper bound using *Randomized Local Greedy*: This algorithm considers each pair of drivers separately, and alternates which one drives each time they form a carpool. The only randomness is in the uniformly random choice of which of the two drives the very first time they carpool. They showed that *Randomized Local Greedy* has maximum unfairness equal to $\Theta(\sqrt{n \log n})$.

They conjectured that *Randomized Global Greedy*, the variant of *Global Greedy* in which ties are broken at random is much better, perhaps even polylog(n).

Finally, they proved that every randomized algorithm has maximum unfairness of at least $\Omega\left((\log n)^{1/3}\right)$.

Our Results

We study the carpool problem in the setting where the people involved belong to a social network G, and every request (carpool) is a pair of people that are connected in the social network, *i.e.* an edge of G. In this context, the work of [5, 1] can be seen as studying the special case where the social network is a clique.

We prove the following results for request sequences restricted to edges of a social network G with n vertices, and of maximum degree d.

Deterministic algorithms, adversarial requests

We show that for every deterministic algorithm there exists a request sequence on G resulting in unfairness of at least $\lfloor d/2 \rfloor$. This is tight: we give a deterministic algorithm that never generates unfairness greater than $d/2$ (Theorem 1).

What is most interesting about this result is that, in contrast to the case where the graph is complete, the optimal deterministic algorithm is *not* the *Global Greedy* algorithm. In fact, we show that for every connected G (irrespective of its maximum degree), there is a request sequence on which *Global Greedy* has worst-case unfairness $\geq \lfloor n/2 \rfloor$ (Theorem 2). Thus, *Global Greedy* can be a factor $\Omega(n)$ worse than the optimal deterministic algorithm (e.g., when the graph has constant degree).

Random requests

Our second set of results concerns random requests: We show that if the sequence of requests is generated by choosing edges of G uniformly at random, then the removal of edges from the graph can increase the unfairness for the *Global Greedy* algorithm: When G is a path, *Global Greedy* has expected unfairness at least $\Omega((\log n / \log \log n)^{1/3})$ (Theorem 5). This stands in contrast to the $O(\log \log n)$ upper bound of Ajtai et al when the graph G is a clique.

For a social network G of bounded genus (e.g., planar graphs, the torus, etc.) – we introduce the "star algorithm" - an algorithm with expected maximum unfairness $O(\log n)$ (Corollary 4).

Randomized algorithms, adversarial requests

Oblivious Adversaries: Oblivious adversaries determine the event sequence in advance, the algorithm may toss coins, and one considers the expected cost to the algorithm.

The results of Ajtai et al. show that *Randomized Local Greedy* gives maximum expected unfairness of $O(\sqrt{d \log d})$, since each vertex has degree at most d and the unfairness at each node is the expected value of the sum of at most d random variables that are equally likely to be 1 or -1. One can view this algorithm as maintaining an invariant probability distribution over unfairness configurations: for each t, regardless of the history of requests, each edge is oriented uniformly at random. In this sense, it is a *static algorithm*. Static algorithms form a very natural class of randomized online algorithms. Intuitively, they render an adversary powerless to construct a bad request sequence: every request sequence will perform the same against such an algorithm. We show that unfortunately, this intuition is incorrect and that the competitive ratio of *every* static algorithm is at least $\Omega(\sqrt{d})$.

As mentioned above, the *Randomized Global Greedy* algorithm has been conjectured to give a good competitive ratio against oblivious adversaries. We prove that *Randomized Global Greedy* has unfairness $\Omega(\log n)$ (on a clique of size n), improving upon the previous lower bound of $\Omega\left((\log n)^{1/3}\right)$ from [1]. This involves a rather complex proof, for which we only give a high level sketch here, the full details are available online.

Adaptive Adversaries: Adaptive adversaries determine the next event in the event sequence as a function of the previous responses of the online algorithm. *I.e.*, as a function of how the previous car pooling events were addressed.

We show that *no* randomized algorithm (static or not) has unfairness better than $d/4$ against an adaptive adversary [11].

Other Related Work

Another problem that can model fairness issues is Tijdeman's *chairman assignment problem* [10] where a chairman has to be appointed by a community of unequal groups. An axiomatic approach to the problem and its relationship to the Shapley value of a game was given in [9]. Generalizations of the carpool problem appear in [4, 3, 2].

Notation

In what follows, we often suppress the dependence on t in our notation for the unfairness of driver i at time t. Specifically, we use x_i to denote the unfairness of driver i at time t (i.e. $x_i := 2\big(D_i(t) - F_i(t)\big) = \text{indegree}(i) - \text{outdegree}(i)$), where t is understood. Note that $\sum_i x_i = \sum_i [\text{indegree}(i) - \text{outdegree}(i)] = 0$ at all times, and that we are assuming that all carpools are of size 2.

2 Deterministic algorithms

▶ **Theorem 1.** *Let G be a graph of maximum degree d. Then for every deterministic algorithm A, there exists a request sequence σ such that after A processes σ the unfairness is $\lceil d/2 \rceil$. This is tight: there is a deterministic algorithm such that after it processes every request sequence the unfairness is at most $\lceil d/2 \rceil$.*

Proof. For the first part, we restrict our sequences to the subgraph of G consisting of a maximum degree vertex and all of its neighbors, *i.e.,* a star with d leaves. For each

deterministic algorithm, we will prove that either the unfairness is unbounded or there is a sequence of requests for which the root of the star has unfairness $\pm\lceil d/2\rceil$.

Fix a deterministic algorithm. We will say that it is in state $\vec{x} = (x_1, \ldots, x_d)$ when the unfairness of leaf i is x_i (any ordering of the children is fine) and therefore the unfairness of the root is $-\sum_1^d x_i$. Let S be the set of states that can be reached by some request sequence. We can assume that for all i, $|x_i| \leq \lceil d/2\rceil$; if not, we are done. Therefore, S is bounded.

Select $\vec{x} \in S$ which minimizes $\varphi(\vec{x}) = x_1 + 2x_2 + \cdots + 2^{d-1}x_d$. Let r be a request sequence that reaches $\vec{x}$. If we extend r with request $(root, 1)$, then by the minimality of $\varphi(\vec{x})$, the online algorithm is not allowed to move to $(x_1 - 1, \vec{x}_{-1})$, so it will move to state $(x_1 + 1, \vec{x}_{-1})$. More generally, for $k \in [1, d]$, suppose that we extend r by requests (root plus leaves) $1, 2, \ldots, k$. Then since $\varphi(x_1 + 1, \ldots, x_{k-1} + 1, x_k - 1, x_{k+1}, \ldots, x_d) > \varphi(\vec{x})$, the online algorithm will end up at state $(x_1 + 1, \ldots, x_{k-1} + 1, x_k + 1, x_{k+1}, \ldots, x_d)$. If we then extend r by the sequence $1, 2, \ldots, d$, the online algorithm will move to state $\vec{x} + \vec{1}$. The first state has root unfairness $-\sum_1^d x_i$ and the second state has root unfairness $d - \sum_1^d x_i$, so one of those two numbers is greater than or equal to $\lceil d/2\rceil$ in absolute value.

For the second part, we show that any G can be oriented so that the indegree and outdegree of every vertex differ by at most 1. We then run the online algorithm that serves requests for the edge $\{i, j\}$ by alternatingly having i as a driver and j as a driver, starting with i iff the edge is directed from j to i. ◀

▶ **Theorem 2.** *Consider the Greedy algorithm and assume that ties are broken by an adversary. Then for any connected request graph, there exists a request sequence for which Greedy has unfairness at least* $\lfloor n/2\rfloor$.

Proof Sketch. The proof is by induction, restricting requests to a subgraph that is a tree. The idea is to increase the spread of the values by taking an edge $\{i, j\}$ between a subtree with low average unfairness and a subtree with high average unfairness. Then perform requests in the first subtree to maximize x_i, in the second subtree to minimize x_j. Then, if $x_i > x_j$, requesting edge $\{i, j\}$ will result in an even greater unbalance between the two subtrees. ◀

3 Random requests

3.1 Random requests on bounded genus graphs

We first prove the result for a star.

▶ **Theorem 3.** *Let G be a star with d leaves, root r and suppose that the requests are uniformly random. Then there is an algorithm such that for any time t, the unfairness at each leaf is at most one and the root has expected unfairness $O(1)$, with an exponential tail:* $\Pr(|Unfairness(r)| > k) \leq c\lambda^k$ *for some (c, λ) with $c > 0$ and $0 < \lambda < 1$.*

Sketch of Proof of Theorem 3. The "star algorithm" is the following:

1. Every leaf $1 \leq i \leq d$ has a counter $x_i \in \{-1, 0, 1\}$. Initially, set $x_i = 0$ for all $i \in S_0 = \{1, \ldots, d/2\}$.[1] Set $x_i = 1$ for all $i \in S_1 = \{d/2 + 1, \ldots, 3d/4\}$, and set $x_i = -1$ for all $i \in S_{-1} = \{3d/4 + 1, \ldots, d\}$. The root r maintains a counter $x_0 = -\sum_1^n x_i$, which is initially equal to zero.

[1] For simplicity, we will assume that d is a multiple of four; this is not necessary.

2. When a random request (r, i) arrives, if $x_i \neq 0$ then the algorithm orients the edge so that $x_i = 0$. If $x_i = 0$ and $x_0 \neq 0$ then the algorithm orients the edge so that $|x_0|$ decreases. If $x_i = x_0 = 0$ then the choice is random.

To analyze this, observe that, in expectation, half of the leaves have value 0. If the root has unfairness x, then a request to an edge connecting the root to a non-zero leaf might increase $|x|$, but any request to an edge connecting the root to a 0 leaf reduces $|x|$. Hence, with some work, we obtain a proof that $|x|$ is bounded on average and that its distribution has an exponential tail. ◀

▶ **Corollary 4.** *Let G be a bounded genus graph on n vertices and suppose that the requests are random. Then there is a deterministic algorithm with average maximum unfairness $O(\log n)$.*

Proof. We partition the edges of G into stars so as to ensure that each vertex is a leaf of at most a constant number of stars, and the center of exactly one star. To handle a sequence of requests, for each star, handle the subsequence of the requests that are edges of that star using the algorithm from Theorem 3. ◀

3.2 Poor performance of *Global Greedy* for random requests on the line

Ajtai et al. [1] prove that a uniformly random sequence of requests in the complete graph induces a unfairness of $O(\log \log n)$ for *Global Greedy* with any tie breaking rule. We show that this is not necessarily true when the possible requests are restricted to edges in the social network.

▶ **Theorem 5.** *Consider a sequence of independent requests drawn uniformly at random when the graph is a line. Then the* Global Greedy *algorithm, with ties broken at random, has expected unfairness $\Omega\left((\log n/\log\log n)^{1/3}\right)$.*

To prove this theorem, we will need the following lemma.

▶ **Lemma 6.** *When the graph G is a line with n vertices, there exists a sequence of length $f(n) = \Theta(n^3)$ that creates maximum unfairness of $n/2$ for the* Global Greedy *algorithm with adversarial tie-breaking. After $f(n)/2$ steps of that sequence the maximum unfairness is already at least $n/8$.*

Proof of Theorem 5. The idea is to consider multiple short segments of the line. The segments are of length k (to be determined below, about $\log^{1/3} n$), every segment has 2 extra vertices on the right, so there are about n/k different segments. We refer to the two extra vertices on the right of each segment as the "buffer zone" of the segment.

Consider a sequence of L random requests on the line, and focus on those, among those L requests, that fall into a particular segment and its buffer zone. A segment received Lk/n requests on average. Let L be such that $Lk/n = (3/4)f(k)$ and restrict attention to *good* segments, i.e. those segments that receive a number of requests in $[f(k)/2, f(k)]$.

With probability about $(1/k)^{k^3}$ the leftmost and rightmost buffer edges are never requested, and the requests to the segment follow the exact pattern of (a prefix of) the $f(k)$ requests required for the lower bound of Lemma 6. Conditioned upon this happening, the probability that the decisions made for tie breaking are as in Lemma 6 is at least $(1/2)^{f(k)}$, thus the probability that an unfairness of at least $k/8$ is reached in a particular good segment is $\geq (1/(2k))^{f(k)}$.

With probability at least $1/2$, the number of good segments is $\Omega(n/k)$, and by independence of the request sequence inside each segment, the probability that no segment gets

unfairness $k/8$ is at most $\left(1-\frac{1}{(2k)^{f(k)}}\right)^{\Omega(n/k)}$. This is at most $1/e$ for $\Omega(n/k) > (2k)^{\Theta(k^3)}$. Taking the logarithm gives $\log n = \Theta(k^3)\log(k)$, hence $k = \Theta\left((\log n/\log\log n)^{1/3}\right)$. ◀

4 Lower bounds against Adaptive Adversaries

In this section we show an adaptive adversary which achieves an $\Omega(d)$ lower bound for any randomized algorithm, on a star with d leaves.

Let $V = \{i_1, i_2, \ldots, i_d\}$ be the leaves of the star and let r be the root. Define the subsets:

$$\begin{aligned} V^+ &= \left\{i_j \in V | x_{i_j} > 0\right\}, \\ V^- &= \left\{i_j \in V | x_{i_j} < 0\right\}, \\ V^0 &= \left\{i_j \in V | x_{i_j} = 0\right\}. \end{aligned}$$

▶ Remark. For simplicity, we assume that d is divisible by 4, but this is not necessary.

Our adversary generates a sequence, until either $|x_r| \geq d/4$ or there is a leaf i_j such that $|x_{i_j}| \geq d/4$. The sequence is generated as follows:

1. If there is a leaf i_j such that $v \in V^0$ then issue the request (r, i_j).
2. If $V^0 = \emptyset$ and $|V^+| \geq d/2$ then let $V^+ = \{i_1, \ldots, i_k\}$ such that $x_{i_j} \leq x_{i_{j+1}}$ and issue the requests (r, i_j) in order of increasing j. Stop after processing a request increases x_{i_j}.
3. If $V^0 = \emptyset$ and $|V^-| > d/2$ then let $V^- = \{i_1, \ldots, i_k\}$ such that $x_{i_j} \geq x_{i_{j+1}}$ and issue the requests (r, i_j) in order of increasing j. Stop after processing a request decreases x_{i_j}.

The following lemma is proved in [11].

▶ **Lemma 7.** *The request sequence generated by the adaptive adversary described above is well defined,* i.e.*,*

1. *Exactly one of the three cases happens at each iteration.*
2. *In case 2 either the unfairness of a leaf increases or $x_r > d/4$.*
3. *In case 3 either the unfairness of a leaf decreases or $x_r < -d/4$.*

For our analysis we define the potential function

$$\Phi(V) = \sum_{i \in V} d^{|x_i|}.$$

Note that this potential function does *not* take into account x_r.

▶ **Lemma 8.** *After each iteration of the adversary's decision loop the potential $\Phi(V)$ increases by at least $d-1$ or $|x_r| \geq d/4$.*

Proof Sketch. We prove this by case analysis:

1. If there exists a leaf i_j such that $i_j \in V^0$ then the potential increases by exactly $d-1$.
2. If $|V^+| \geq d/2$ then from Lemma 7 the unfairness of one leaf was increased and at most the unfairness of $d-1$ leaves was decreased.
3. If $|V^-| > d/2$ then from Lemma 7 the unfairness of one leaf was decreased and at most the unfairness of $d-1$ leaves was increased.

In all three cases, the increase in potential caused by just one leaf is greater by more than $d-1$ than the decrease caused by the other leaves. ◀

▶ **Theorem 9.** *Assume that the social network is a star with d leaves. For any randomized algorithm, the adaptive adversary presented achieves unfairness $\Omega(d)$.*

Proof. From Lemma 7 the request sequence generated by the adversary is well defined and from Lemma 8, after each iteration of the adversary's decision loop either the potential $\Phi(V)$ increases by at least $d-1$ or $|x_r| \geq d/4$.

The initial potential is $\Phi(V) = d$. If after $(d \cdot d^{d/4-1})/(d-1) - 1$ iterations of the loop the inequality $|x_r| < d/4$ always holds then

$$\Phi(V) \geq d \cdot d^{d/4-1} - (d-1) + d \geq d \cdot d^{d/4-1} + 1.$$

So there must be at least one leaf with unfairness $\geq d/4$. ◀

5 Lower bounds against Oblivious Adversaries

5.1 Static Algorithms

Next, we consider *static algorithms* and bound the optimal unfairness that can be achieved by an algorithm in this class.

▶ **Definition 10.** A *state* is a vector $(x_i)_i$ where $x_i \in \mathbb{Z}$ represents the unfairness of vertex i. A randomized online algorithm is called *static* if there exists a probability distribution π over the set of states such that if the algorithm starts in π (*i.e.*, it starts at a state drawn according to π) then it remains in π after every possible request sequence.

Let $U(\vec{x})$ denote the maximum unfairness of state x. Then[2] the *expected maximum unfairness* of a static algorithm is

$$\sum_{\vec{x}} \pi(\vec{x}) \cdot U(\vec{x}).$$

As discussed in the introduction, *Randomized Local Greedy* preserves the distribution π in which each edge is oriented uniformly at random, and so *Randomized Local Greedy* is a static algorithm. The expected unfairness of every vertex i is $\Theta(\sqrt{d_i})$, where d_i is the degree of i, and the maximum unfairness is at most $O(\sqrt{n \log n})$ [1]. In particular, for the star with d leaves the unfairness of each leaf is at most 1, and the unfairness of the root is $\Theta(\sqrt{d})$.

▶ **Theorem 11.** *Assume G is the star with d leaves.*
1. *A slight variant of Local Greedy (Balanced Local Greedy) has optimal unfairness.*
2. *This value is $\Theta(\sqrt{d})$.*

Since states $\vec{x}$ that correspond to unfairness always satisfy $\sum_{i \in V} x_i = 0$, one of the coordinates can be inferred from the others. For compactness, we will drop the coordinate associated to the root and represent the state as a vector indexed by the leaves only.

To prove Theorem 11, we first state a property satisfied by the static distributions of static algorithms for the star with d leaves.

▶ **Lemma 12.** *Fix a static algorithm and let $\pi(\vec{x})$ be the corresponding static distribution. Fix a leaf j and $\vec{x}_{-j}$. Then the probability of states with even x_j must be equal to the probability of states with odd x_j:*

$$\sum_{k \in \mathbb{Z}} \pi(k, \vec{x}_{-i})(-1)^k = 0. \tag{2}$$

[2] There is a subtlety here in that the algorithm does not actually move initially to a state drawn from this distribution. Rather, it "pretends" to, and hence this definition of expected unfairness can be off by a factor of 2.

Proof. Suppose that the next request is the edge from root to leaf j. Conditioning on being on one of the states with given $\vec{x}_{-j}$, the state moves from even x_j to odd x_j and vice versa. Since the distribution before and after the request is the same, the two events must be equiprobable. ◀

Proof of Theorem 11. We define an infinite linear programming relaxation P with one variable $\pi(\vec{x})$ for each $\vec{x} \in \mathbb{Z}^d$. By Lemma 12, the expected unfairness of any static algorithm is at least

$$\min \sum_x \pi(\vec{x})U(\vec{x}) \text{ s.t. } \quad \begin{cases} \sum_{\vec{x}} \pi(\vec{x}) & = 1 \\ \sum_k \pi(k, \vec{x}_{-i})(-1)^k & = 0 \quad \forall i \forall \vec{x}_{-i} \end{cases}$$

Consider a vector $\vec{x}$ such that $x_i = 0$. Then, by elimination, the second constraint determines $\pi(\vec{x})$ in terms of the variables $\pi(k, \vec{x}_{-i})$ for $k \neq 0$, and the vectors $(k, \vec{x}_{-i})$ have one fewer zero coordinate than $\vec{x}$. Extending this by induction on the number of non-zero coordinates of $\vec{x}$, we show in Claim 13 below that each variable $\pi(\vec{x})$ with at least one zero coordinate in $\vec{x}$ can be expressed as a linear combination of the variables $\pi(\vec{y})$ with $\vec{y} \in (\mathbb{Z}_{\neq 0})^d$.

▶ **Claim 13.** *Let $Q(\vec{x}) = \{\vec{y} : x_i(y_i - x_i) = 0$ and $y_i \neq 0$, for all $i\}$. Then for all $\vec{x}$*

$$\pi(\vec{x}) = (-1)^{\#0(\vec{x})} \sum_{\vec{y} \in Q(\vec{x})} \pi(\vec{y})(-1)^{\sum_i y_i - x_i}. \tag{3}$$

Moreover, the set of Equations (3) *for all $\vec{x} \notin (\mathbb{Z}_{\neq 0})^d$ are equivalent to the set of Equations* (2) *in the linear program P.*

Proof. We transform Equations (2) to the form of Equations (3), inductively by the number of zero entries of $\vec{x}$.

The base case, in which $\vec{x}$ has no 0's is vacuous. For the induction step, assume that Equation (3) holds for all $\vec{x}$ that have at most k 0's. Consider some $\vec{x}$ that has $k+1$ 0's: $\vec{x} = (0, \vec{x}_{-i})$ for some x_{-i} that has k 0's. Solving (2) for $\pi(\vec{x})$ we get

$$\begin{aligned} \pi(0, \vec{x}_{-i}) &= -\sum_{y_i \neq 0} \pi(y_i, \vec{x}_{-i})(-1)^{y_i} \\ &= -\sum_{y_i \neq 0} (-1)^{\#0(\vec{x}_{-i})} \sum_{\vec{y}_{-i} \in Q(\vec{x}_{-i})} \pi(y_i, \vec{y}_{-i})(-1)^{\sum_{k \neq i} y_k - x_k}(-1)^{y_i} \\ &= (-1)^{\#0(\vec{x})} \sum_{\vec{y} \in Q(\vec{x})} \pi(\vec{y})(-1)^{\sum_i y_i - x_i}. \end{aligned}$$

Notice that there are $k+1$ different equations in the set of Equations (2), one for each 0 in $\vec{x}$, that are transformed into the same equation. ◀

To simplify the linear program P, substitute the right hand side of every equality of the form given in Equation (3) for $\pi(\vec{x})$ (for all $\vec{x} \notin (\mathbb{Z}_{\neq 0})^d$) into the constraint $\sum_{\vec{x}} \pi(\vec{x}) = 1$. This yields

$$\sum_{\vec{y} \notin (\mathbb{Z}_{\neq 0})^d} \alpha_{\vec{y}} \pi(\vec{y}) = 1, \tag{4}$$

for some constants $\{\alpha_{\vec{y}}\}$.

By construction, any solution to this equation can be extended to a solution to the set of Equations (2). This allows us to reduce the linear programing relaxation to one with a single constraint. Therefore it has an optimal solution with only one non-zero variable. Let $\pi(\vec{y}^*)$ denote this variable. By substituting back to (3), we get that $\pi(\vec{x})$ is zero unless

$$\vec{x} \in H := \{\vec{z} \quad \text{s.t. for all } i \quad z_i \in \{0, y_i^*\}\}.$$

Moreover, since for $\vec{x} \in H$, there is only one non-zero term on the right hand side of (3), we can conclude that $\forall \vec{x} \in H$, either $\pi(\vec{x}) = \pi(\vec{y}^*)$ or $\pi(\vec{x}) = -\pi(\vec{y}^*)$.

Next, observe that all coordinates of $\vec{y}^*$ must be odd. Indeed, if some y_i^* is even, then (3) implies that for every vector $\vec{x} \in H$ we have $\pi(y_i^*, \vec{x}_{-i}) = -\pi(0, \vec{x}_{-i})$, which implies that the sum of π on the vectors in H is 0, a contradiction. Thus, all coordinates of $\vec{y}^*$ are odd, and so $\pi(\vec{x}) = \pi(\vec{y}^*)$ for all $\vec{x} \in H$. Since these probabilities sum to 1, they must all be equal to $1/2^d$.

Let $|\vec{x}| = |\sum_i x_i|$. The expected unfairness of the root is

$$\sum_{\vec{x}} \pi(\vec{x})|\vec{x}| = \sum_{x \in H} \pi(\vec{x})|\vec{x}| = \sum_{x \in \{0, y_i^*\}^d} \frac{1}{2^d}|\vec{x}| = \mathrm{E}[|\sum_i y_i^* X_i|],$$

where X_i are 0-1 unbiased Bernoulli random variables. The following claim shows that this quantity is minimized when exactly half ($\lfloor d/2 \rfloor$, to be precise) of the y_i^*'s are 1 and the rest are -1, exactly as in the case of *Balanced Local Greedy*.

▶ **Claim 14.** *Let $y_1^*, \ldots, y_d^*$ be odd integers and $X_1, \ldots, X_n$ be independent unbiased Bernoulli variables. The expectation*

$$E[|\sum_i y_i^* X_i|]$$

is minimized when half ($\lfloor d/2 \rfloor$, to be precise) of the y_i^*'s are -1 and the remaining are +1. This minimum value is $\Theta(\sqrt{d})$.

It follows from the theorem that the optimal unfairness among static algorithms is at least equal to the unfairness at the root of the *Balanced Local Greedy*. Since its unfairness on the leaves is at most one, *Balanced Local Greedy* has almost optimal unfairness among the static algorithms, within an additive term of 1. In fact, the additive term can be reduced to $O(1/\sqrt{d})$.

The main result of this section dashes any hopes to find a static algorithm with small unfairness. However, many natural algorithms – among them the *Global Greedy* algorithm – are not static and we hope that they will be shown to have small unfairness (substantially less than $O(\sqrt{d})$).

5.2 Bounds on *Randomized Global Greedy*

A very natural algorithm, *Randomized Global Greedy* is a candidate algorithm to give small competitive ratios against an oblivious algorithm. This algorithm is greedy, choosing the driver with the lower number of times to her credit, and breaking ties at random. The difference between *Randomized Global Greedy* and *Randomized Local Greedy* is that *Randomized Local Greedy* only uses randomization initially to determine the initial state and then alternates drivers whenever the same pair reappear.

Previously, the lower bound on the competitive ratio of *Randomized Global Greedy* was $\Omega(\log^{1/3} n)$ (on a clique) whereas the upper bound was $O(n)$. We improve the lower bound

to $\Omega(\log n)$. To do this we make use of a potential function, the sum over drivers of the differences between the number of trips minus the number of times that the driver drove.

The key idea is to repeatedly generate unfairness and "push" it to some target set of drivers. This process works essentially as follows:

1. Apply a sequence of requests to generate potential in some set of drivers.
2. "Push" the potential to a target set of drivers.
3. Repeat.
4. Eventually, the target set will have high potential, which implies high unfairness.

We present here a sketch of the lower bound proof, full details of this process can be seen online at [11]. In this MSc thesis (of one of the authors) one can also see similar techniques applied to other settings.

First, we begin with a definition of the potential function.

▶ **Definition 15.** Let $i_1, i_2, \ldots, i_n$ be n vertices where x_{i_j} is the unfairness of vertex i_j. Assume that there is a value $u \in \mathbb{Z}$ such that for all $1 \leq j \leq n$ it holds that $x_{i_j} \in \{u-1, u, u+1\}$. Define the potential with respect to base unfairness u as

$$\Phi_u(\{x_{i_1}, x_{i_2}, \ldots, x_{i_n}\}) = \sum_{1 \leq j \leq n} (x_{i_j} - u).$$

Now that we have defined the potential function, we define "pushing".

▶ **Definition 16.** Let $v', v'', r \in V$ be distinct vertices and let $S = \{i_1, i_2, i_3, \ldots, i_n\}$ be a set of n vertices such that $v', v'', r \notin S$.

Let $\alpha \in \mathbb{N}$ be a large number, as a function of n. The sequence $\text{push}_u(\{v', v''\}, S, r)$ is composed of three subsequences:

$$\begin{aligned}
&\text{push}^{s1}_u(\{v', v''\}, S, r) = (r, v'), (r, v''),\\
&\text{push}^{d}_u(\{v', v''\}, S, r) = (r, i_n), (r, i_n), (r, i_{n-1}), (r, i_{n-1}), \ldots, (r, i_1), (r, i_1),\\
&\text{push}^{s2}_u(\{v', v''\}, S, r) = (r, v''), (r, v').
\end{aligned}$$

And,

$$\begin{aligned}
\text{push}_u(\{v', v''\}, S, r) = \ &\big(\text{push}^{s1}_u(\{v', v''\}, S, r) \| \\
&\text{push}^{d}_u(\{v', v''\}, S, r) \| \text{push}^{s2}_u(\{v', v''\}, S, r)\big)^{\alpha}.
\end{aligned}$$

▶ Remark. Notation: (v, w) is a shorthand for requesting that v and w carpool together.

We use the following property thats holds after the *Randomized Global Greedy* algorithm processes $\text{push}_u(\{v', v''\}, S, r)$:

Define $\Phi_{\text{init}}(S) = \Phi(S)$ and $\Phi_{\text{init}}(\{v', v''\}) = \Phi(\{v', v''\})$ as the potentials before processing the sequence $\text{push}_u(\{v', v''\}, S, r)$. Define $\Phi_{\text{end}}(S) = \Phi(S)$ and $\Phi_{\text{end}}(\{v', v''\}) = \Phi(\{v', v''\})$ as the potentials after processing the sequence.

▶ **Lemma 17.** *Assume that: (a) For some $u \in \mathbb{Z}$, $x_r = u$, (b) That $x'_v, x''_v \in \{u-1, u+1\}$, and (c) For all $i \in S$, $x_i \in \{u-1, u+1\}$.*

If $|(x_{v'} - u) + (x_{v''} - u) + \Phi_{\text{init}}(S)| \leq |S|$ then, with high probability, after the Randomized Global Greedy *algorithm processes the sequence $\text{push}_u(\{v', v''\}, S, r)$ the equality $\Phi_{\text{end}}(S) = \Phi_{\text{init}}(S) + \Phi_{\text{init}}(\{v', v''\})$ holds.*

Proof Sketch. Observe that if $x_{v'} = u+1$, $x_{v''} = u+1$ and $x_r = u$ then after processing the subsequence $(r, v'), (r, v'')$, with probability $1/2$, $x_r = u+2$. Now observe that if $x_r = u+2$ and for some j, $x_{i_j} = u-1$ then after processing the two requests $(r, i_j), (r, i_j)$ it holds that $x_{i_j} = u+1$ and $x_r = u$. Thus the potential of $\{v', v''\}$ was "pushed" into i_j.

A similar thing happens if $x_{v'} = u-1$, $x_{v''} = u-1$ and $x_{i_j} = u+1$. The constant α is chosen to be large such this occurs with high probability.

The full proof is available online in [11]. ◀

Now, using this push sequence we "accumulate" a large potential (in respect to base unfairness u) in a specific set A such that $|A| = O(n)$. *I.e.*, we define a sequence such that after *Randomized Global Greedy* processes it $Pr(|\Phi_u(A)| > |A/2|) > 1/2$. In essence, this is done by repeatedly using a sequence that creates unfairness in a small set G and "pushing" this unfairness into A. This is called "accumulation".

Another useful subsequence is that of "distillation", which takes the potential of a set and, with high probability, pushes it into a subset. Let r be a vertex and $S = \{i_1, i_2, i_3, \ldots, i_m\}$ be a set of m vertices such that m is even and $r \notin S$. The $\text{distill}_u(S, r)$ event sequence is

$$\text{distill}_u(S, r) = \text{push}_u(\{i_1, i_2\}, \{i_3, \ldots, i_m\}, r) \| \text{push}_u(\{i_3, i_4\}, \{i_5, \ldots, i_m\}, r) \| \cdots \| \text{push}_u(\{i_{m-3}, i_{m-2}\}, \{i_{m-1}, i_m\}, r).$$

Define the tail of S, S_ℓ, as $S_\ell = \{i_k \in S | k \geq \ell\}$.

▶ **Lemma 18.** *With high probability, after the* Randomized Global Greedy *algorithm processes the sequence* $\text{distill}_u(S)$

$$\Phi_u(S_{|S|-|\Phi_u(S)|+1}) = \Phi_u(S).$$

Proof Sketch. This stems from repeatedly applying Lemma 17 to increasingly smaller sets. The full proof is available online in [11]. ◀

Now using the sequences "accumulation" and "distillation" we are able to create (with probability greater than $1/2$) a set A such that $|A| = O(n)$ and $|\Phi_u(A) = |A|$. The latter is equivalent to all the unfairnesses of vertices in A being equal to either $u-1$ or $u+1$.

Now split A into three different sets, A_1, A_2, and A_3 and do "accumulation" and "distillation" to each set individually. Now, with some probability, one of these subsets has unfairnesses which are all equal to either $u-2$ or $u+2$. Repeat this process. In [11] we prove that this can be repeated $\Omega(\log n)$ times with constant probability. And thus the following theorem is proved:

▶ **Theorem 19.** *The sequence above achieves an expected unfairness of* $\Omega(\log n)$ *for the* Randomized Global Greedy *algorithm when run on a clique.*

6 Open Questions

The outstanding open questions that follow immediately from this work are:

- Is there any randomized algorithm with unfairness $o(\sqrt{d})$ on the star with d leaves?
- Does *Randomized Global Greedy* have $o(n)$ unfairness on the star or on the line?

At this point we have no non-trivial upper bound on the star. The best algorithm we know is *Randomized Local Greedy*, which achieves $\sqrt{n}$ unfairness.

Acknowledgments. We are grateful to Nimrod Fiat and Clemens von Stengel for their help during the early stages of this work.

References

1. Miklos Ajtai, James Aspnes, Moni Naor, Yuval Rabani, Leonard J Schulman, and Orli Waarts. Fairness in scheduling. *Journal of Algorithms*, 29(2):306–357, 1998.
2. Amihood Amir, Oren Kapah, Tsvi Kopelowitz, Moni Naor, and Ely Porat. The family holiday gathering problem or fair and periodic scheduling of independent sets. *CoRR*, abs/1408.2279, 2014. URL: http://arxiv.org/abs/1408.2279.
3. Joao Pedro Boavida, Vikram Kamat, Darshana Nakum, Ryan Nong, Chai Wah Wu, and Xinyi Zhang. Algorithms for the carpool problem. *IMA Preprint Series*, pages 2133–6, 2006.
4. Don Coppersmith, Tomasz Nowicki, Giuseppe Paleologo, Charles Tresser, and Chai Wah Wu. The optimality of the online greedy algorithm in carpool and chairman assignment problems. *ACM Trans. Algorithms*, 7(3):37:1–37:22, July 2011. doi:10.1145/1978782.1978792.
5. Ronald Fagin and John H Williams. A fair carpool scheduling algorithm. *IBM Journal of Research and development*, 27(2):133–139, 1983.
6. Ernst Fehr and Klaus M Schmidt. A theory of fairness, competition, and cooperation. *Quarterly journal of Economics*, pages 817–868, 1999.
7. L. Festinger. *A Theory of Cognitive Dissonance*. Mass communication series. Stanford University Press, 1962.
8. George F. Loewenstein, Leigh L. Thompson, and Max H. Bazerman. Social utility and decision making in interpersonal contexts. *Journal of Personality and Social Psychology*, 57(3):426–441, 1989. doi:10.1037/0022-3514.57.3.426.
9. Moni Naor. On fairness in the carpool problem. *Journal of Algorithms*, 55(1):93 – 98, 2005. doi:10.1016/j.jalgor.2004.05.001.
10. R. Tijdeman. The chairman assignment problem. *Discrete Mathematics*, 32(3):323 – 330, 1980. doi:10.1016/0012-365X(80)90269-1.
11. Zach and Fiat. Lower bounds for carpooling, 2015. URL: https://www.cs.tau.ac.il/~fiat/rotem_msc_thesis.pdf.

An Improved Analysis of the ER-SpUD Dictionary Learning Algorithm*

Jarosław Błasiok[†1] **and Jelani Nelson**[‡2]

1 **Harvard University, Cambridge, MA, USA**
jblasiok@g.harvard.edu
2 **Harvard University, Cambridge, MA, USA**
minilek@g.harvard.edu

Abstract

In *dictionary learning* we observe $Y = AX + E$ for some $Y \in \mathbb{R}^{n\times p}$, $A \in \mathbb{R}^{m\times n}$, and $X \in \mathbb{R}^{m\times p}$, where $p \geq \max\{n, m\}$, and typically $m \geq n$. The matrix Y is observed, and A, X, E are unknown. Here E is a "noise" matrix of small norm, and X is column-wise sparse. The matrix A is referred to as a *dictionary*, and its columns as *atoms*. Then, given some small number p of samples, i.e. columns of Y, the goal is to learn the dictionary A up to small error, as well as the coefficient matrix X. In applications one could for example think of each column of Y as a distinct image in a database. The motivation is that in many applications data is expected to sparse when represented by atoms in the "right" dictionary A (e.g. images in the Haar wavelet basis), and the goal is to learn A from the data to then use it for other applications.

Recently, the work of [24] proposed the dictionary learning algorithm **ER-SpUD** with provable guarantees when $E = 0$ and $m = n$. That work showed that if X has independent entries with an expected θn non-zeroes per column for $1/n \lesssim \theta \lesssim 1/\sqrt{n}$, and with non-zero entries being subgaussian, then for $p \gtrsim n^2 \log^2 n$ with high probability **ER-SpUD** outputs matrices A', X' which equal A, X up to permuting and scaling columns (resp. rows) of A (resp. X). They conjectured that $p \gtrsim n \log n$ suffices, which they showed was information theoretically necessary for *any* algorithm to succeed when $\theta \simeq 1/n$. Significant progress toward showing that $p \gtrsim n \log^4 n$ might suffice was later obtained in [17].

In this work, we show that for a slight variant of **ER-SpUD**, $p \gtrsim n \log(n/\delta)$ samples suffice for successful recovery with probability $1 - \delta$. We also show that without our slight variation made to **ER-SpUD**, $p \gtrsim n^{1.99}$ samples are required even to learn A, X with a small success probability of $1/\mathit{poly}(n)$. This resolves the main conjecture of [24], and contradicts a result of [17], which claimed that $p \gtrsim n \log^4 n$ guarantees high probability of success for the original **ER-SpUD** algorithm.

1998 ACM Subject Classification I.2.6 Learning

Keywords and phrases dictionary learning, stochastic processes, generic chaining

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.44

1 Introduction

The *dictionary learning* or *sparse coding* problem is defined as follows. There is a hidden set of vectors $a_1, a_2, \ldots a_m \in \mathbb{R}^n$ (called a "dictionary"), with $\mathit{span}\{a_1, \ldots a_m\} = \mathbb{R}^n$. We

* A full version of the paper is available at http://arxiv.org/abs/1602.05719.
† JB was supported by NSF grant IIS-1447471.
‡ JN was supported by NSF grant IIS-1447471 and CAREER CCF-1350670, ONR grant N00014-14-1-0632 and Young Investigator N00014-15-1-2388, and a Google Faculty Research Award.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 44; pp. 44:1–44:14


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

are given a sequence of samples $y_i = Ax_i + \epsilon_i$, where each x_i is a sparse vector and ϵ_i is noise. In other words each y_i is close to a linear combination of few vectors a_k. The goal is to recover both matrix A and the sparse representations x_i. We can write it as a matrix equation, $Y = AX + E$, where the vectors y_i are the columns of Y, and x_i are columns of X. Let $A \in \mathbb{R}^{n \times m}$ and $X \in \mathbb{R}^{m \times p}$. Traditionally, and as motivated by applications, the interesting regime of parameters is when A is of full row rank (in particular $n \leq m$) [2].

The dictionary learning problem is motivated by the intuition that the dictionary A is in some sense the "right" spanning set for representing vectors y_i since it allows sparse representation. In some domains this correct basis is known thanks to a deep understanding of the domain in question: for example the Fourier basis for audio processing, or Haar wavelets for images. Here we want to infer analogous "nice" representations of the data from the data itself. As it turns out, even in situations such as audio and image processing in which traditional transforms are useful, replacing them with dictionaries learned directly from data turned out to improve quality of the solution (see for example [12], which applied a dictionary learning algorithm for image denoising).

This problem has found a tremendous number of applications in various areas, such as image and video processing (e.g. [21, 10, 12]; see [19] for more references), image classification [23, 20] as well as neurobiology [16]. Given its huge practical importance, a number of effective heuristics for dictionary learning were proposed [3, 18] – those are based on iterative methods for solving the (non-convex) optimization problem of minimizing the sparsity of X' subject to Y being close to $A'X'$. Some of these algorithms work well in practice but without provable guarantees.

1.1 Prior work

Until recently there was little theoretical understanding of the dictionary learning problem. Spielman, Wang and Wright in [24] proposed the first algorithm that provably solves this problem in some regime of parameters. More concretely, they assumed no presence of noise (i.e. $E = 0$), and that A is a basis (that is $n = m$), potentially adversarially chosen. The vectors x_i are sampled independently at random from some distribution – specifically, each entry $x_{i,j}$ is nonzero with probability $1 - \theta$, and once it is nonzero, it is a symmetric subgaussian random variable (i.e. with tails decaying at least as fast as a gaussian), independent from every other entry. Henceforth we say that a matrix $X \in \mathbb{R}^{n \times p}$ follows the *Bernoulli-subgaussian model* with parameter θ, if the entries $X_{i,j}$ are i.i.d. with $X_{i,j} = \chi_{i,j} g_{i,j}$, where $\chi_{i,j} \in \{0, 1\}$ are Bernoulli random variables with $\mathbb{E}\, \chi_{i,j} = \theta$, and $g_{i,j}$ are symmetric subgaussian random variables. We also say that X follows the Bernoulli-Rademacher model if $g_{i,j}$ in the above definition are independent Rademachers (i.e. uniform ± 1).

Under the Bernoulli-subgaussian model for X, [24] proved that once the number of samples p is $\Omega(n \log n)$ and the sparsity $s = \theta n$ (i.e. expected number of nonzero entries in each column of X) is at least constant and at most $\mathcal{O}(n)$, the matrix Y with high probability has a unique decomposition as a product $Y = AX$, up to permuting and rescaling rows of X and columns of A. Moreover, the number of samples $p = \Omega(n \log n)$ was proven to be optimal in the constant sparsity regime $s = \Theta(1)$. In particular, it is possible in principle to find such a decomposition information-theoretically, but unfortunately not necessarily with an efficient algorithm.

[1] As written, their work has certain errors which we discuss later in detail. Nevertheless, using some of our approaches we believe it should be possible to salvage their sample complexity bound in the

ref	sample complexity	noise	overcomplete	sparsity	arbitrary dict.
[24]	$\mathcal{O}(n^2 \log^2 n)$	No	No	$\mathcal{O}(\sqrt{n})$	Yes
[2]	$\mathcal{O}(m^2)$	No	Yes	$\mathcal{O}(n^{1/4})$	No
[5]	$\mathcal{O}(m^2 s^{-2} + s^2 m)$	Yes	Yes	$\mathcal{O}(\min(m^{2/5}, \frac{\sqrt{n}}{\log n}))$	No
[5]	$\mathcal{O}(\mathrm{poly}(m))$	Yes	Yes	$\mathcal{O}(n^{1/2-\epsilon})$	No
[4]*	$\mathcal{O}(\mathrm{poly}(m))$	No	Yes	$\mathcal{O}(n/\mathrm{polylog}(n))$	No
[7]	$\mathcal{O}(\mathrm{poly}(m))$	Yes	Yes	$\mathcal{O}(n^{1-\epsilon})$	Yes
[7]*	$\mathcal{O}(\mathrm{poly}(m))$	Yes	Yes	$\mathcal{O}(n)$	Yes
[25]	$\mathcal{O}(\mathrm{poly}(m, \kappa(A)))$	No	No	$\mathcal{O}(n)$	Yes
[27]	$\mathcal{O}(\mathrm{poly}(n))$	Yes	No	$\mathcal{O}(n)$	Yes
[17][1]	$\mathcal{O}(n \log^4 n)$	No	No	$\mathcal{O}(\sqrt{n})$	Yes
This work	$\mathcal{O}(n \log n)$	No	No	$\mathcal{O}(\sqrt{n})$	Yes

Figure 1 Comparison of algorithms with proven guarantees for dictionary learning. Last column indicates whether the dictionary can be arbitrary, or if additional structure is assumed in order to guarantee recovery. Algorithms marked with star require quasi-polynomial running time. $\kappa(A)$ denotes condition number.

In addition to the above, they proposed an efficient algorithm **ER-SpUD** (*Efficient Recovery of Sparsely Used Dictionaries*) to find this unique decomposition, in a more restricted regime of parameters. Namely, they proposed an algorithm and proved that it finds correctly the unique decomposition $Y = AX$, with high probability over X, as long as the sparsity s is at least constant and at most $\mathcal{O}(\sqrt{n})$, and the number of samples p is at least $\Omega(n^2 \log^2 n)$. The low sparsity constraint was inherent to their solution: according to the proof in the same paper, if $s = \Omega(\sqrt{n \log n})$ the algorithm with high probability fails to find the correct decomposition. They conjectured however, that with the number of samples p as small as $\mathcal{O}(n \log n)$, **ER-SpUD** should return the correct decomposition with high probability, matching the sample lower bound for when $s = O(1)$.

Since then, much more theoretical work has been dedicated to the dictionary learning problem; see Figure 1. In the work of Agarwal et al. [2], and independently Arora et al. [5], an algorithm was proposed that works for overcomplete dictionaries A (i.e. when $m > n$), under additional structural assumptions on A – namely that A is *incoherent*, i.e. the projection of any standard basis vector onto the column space of A has small norm. The algorithm presented in [2] requires $p = \tilde{\mathcal{O}}(m^2)$ samples, where $\tilde{\mathcal{O}}(f) = \mathcal{O}(f \cdot \log^{O(1)}(f))$. A more detailed analysis of the dependence between sparsity and number of samples was provided in the work [5] for their algorithm – for $s = \mathcal{O}(\min(\frac{\sqrt{n}}{\log n}, m^{2/5}))$, they require $\tilde{\Omega}(m^2 s^{-2} + m s^2)$ samples; if s is larger than $m^{2/5}$, but smaller than $\mathcal{O}(\min(m^{1/2-\varepsilon}, \frac{\sqrt{n}}{\log n}))$ the algorithm requires $\mathcal{O}(m^C)$ samples, where C is a large constant depending on ε. In the lowest sparsity regime, i.e. $s = \mathcal{O}(\mathrm{polylog}(n))$, the sample complexity stated in their analysis simplifies to $\tilde{\Omega}(m^2)$. For comparison, in the most favorable sparsity regime $s = \Theta(m^{1/4})$, the number of samples necessary for correct recovery is $\Omega(m^{3/2})$. The work [5] also proves correct recovery by this algorithm in the presence of noise. Later Arora et al. [4] gave a quasipolynomial time algorithm working for sparsity up to $\mathcal{O}(n/\mathrm{polylog}(n))$, but under much stronger assumptions

Bernoulli-gaussian model for X, but not in the more general Bernoulli-subgaussian model (since in particular, $p \gtrsim n^{1.99}$ samples are required for that algorithm even to succeed with polynomially small *success* probability; see the full version for details.

on the structure of A. Those assumptions include in particular, that the dictionary A itself is assumed to be sparse, which is violated in many natural examples, e.g. the discrete Fourier basis. They prove that their algorithm correctly recovers the hidden dictionary given access to $p = \mathcal{O}(m^C)$ samples, for some unspecified constant C.

Barak et al. [7] proposed an algorithm fitting in the Sum-of-Squares framework, which works in polynomial time for sparsity $\mathcal{O}(n^{1-\epsilon})$ for any constant $\epsilon > 0$ and in quasipolynomial time for sparsity as large as $\mathcal{O}(n)$, again given access to $\mathcal{O}(m^C)$ samples for some unspecified constant C. Moreover, this algorithm works under the presence of noise and a more general model of X. In particular, coordinates within a single column are not required to be fully independent. Recently, Sun et al. [25] proposed a polynomial time algorithm for the case when $n = m$ and sparsity is as large as $\mathcal{O}(n)$. Their result works in a similar model as in [24], without any additional assumptions on the matrix A, and with matrix X having independent entries that are product of Bernoulli and gaussian random variables (as opposed to the weaker subgaussian assumption in [24]). The sample complexity depends polynomially on n and the condition number of the dictionary matrix A. In particular, in the low sparsity regime ($s = \Theta(\text{polylog}(n))$), this sample complexity is as large as $\tilde{\Omega}(n^9)$ even if the matrix A is well conditioned.

Work on Independent Component Analysis (ICA) [13, 22, 8, 6, 14, 27] is also relevant to the dictionary learning problem. In this problem, again one is given $Y = AX + E$ for square A, with the assumption that the entries of X are i.i.d. (and X need not necessarily be sparse). The works in ICA then say that A, X can be efficiently recovered using few samples, but where the sample complexity depends on the distribution of entries of X. For example in the case of Bernoulli-Rademacher entries with $\theta = 1/n$ (constant sparsity per column of X), these works require large polynomial sample complexity. For example, [27, Theorem 1] implies a sufficient sample complexity in this setting of $p \gg n^{12}$.

From Figure 1, one can see that the "holy grail" of dictionary learning is to achieve the following features simultaneously: (1) low sample complexity, i.e. nearly-linear in the dimension n and number of atoms m, (2) the ability to handle noise (the more noise handled the better), (3) handling overcomplete dictionaries (i.e. dictionaries for which m may be larger than n), (4) handling a larger range of sparsity, with $s = O(n)$ being the best, (5) making no assumptions on the dictionary A, (6) a fast algorithm to actually learn the dictionary from samples, and (7) making few assumptions on the matrix X.

Most of the aforementioned results focus on weakening the sparsity constraint under which it is possible to perform learning, or handling overcomplete dictionaries or noise. These all, however, come at an expense: the number of samples necessary for those algorithms to provably work is quite large, often of order n^C for large constant C. Some of the algorithms also make strong assumptions on A, and/or have quasi-polynomial running time.

Recently, Luh and Vu in [17] made significant progress toward showing that the **ER-SpUD** algorithm proposed in [24] actually solves the dictionary learning problem already with $p = \mathcal{O}(n \log^4 n)$ samples. They claimed to prove that this p in fact suffices for dictionary learning. In fact however, several probabilistic events were analyzed in [24], and if they all occurred then **ER-SpUD** performed correct recovery. The work [17] analyzed arguably the most complex of these events more efficiently, showing a certain crucial inequality held with good probability when $p \gtrsim n \log^4 n$ ($x \gtrsim y$ means that $x > Ky$ for some universal constant $K > 0$). Unfortunately there is a gap: [24] required this inequality to hold for exponentially many settings of variables, and thus one wants the inequality to hold for any fixed instantiation with very high probability to then union bound, and [17] does not provide such a probabilistic analysis. More seriously, there are other events defined in [24]

which *require* $p \gtrsim n^2$ to hold whp in the Bernoulli-subgaussian model (except in the case the subgaussians are actual gaussians), and [17] did not discuss these events at all. In fact, in the full version we prove that in the Bernoulli-Rademacher model the **ER-SpUD** algorithm of [24] actually *requires* $p \gtrsim n^{1.99}$ to succeed with probability even polynomially small in n, contradicting the main result of [17] which claimed $1 - o(1)$ successful learning for p nearly linear in n.

Our contribution: We very slightly modify the algorithm **ER-SpUD** to obtain another polynomial-time dictionary learning algorithm "**ER-SpUD(DCv2)**" for the noiseless case with $m = n$, which circumvents our $p \gtrsim n^{1.99}$ lower bound for **ER-SpUD** in the Bernoulli-subgaussian model. We then show that **ER-SpUD(DCv2)** provides correct dictionary learning with probability $1 - \delta$ with sparsity $s = \mathcal{O}(\sqrt{n})$ as long as $p \gtrsim n \log(n/\delta)$. In particular our result shows that a slight modification of **ER-SpUD** provides correct dictionary learning for complete dictionaries with no noise, which provably works with high probability using $p \gtrsim n \log n$ samples. This resolves the main open problem of [24].

Furthermore, the work of [17] observed that the method of their proof is connected to generic chaining, but that after a certain point the methods "become different in all aspects" [17, Section G]. They also advertised and proved a new "refined version of Bernstein's concentration inequality for a sum of independent variables". Unlike their work, our analysis has the benefit of using standard off-the-shelf concentration and chaining results, thus making the proof simpler and more easily accessible since it is less ad-hoc.

1.2 Approach overview

In Figure 2 we give the algorithm **ER-SpUD(DCv2)** analyzed in this work, a slight modification of **ER-SpUD(DC)** from [24]. The only difference between DCv2 and the original DC variant in [24] is that we try all $\binom{p}{2}$ pairings of columns, whereas DC tried a random pairing of the p columns into $p/2$ pairs. As we show in the full version, one of the several conditions in [24] necessary for their proof of successful recovery of (A, X) from Y actually requires $p = \Omega(n^2)$ if using the DC variant, and hence our switch to DCv2 allows p to be reduced to $\mathcal{O}(n \log n)$.

Henceforth when we refer to **ER-SpUD**, we are referring to **ER-SpUD(DCv2)** unless we state otherwise.

The main insight in the recovery analysis of [24] is that the last line of the **ER-SpUD** pseudocode in Figure 2 can be rewritten (only in the analysis, since A, X are unknown) as $\min_w \|w^T AX\|_1$ subject to $(A(Xe_{j_1} + Xe_{j_2}))^T w = 1$. Then writing $z = A^T w$, this linear program (LP) is equivalent to the secondary LP $\min_z \|z^T X\|_1$ subject to $b_j^T z = 1$, since we could recover $w = (A^T)^{-1} z$ since A is invertible. Here b_j denotes $Xe_{j_1} + Xe_{j_2}$. The ideal case then is that the only optimal solution to the second LP will be a vector z_* that is 1-sparse. In this case, the solution to the LP that we *actually* solve is equal to $w_* = (A^T)^{-1} z_* = (z_*^T A^{-1})^T$ and thus a scaled row of A^{-1}, implying $w_*^T Y$ is a scaled row of X. Thus, if z_* is 1-sparse in the second LP, then the solution to the first LP allows us to recover a scaled row of X.

The work [24] then outlines certain conditions for X that, if they hold, guarantee correct recovery of (A, X). We now state these deterministic conditions, as per [24], which imply correct recovery of (A, X) via **ER-SpUD** when they all simultaneously hold.

(P0) Every row of X has positive support size at most $(10/9)\theta p$. Furthermore, every linear combination of rows of X in which at least two of the coefficients in the linear combination are non-zero has support size at least $(11/9)\theta p$.

ER-SpUD(DCv2): `Exact Recovery of Sparsely-Used Dictionaries using the sum of two columns of` Y `as constraint vectors.`

1. For all pairs $j_1 < j_2 \in \{1, \dots, p\}$
   Let $\boldsymbol{r}_j = \boldsymbol{Y}\boldsymbol{e}_{j_1} + \boldsymbol{Y}\boldsymbol{e}_{j_2}$
   Solve $\min_{\boldsymbol{w}} \|\boldsymbol{w}^T\boldsymbol{Y}\|_1$ subject to $\boldsymbol{r}_j^T\boldsymbol{w} = 1$, and set $\boldsymbol{s}_j = \boldsymbol{w}^T\boldsymbol{Y}$.
   $j \leftarrow j + 1$

Greedy: `A Greedy Algorithm to Reconstruct` $\boldsymbol{X}$ `and` $\boldsymbol{A}$.

1. **REQUIRE:** $\mathcal{S} = \{\boldsymbol{s}_1, \dots, \boldsymbol{s}_T\} \subset \mathbb{R}^p$.
2. For $i = 1 \dots n$
   REPEAT
   $l \leftarrow \arg\min_{\boldsymbol{s}_l \in \mathcal{S}} \|\boldsymbol{s}_l\|_0$, breaking ties arbitrarily
   $\boldsymbol{x}_i = \boldsymbol{s}_l$
   $\mathcal{S} = \mathcal{S} \backslash \{\boldsymbol{s}_l\}$
   UNTIL `rank([`$\boldsymbol{x}_1, \dots, \boldsymbol{x}_i$`])=` i
3. Set $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]^T$, and $\boldsymbol{A} = \boldsymbol{Y}\boldsymbol{Y}^T(\boldsymbol{X}\boldsymbol{Y}^T)^{-1}$.

Figure 2 ER-SpUD recovery algorithm.

(P1) For every b satisfying $\|b\|_0 \le 1/(8\theta)$, any solution z_* to the optimization problem

$$\min \|z^T X\|_1 \text{ subject to } b^T z = 1 \tag{1}$$

has support$(z_*) \subseteq$ support(b).

(P2) Let q be $\frac{1}{8\theta}$. For every $J \in \binom{[n]}{q}$ and every $b \in \mathbb{R}^n$ satisfying $|b|_{(2)}/|b|_{(1)} \le 1/2$, the solution to the restricted problem

$$\|z^T X_{J,*}\|_1 \text{ subject to } b^T z = 1 \tag{2}$$

is unique, 1-sparse, and is supported on the index of the largest entry of b. Here $|b|$ is the vector whose ith entry is $|b_i|$, and $|b|_{(j)}$ is the jth largest entry of $|b|$. Also, $X_{J,*}$ denotes the submatrix of X with rows in J.

(P3) For every $i \in [n]$ there exist a pair of columns Xe_{j_1} and Xe_{j_2} in X such that for $b = Xe_{j_1} + Xe_{j_2}$ with support J, we have that $0 < |J| \le 1/(8\theta)$, $|b|_{(2)}/|b|_{(1)} \le 1/2$, and the unique largest entry of $|b|$ has index i.

The main result of [24] is then obtained by proving the following theorem, and then by showing that **(P0)**–**(P3)** all hold whp for $p \gtrsim n^2 \log^2 n$.

▶ **Theorem 1** ([24]). *Suppose conditions* **(P0)**–**(P3)** *all hold. Then* ***ER-SpUD*** *and* ***Greedy*** *from Figure 2 recover* (A', X') *such that* $X' = \Pi D X$ *and* $A = AD^{-1}\Pi^{-1}$ *for some diagonal scaling matrix* D *and permutation matrix* Π. *That is, the recovered* (A', X') *are correct up to scaling and permuting rows (resp. columns) of* X *(resp.* A*).*

It was implicit in [24], and made explicit in [17], that to analyze the probability **(P1)** holding as a function of p, it suffices to prove some upper bound on some stochastic process. Namely, [17] proves that for Π a Bernoulli-subgaussian matrix with p rows, for $p = \Omega(n \log^4 n)$

$$\mathbb{P}\left(\sup_{\|v\|_1 = 1} |\|\Pi v\|_1 - \mathbb{E}\,\|\Pi v\|_1| < c_0 \mu_{min} \right) > 1 - o(1) \tag{3}$$

for some constant $c_0 < 1$, and $\mu_{min} := \inf_{\|v\|_1=1} \mathbb{E}\,\|X^T v\|_1$. Both [24, 17] though required the stochastic process of Eq. (3) to be bounded for roughly $\binom{n}{1/(8\theta)}$ choices of Π, formed by taking various submatrices of X^T. The naive approach is to then argue that the inequality holds with failure probability $\ll 1/\binom{n}{1/(8\theta)}$ for a fixed Π so then union bound over all such submatrices. Unfortunately the failure probability in [17] was not made explicit and was only given as $o(1)$, so it does not clearly allow for this union bound.

We show that, first of all, **(P1)** can be relaxed to some **(P1')** such that it suffices to only show Eq. (3) holds for polynomially many submatrices of X; showing **(P1')** suffices requires only a very minor change in the previous analysis of [24]. Next, more importantly, we show that $p \gtrsim n \log(n/\delta)$ suffices for Eq. (3) to hold with probability $1 - \delta$. This is one of our main technical contributions, and is established using a generic chaining argument [26]. It is worth pointing out that simpler chaining inequalities, such as Dudley's inequality, would yield suboptimal results in our setting by logarithmic factors.

Next, we also show that **(P2)** can be weakened to some other event **(P2')** that holds whp as long as $p \gtrsim \theta^{-1} \log(n/\delta)$ – this requires only a minor change in the analysis of [24].

Finally, in Lemma 4 we show that event **(P3)** holds whp for $p \gtrsim n \log(n/\delta)$. This is the part where the modification of the algorithm was necessary, so that pairs of columns Xe_{j_1} and Xe_{j_2} mentioned in this condition refers to all $\binom{p}{2}$ pairs of columns, as opposed to a fixed pairing (with $\lfloor \frac{p}{2} \rfloor$ pairs). Note that this condition actually fails to hold for the unmodified version of the algorithm with $p \ll n^2$, for example when the matrix X is drawn from the Bernoulli-Rademacher model, which is the main reason the unmodified algorithm fails to perform recovery (see the full version).

1.3 Recent and independent work

In a recent and independent work, Adamczak showed a main result similar to ours [1]. In particular, he showed that by making the same modification to **ER-SpUD** that we have made (**ER-SpUD(DCv2)**), $p \gtrsim n \log n$ suffices for successful dictionary learning with probability $1 - 1/p$. Unlike our analysis which is based on Bernstein's inequality and generic chaining, the proof in [1] combines Bernstein's inequality with Talagrand's contraction principle, which leads to an overall simpler proof than ours. The main differences in the results themselves are that attention in [1] was not given to dependence of p on the failure probability δ, and the analysis in our full version that **ER-SpUD(DC)** fails for $p \ll n^2$ also does not appear there, so that our stated results are slightly stronger in these regards.

2 Sufficient conditions for successful recovery

We first define **(P1')**, **(P2')** as follows.

(P1') For every b that can be expressed as the sum of two columns of X, $|S| < p/4$ and

$$\forall v \in \mathbb{R}^{|\bar{J}|},\ \|v^T X_{\bar{J},*}\|_1 - 2\|v^T X_{\bar{J},S}\|_1 > Cp\sqrt{\frac{\theta}{|\bar{J}|}}\|v\|_1 \tag{4}$$

where $C > 0$ is some fixed constant, $J = \mathrm{support}(b)$, $\bar{J} = [n] \backslash J$, and $S \subseteq [p]$ is the set of columns of X with support intersecting J.

(P2') Let q be $\frac{1}{8\theta}$. For every b equaling the sum of two columns of X and with $J \subset [n]$ its support, let $b' \in \mathbb{R}^{|J|}$ be the projection of b onto its support. If $0 < |J| \leq q = 1/(8\theta)$ and $|b|_{(2)}/|b|_{(1)} \leq 1/2$, then the solution to the restricted problem

$$\|z^T X_{J,*}\|_1 \text{ subject to } (b')^T z = 1 \tag{5}$$

is unique, 1-sparse, and is supported on the index of the largest entry of b'. Here $|b'|$ is the vector whose ith entry is $|b'_i|$, and $|b'|_{(j)}$ is the jth largest entry of $|b'|$. Also, $X_{J,*}$ denotes the submatrix of X with rows in J.

In the full version, we show that it suffices that **(P1')** and **(P2')** hold instead of **(P1)** and **(P2)** to guarantee correctness of **ER-SpUD(DCv2)**. In particular, we show the following lemma.

▶ **Lemma 2.** *Suppose conditions* **(P0)**, **(P1')**, **(P2')**, *and* **(P3)** *all hold. Then* ***ER-SpUD*** *and* ***Greedy*** *from Figure 2 recover* (A', X') *such that* $X' = \Pi DX$ *and* $A = AD^{-1}\Pi^{-1}$ *for some diagonal scaling matrix* D *and permutation matrix* Π. *That is, the recovered* (A', X') *are correct up to scaling and permuting rows (resp. columns) of* X *(resp.* A*).*

In the full version, we then show **(P0)**, **(P1')**, **(P2')**, and **(P3)** all simultaneously hold with probability $1 - \delta$ as long as $p \gtrsim n \log(n/\delta)$ and $1/n \lesssim \theta \lesssim 1/\sqrt{n}$, which when combined with Lemma 2 implies that **ER-SpUD** has the desired correctness guarantee under this same regime for p, θ.

▶ **Theorem 3.** *For* $p \gtrsim n \log(n/\delta)$ *and* $1/n \lesssim \theta \lesssim 1/\sqrt{n}$,

$$\mathbb{P}(\neg(\mathbf{P0}) \vee \neg(\mathbf{P1'}) \vee \neg(\mathbf{P2'}) \vee \neg(\mathbf{P3})) < \delta \tag{6}$$

▶ Remark. Here we sketch just how **(P1')** is analyzed in Theorem 3, similarly to the discussion in [24, 17]. This reveals why our chaining result, Theorem 12, is relevant.

For **(P1')**, the analysis is almost identical to the proofs of [24, Lemma 11] and [17, Lemma V.2] regarding **(P1)**. We repeat the slightly modified argument here for **(P1')**. Let b be a particular sum of two columns of X. We will show that the condition of **(P1')** fails to hold for b with probability at most δ/p^2, which implies $\mathbb{P}(\neg\textbf{(P1')}) \le \delta$ by a union bound over all $\binom{p}{2}$ such b. Let J, S be as in the definition of **(P1')** above. Define the event $\mathcal{E}_S$ as the event that $|S| < p/4$. Since $\theta n \le c\sqrt{n}$ for some small $c > 0$, if $b = X_{*,j_1} + X_{*,j_2}$, it follows that any column index $j \notin \{j_1, j_2\}$ has support intersecting J with probability at most $1/10$ (by making c sufficiently small). Thus $\mathbb{E}\,|S| < p/10$, implying $\mathbb{P}(\neg\mathcal{E}_S) = \mathbb{P}(|S| \ge p/4)$ is at most $\exp(-\Omega(p)) \le \delta/p^2$ by the Chernoff bound and fact that $p \gtrsim \log(p^2/\delta)$.

The definition of $\mathcal{E}_N$ is the following event:

$$\forall v \in \mathbb{R}^{|\bar{J}|},\ \|v^T X_{\bar{J},*}\|_1 - 2\|v^T X_{\bar{J},S}\|_1 > Cp\sqrt{\frac{\theta}{|\bar{J}|}}\|v\|_1 \tag{7}$$

for some constant C, where $\bar{J}$ denotes $[n]\backslash J$. Note though that $X_{\bar{J},*}$ is itself a matrix of i.i.d. Bernoulli-subgaussian entries (except for the two columns j_1, j_2, which are both zero). Thus setting $\Pi = X^T_{\bar{J},*}$ and applying Theorem 12 with our choice of p, with probability at least $1 - \delta/p^2$, for all $v \in B_1$,

$$\|v^T X_{\bar{J},*}\|_1 \ge \frac{7}{8}\,\mathbb{E}\,\|v^T X_{\bar{J},*}\|_1 = \frac{7p}{8}\,\mathbb{E}\,|v^T (X_{\bar{J},*})_{*,1}| \stackrel{\text{def}}{=} \frac{7p}{8}\alpha(v), \tag{8}$$

where $(X_{\bar{J},*})_{*,1}$ clumsily denotes the first column of the matrix $X_{\bar{J},*}$. The last inequality follows from [24, Lemma 16]. Also, conditioned on $\mathcal{E}_S$, $|S| < p/4$. Let X' be the matrix $X_{\bar{J},S}$ padded with $p/4 - |S|$ additional columns, each independent of but identically distributed to the columns of X. Then, even conditioned on $\mathcal{E}_S$, X' is a $|\bar{J}| \times p/4$ matrix of i.i.d. Bernoulli-gaussian entries (except for two columns which are both identically zero, corresponding to j_1, j_2). Thus applying Theorem 12 to $\Pi = (X')^T$, with probability at least $1 - \delta/p^2$,

$$\forall v \in B_1,\ \|v^T X'\|_1 \le \frac{3}{2}\,\mathbb{E}\,\|v^T X'\|_1 = \frac{3p}{8}\,\mathbb{E}\,|v^T X'_{*,1}| = \frac{3p}{8}\alpha(v). \tag{9}$$

Then by combining (8), (9) and scaling by $\|v\|_1$, we see that the left hand side of (7) is at least $\frac{p}{8}\alpha(v) \gtrsim p\sqrt{\frac{\theta}{|J|}}\|v\|_1$, with the inequality following from [24, Lemma 16].

We are now going to sketch how to show that with probability $1-\delta$ condition **(P3)** holds. The full proof is in the full version.

Consider the special case that $X_{ij} = b_{ij}g_{ij}$ with Bernoulli random variable b_{ij} and independent *continuous* subgaussian random variable g_{ij}. In such a case there would exist some fixed threshold t_0, such that $\mathbb{P}(|X_{ij}| > t_0) = \frac{1}{n}$ – it would mean that a constant fraction of columns would have *unique* entry larger than this threshold. For a single index $i \in [n]$ we would expect that at least $C\frac{p}{n} > \log\frac{n}{\delta}$ columns have a unique entry larger than t_0 and such that this entry has index i. Let us focus on this set of columns. If supports of any two such columns had common intersection exactly equal to $\{i\}$ – and if the sign on this i-th coordinate were matching, then in fact sum of those two columns would exhibit a factor two gap between the largest and the second largest entry, with largest entry being on the i-th position – indeed, entry on position i would have magnitude larger than $2t_0$, whereas all other entries are at most t_0 in absolute value. We can expect to find such a pair with probability $1-\frac{\delta}{n}$, as all columns are expected to be $\mathcal{O}(\sqrt{n})$ sparse – therefore for a fixed pair containing $\{i\}$, their supports would intersect on exactly $\{i\}$ with constant probability. We then prove that there exist such a pair with probability at least $\frac{\delta}{n}$ for every fixed i, and hence by union bound property **(P3)** holds with probability δ.

In the actual proof we do not assume that g_{ij} is continuous, and hence a threshold t_0 for which $\mathbb{P}(|X_{ij}| > t_0) = \frac{1}{n}$ might not exist, and the proof is slightly more complicated, but it follows the same general intuition. We prove the following in the full version.

▶ **Lemma 4.** *Let $X \in \mathbb{R}^{n\times p}$ be a Bernoulli-Subgaussian matrix with $\theta = \mathcal{O}(\frac{1}{\sqrt{n}})$. If $p = \Omega(n\log\frac{n}{\delta})$, then with probability at least $1-\delta$ condition* **(P3)** *holds.*

3 Chaining background

We now provide some preliminary definitions and results we will need to prove Theorem 12. As per Lemma 2 and the proof of Theorem 3, Theorem 12 fits in to show that **ER-SpUD** achieves correct recovery with probability $1-\delta$ for $p \gtrsim n\log(n/\delta)$ and $1/n \lesssim \theta \lesssim 1/\sqrt{n}$.

In this subsection we provide relevant definitions for a technique called *generic chaining*, as well as statements of some of the results in the area. Those tools have been designed to provide answers about the supremum of the fluctuations from the mean for a large collection of random variables, when the reasonable bounds for covariances in terms of the geometry of the set of indices are at hand.

▶ **Definition 5** (Admissible sequence)**.** For an arbitrary set T, we say that a sequence of its subsets $(T_k)_{k=0}^{\infty}$ is admissible if for every number k it is true that $T_k \subset T_{k+1}$ and $|T_k| \le 2^{2^k}$ for $k \ge 1$ and $|T_0| = 1$.

▶ **Definition 6** (Gamma functionals)**.** For a metric space (T,d) we define

$$\gamma_\alpha(T,d) := \inf_{(T_k)} \sup_{x\in T} \sum_{k=0}^{\infty} 2^{k/\alpha} d(x,T_k) \tag{10}$$

where the infimum is taken over all admissible sequences T_k. In the above formula we define as usual $d(x,T_k) := \inf_{t\in T_k} d(x,t)$.

▶ **Fact 7.** *If d and d' are two metrics such that for $d(t_1,t_2) = Cd'(t_1,t_2)$ for every pair of points t_1,t_2, then $\gamma_\alpha(T,d) = C\gamma_\alpha(T,d')$*

▶ **Theorem 8** (Generic chaining [26], Theorem 2.2.23)**.** *Let T be an arbitrary set of indices, and $d_1, d_2 : T \times T \to \mathbb{R}_{\geq 0}$ two metrics on T. Suppose that with any point $t \in T$ we have associated random variable X_t, with $\mathbb{E}\, X_t = 0$. Suppose moreover, that for any two points $u, w \in T$ we have a tail bound: $\mathbb{P}\left(|X_u - X_v| > \lambda\right) \lesssim \exp\left(-\frac{\lambda^2}{d_1(u,v)^2}\right) + \exp\left(-\frac{\lambda}{d_2(u,v)}\right)$. Then $\mathbb{E} \sup_{u\in T} |X_u| \lesssim \gamma_2(T, d_1) + \gamma_1(T, d_2)$.*

▶ **Theorem 9** (Dirksen, [11])**.** *Let T be an arbitrary set of indices and $d_1, d_2 : T \times T \to \mathbb{R}_{\geq 0}$ two metrics on T. Suppose that with any point $t \in T$ we have associated random variable X_t, such that $\mathbb{E}\, X_t = 0$. Suppose moreover that for any two points $u, w \in T$, we have a tail bound*

$$\mathbb{P}(|X_u - X_v| > \lambda) \lesssim \exp\left(-\frac{\lambda^2}{d_1(u,v)^2}\right) + \exp\left(-\frac{\lambda}{d_2(u,v)}\right)$$

Then for there exists an universal constant C, such that for any $u > 0$

$$\mathbb{P}\left(\sup_{u\in T} |X_u| > C(\gamma_2(T, d_1) + \gamma_1(T, d_2) + \sqrt{u}\Delta(T, d_1) + u\Delta(T, d_2))\right) < e^{-u}$$

where $\Delta(T, d) := \sup_{u,v\in T} d(u, v)$.

▶ **Theorem 10** (Majorizing measures [26], Theorem 2.4.1)**.** *Let $T \subset \mathbb{R}^n$, and assume that $g = (g_1, \dots g_n)$ is a vector of i.i.d. standard normal random variables. Then $\mathbb{E} \sup_{t\in T} \langle g, t\rangle \simeq \gamma_2(T, d_2)$, where d_p is the metric induced by the ℓ_p norm.*

▶ **Theorem 11** ([26], Theorem 10.2.8)**.** *Let $T \subset \mathbb{R}^n$, and assume that $x = (x_1, \dots, x_n)$ is a vector of i.i.d. standard exponential random variables. Then $\mathbb{E} \sup_{t\in T} \langle t, x\rangle \simeq \gamma_2(T, d_2) + \gamma_1(T, d_\infty)$*

4 Proof of the stochastic process bound

In this section we will prove the following theorem, which provides a stronger form of Eq. (3).

▶ **Theorem 12.** *Let $\Pi \in \mathbb{R}^{m\times n}$ be a random matrix with i.i.d. random entries $\pi_{ij} = \chi_{ij} g_{ij}$, where $\chi_{ij} \in \{0, 1\}$ is a Bernoulli random variable with $\mathbb{E}\, \chi_{ij} = \theta$, and g_{ij} symmetric subgaussian random variable. Moreover, assume that $\frac{1}{n} \leq \theta$. When $m = \Omega(\varepsilon^{-2} n \log \frac{n}{\delta})$,*

$$\mathop{\mathbb{P}}_{\Pi}\left(\sup_{v\in B_1} |\|\Pi v\|_1 - \mathbb{E}\, \|\Pi v\|_1| > \varepsilon \cdot \mathbb{E}\, \|\Pi v\|_1\right) < \delta \tag{11}$$

We now prove the theorem. Define $B_1 := \{t \in \mathbb{R}^n : \|t\|_1 \leq 1\}$. For each $v \in B_1$, consider $\tilde{X}_v := \|\Pi v\|_1 - \mathbb{E}\, \|\Pi v\|_1$. We wish to prove that with high probability over Π we have $\sup_{v\in B_1} |\tilde{X}_v| \leq \varepsilon\mu_{min}$, where $\mu_{min} := m\sqrt{\frac{\theta}{n}}$ is such that for every $v \in B_1$ we have $\mathbb{E}\, \|\Pi v\|_1 \geq \mu_{min}$ (see [24, Lemma 16] for a proof). Let $\pi_1, \dots \pi_m$ be the rows of matrix Π. With each $v \in B_1$ we associate another random variable $X_v := \sum_{i=1}^m \sigma_i |\langle \pi_i, v\rangle|$, with the σ_i being independent Rademachers.

▶ **Lemma 13.** *For every integer p we have*

$$\| \sup_{v\in B_1} |\tilde{X}_v| \|_p \lesssim \| \sup_{v\in B_1} |X_u| \|_p . \tag{12}$$

Proof. Without loss of generality consider even integer p, so that $|X_u|^p = X_u^p$. Let $\tilde{\Pi}$ be a random matrix, independent and identically distributed as Π. By Jensen's inequality we

have

$$\begin{aligned}\|\sup_{v\in B_1}|\tilde{X}_v|\|_p &= \left\|\sup_{v\in B_1}\|\Pi v\|_1 - \mathop{\mathbb{E}}_{\tilde{\Pi}}\|\tilde{\Pi}v\|_1\right\|_p \\ &\le \left\|\sup_{v\in B_1}\|\Pi v\|_1 - \|\tilde{\Pi}v\|_1\right\|_p \\ &= \left\|\sup_{v\in B_1}\sum_{i=1}^m |\langle\pi_i,v\rangle| - |\langle\tilde{\pi}_i,v\rangle|\right\|_p\end{aligned}$$

Now each summand $|\langle\pi_i,v\rangle| - |\langle\pi_i,v\rangle|$ is symmetric random variable, and they are independent. We can thus introduce independent random signs σ_i without altering the distribution:

$$\begin{aligned}\|\sup_{v\in B_1}|\tilde{X}_v|\|_p &\lesssim \left\|\sup_{v\in B_1}\sum_{i=1}^m \sigma_i(|\langle\pi_i,v\rangle| - |\langle\tilde{\pi}_i,v\rangle|)\right\|_p \\ &\le \left\|\sup_{v\in B_1}\sum_{i=1}^m \sigma_i|\langle\pi_i,v\rangle|\right\|_p + \left\|\sup_{v\in B_1}\sum_{i=1}^p(-\sigma_i)|\langle\tilde{\pi}_i,v\rangle|\right\|_p \\ &= 2\left\|\sup_{v\in B_1}\sum_{i=1}^m \sigma_i|\langle\pi_i,v\rangle|\right\|_p \lesssim \left\|\sup_{v\in B_1}|X_v|\right\|_p\end{aligned}$$

◀

We will first analyze tail behavior of the random variable $\sup_{v\in B_1}|X_v|$, and then use Lemma 13 together with [15, Lemma 4.10] to obtain tail bounds for the random variable of original interest $\sup_{v\in B_1}|\tilde{X}_v|$.

In order to use Theorem 9 to obtain tail bounds for supremum of X_u, we need to bound tails of random variables $X_u - X_v$ for $u, v \in B_1$.

▶ **Lemma 14.** *For every pair of points $u, v \in B_1$, we have*

$$\mathbb{P}(|X_u - X_v| > \lambda) \lesssim \exp\left(-\frac{\lambda^2}{2m\theta\|u-v\|_2^2}\right) + \exp\left(-\frac{\lambda}{\|u-v\|_\infty}\right) \tag{13}$$

Proof. We can write

$$X_u - X_v = \sum_{i=1}^m \sigma_i(|\langle\pi_i,u\rangle| - |\langle\pi_i,v\rangle|) \tag{14}$$

Define $Q_i := \sigma_i(|\langle\pi_i,u\rangle| - |\langle\pi_i,v\rangle|)$. We have $X_u - X_v = \sum_{i=1}^m Q_i$, where all Q_i are symmetric and identically distributed.

Moreover, we have $|Q_i| = ||\langle\pi_i,u\rangle| - |\langle\pi_i,v\rangle|| \le |\langle\pi_i,u-v\rangle|$. Observe that each π_{ij} is $(\sqrt{2\theta},1)$-subgamma. Here we say a random variable Z is (σ,B)-subgamma if $\mathbb{E}\,Z = 0$ and $\psi_Z(\lambda) \le \lambda^2\sigma^2/(2(1-B\lambda))$ for all $|\lambda| < 1/|B|$, where $\psi_Z(\lambda) = \ln \mathbb{E}\,e^{\lambda Z}$. By basic properties of subgamma random variables (see [9, Section 2.4]), we know that $\langle\pi_i, u-v\rangle$ is $(\sqrt{2\theta}\|u-v\|_2, \|u-v\|_\infty)$-subgamma.

Now, as both Q_i and $\langle\pi_i,u-v\rangle$ are symmetric, and $|Q_i| \le |\langle\pi_i,u-v\rangle|$ always, we deduce that each Q_i is also $(\sqrt{2\theta}\|u-v\|_2, \|u-v\|_\infty)$-subgamma.

Finally, $X_u - X_v$, as a sum of independent subgamma random variables is $(\sqrt{2m\theta}\|u-v\|_2^2, \|u-v\|_\infty)$-subgamma. This, together with [9, Section 2.4] implies the tail bound

$$\mathbb{P}\left(\left|\sum_{i=1}^m Q_i\right| > \lambda\right) \lesssim \exp\left(\frac{\lambda^2}{2m\theta\|u-v\|_2^2}\right) + \exp\left(\frac{\lambda}{\|u-v\|_\infty}\right) \tag{15}$$

◀

With this lemma in hand, we can use Theorem 9, to deduce the tail bound for supremum of $|X_v|$.

$$\mathbb{P}\left(\sup_{v\in B_1}|X_u| > M + \sqrt{u}D_1 + uD_2\right) < e^{-u} \tag{16}$$

Where $M := C_1(\gamma_2(B_1, \sqrt{2m\theta}d_2) + \gamma_1(B_1, d_\infty))$, $D_1 := C_2\Delta(B_1, \sqrt{2m\theta}d_2)$, and $D_2 := C_3\Delta(B_1, d_\infty)$, with d_2, d_∞ being metrics on $\mathbb{R}^n$ induced by norms ℓ_2, ℓ_∞ respectively, and C_1, C_2, C_3 are universal constants.

We claim that, we can deduce similar tail bounds for $\sup_{v\in B_1}|\tilde{X}_u|$. Namely

$$\mathbb{P}\left(\sup_{v\in B_1}|\tilde{X}_u| > L(M + \sqrt{u}D_1 + uD_2)\right) < e^{-u} \tag{17}$$

for some universal constant L.

Indeed it is known (see [15, Lemma 4.10]) that tail bounds of the form Eq. (16) imply moment bounds of the form $\|\sup_{v\in B_1}|X_v|\|_p \lesssim M + \sqrt{p}D_1 + pD_2$. By Lemma 13, the same (up to a constant) p-norm bounds are true for $\sup_{v\in B_1}|\tilde{X}_v|$. Finally, [15, Lemma 4.10] also implies similar tail behavior of the random variable $\sup_{v\in B_1}\tilde{X}_v$, as in Eq. (17).

If we set $u := \log\frac{1}{\delta}$ in Eq. (17), we will get an upper bound for $\sup_{v\in B_1}\tilde{X}_v$ which is satisfied with probability at least $1-\delta$. We need to understand the values of M, $\sqrt{u}D_1$ and uD_2, for this setting of u, and we will show how to pick m such that sum of those values is smaller than $\varepsilon\mu_{min}$.

Let us focus now on bounding M. We have $\gamma_2(B_1, \sqrt{2m\theta}d_2) = \sqrt{2m\theta}\gamma_2(B_1, d_2)$. We need an upper bound for $\gamma_2(B_1, d_2)$ and $\gamma_1(B_1, d_\infty)$. We prove the following in the full version, using Theorem 10 and Theorem 11.

▶ **Fact 15.** *$\gamma_2(B_1, d_2) \lesssim \sqrt{\log n}$ and $\gamma_1(B_1, d_\infty) \lesssim \log n$.*

Fact 15 together with previous discussion yield an upper bound $M \lesssim \sqrt{m\theta\log n} + \log n$.

Moreover, as $d_2(u, v) \le d_1(u, v)$ for any $u, v \in \mathbb{R}^n$, where d_1 is the metric induced by the ℓ_1 norm, we can easily upper bound diameter of B_1 in d_2 by diameter of B_1 and d_1 and therefore obtain an upper bound for D_1

$$D_1 = C_1\Delta(B_1, \sqrt{em\theta}d_2) = C_1\sqrt{em\theta}\Delta(B_1, d_2) \le C_1 2\sqrt{em\theta}$$

and similarly $\Delta(B_1, d_\infty) = 2$. Altogether, we have following inequalities: $M \lesssim \sqrt{m\theta\log n} + \log n$, $D_1 \lesssim \sqrt{m\theta}$, and $D_2 \lesssim 1$. Plugging this back to Eq. (17), we have

$$\mathbb{P}\left(sup_{v\in B_1}|\tilde{X}_v| < L_2\left(\sqrt{m\theta(\log n + \log\frac{1}{\delta})} + \log n + \log\frac{1}{\delta}\right)\right) < \delta \tag{18}$$

where again L_2 is some constant.

The following inequalities are equivalent: $L_2\sqrt{m\theta\log\frac{n}{\delta}} \le \frac{1}{2}\varepsilon\mu_{min}$, $L_2\sqrt{m\theta\log\frac{n}{\delta}} \le \frac{1}{2}\varepsilon\sqrt{\frac{\theta}{n}}m$, and $\frac{4L_2^2}{\varepsilon^2}n\log\frac{n}{\delta} \le m$. Similarly, the assumption $\theta \ge \frac{1}{n}$ implies that if $m > \frac{2L_2}{\varepsilon}n\log\frac{n}{\delta}$, then also $L_2\log\frac{n}{\delta} \le \frac{1}{2}\varepsilon\mu_{min}$, so once m is larger than both those values, Eq. (18) implies $\mathbb{P}\left(\sup_{v\in B_1}|\tilde{X}_v| > \varepsilon\mu_{min}\right) < \delta$, as desired.

Acknowledgments. We thank John Wright for pointing out to us the recent independent work [1].

References

1. Radosław Adamczak. A note on the sample complexity of the Er-SpUD algorithm by Spielman, Wang and Wright for exact recovery of sparsely used dictionaries. *CoRR*, abs/1601.02049, 2016.
2. Alekh Agarwal, Animashree Anandkumar, Prateek Jain, Praneeth Netrapalli, and Rashish Tandon. Learning sparsely used overcomplete dictionaries. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, pages 123–137, 2014.
3. M. Aharon, M. Elad, and A. Bruckstein. SVDD: An algorithm for designing overcomplete dictionaries for sparse representation. *Trans. Sig. Proc.*, 54(11):4311–4322, November 2006.
4. Sanjeev Arora, Aditya Bhaskara, Rong Ge, and Tengyu Ma. More algorithms for provable dictionary learning. *CoRR*, abs/1401.0579, 2014.
5. Sanjeev Arora, Rong Ge, and Ankur Moitra. New algorithms for learning incoherent and overcomplete dictionaries. In *Proceedings of The 27th Conference on Learning Theory (COLT)*, pages 779–806, 2014.
6. Sanjeev Arora, Rong Ge, Ankur Moitra, and Sushant Sachdeva. Provable ICA with unknown gaussian noise, and implications for gaussian mixtures and autoencoders. *Algorithmica*, 72(1):215–236, 2015.
7. Boaz Barak, Jonathan A. Kelner, and David Steurer. Dictionary learning and tensor decomposition via the sum-of-squares method. In *Proceedings of the 47th Annual ACM on Symposium on Theory of Computing (STOC)*, pages 143–151, 2015.
8. Mikhail Belkin, Luis Rademacher, and James R. Voss. Blind signal separation in the presence of gaussian noise. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT)*, pages 270–287, 2013.
9. Stephane Boucheron, Gabor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.
10. Ori Bryt and Michael Elad. Compression of facial images using the K-SVD algorithm. *J. Visual Communication and Image Representation*, 19(4):270–282, 2008.
11. Sjoerd Dirksen. Tail bounds via generic chaining. *Electron. J. Probab.*, 20(53):1–29, 2015.
12. Michael Elad and Michal Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, 2006.
13. Alan M. Frieze, Mark Jerrum, and Ravi Kannan. Learning linear transformations. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 359–368, 1996.
14. Navin Goyal, Santosh Vempala, and Ying Xiao. Fourier PCA and robust tensor decomposition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 584–593, 2014.
15. Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer-Verlag, 1991.
16. Yuanqing Li, Zhu Liang Yu, Ning Bi, Yong Xu, Zhenghui Gu, and S.-I. Amari. Sparse representation for brain signal processing: A tutorial on methods and applications. *Signal Processing Magazine, IEEE*, 31(3):96–106, May 2014. doi:10.1109/MSP.2013.2296790.
17. Kyle Luh and Van Vu. Random matrices: l_1 concentration and dictionary learning with few samples. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1409–1425, 2015.
18. Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60, 2010.
19. Julien Mairal, Francis R. Bach, and Jean Ponce. Sparse modeling for image and vision processing. *Foundations and Trends in Computer Graphics and Vision*, 8(2-3):85–283, 2014.

20 Julien Mairal, Francis R. Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Supervised dictionary learning. In *Proceedings of the 22nd Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1033–1040, 2008.

21 Julien Mairal, Francis R. Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Non-local sparse models for image restoration. In *IEEE 12th International Conference on Computer Vision (ICCV)*, pages 2272–2279, 2009.

22 Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. *J. Cryptology*, 22(2):139–160, 2009.

23 Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML)*, pages 759–766, 2007.

24 Daniel A. Spielman, Huan Wang, and John Wright. Exact recovery of sparsely-used dictionaries. In *The 25th Annual Conference on Learning Theory (COLT)*, pages 37.1–37.18, 2012. Full version: `http://arxiv.org/abs/1206.5882v1`.

25 Ju Sun, Qing Qu, and John Wright. Complete dictionary recovery over the sphere. *CoRR*, abs/1504.06785, 2015.

26 Michel Talagrand. *Upper and lower bounds for stochastic processes: modern methods and classical problems*. Springer, 2014.

27 Santosh Vempala and Ying Xiao. Max vs Min: Tensor decomposition and ICA with nearly linear sample complexity. In *Proceedings of The 28th Conference on Learning Theory (COLT)*, pages 1710–1723, 2015.

Approximation via Correlation Decay When Strong Spatial Mixing Fails*†‡§

Ivona Bezáková[1], Andreas Galanis[2], Leslie Ann Goldberg[3], Heng Guo[4], and Daniel Štefankovič[5]

1 **Rochester Institute of Technology, Rochester, NY, USA**
2 **University of Oxford, Oxford, UK**
3 **University of Oxford, Oxford, UK**
4 **Queen Mary University of London, London, UK**
5 **University of Rochester, Rochester, NY, USA**

Abstract

Approximate counting via correlation decay is the core algorithmic technique used in the sharp delineation of the computational phase transition that arises in the approximation of the partition function of anti-ferromagnetic two-spin models.

Previous analyses of correlation-decay algorithms implicitly depended on the occurrence of *strong spatial mixing*. This, roughly, means that one uses worst-case analysis of the recursive procedure that creates the sub-instances. In this paper, we develop a new analysis method that is more refined than the worst-case analysis. We take the shape of instances in the computation tree into consideration and we amortise against certain "bad" instances that are created as the recursion proceeds. This enables us to show correlation decay and to obtain an FPTAS even when strong spatial mixing fails.

We apply our technique to the problem of approximately counting independent sets in hypergraphs with degree upper-bound Δ and with a lower bound k on the arity of hyperedges. Liu and Lin gave an FPTAS for $k \geq 2$ and $\Delta \leq 5$ (lack of strong spatial mixing was the obstacle preventing this algorithm from being generalised to $\Delta = 6$). Our technique gives a tight result for $\Delta = 6$, showing that there is an FPTAS for $k \geq 3$ and $\Delta \leq 6$. The best previously-known approximation scheme for $\Delta = 6$ is the Markov-chain simulation based FPRAS of Bordewich, Dyer and Karpinski, which only works for $k \geq 8$.

Our technique also applies for larger values of k, giving an FPTAS for $k \geq 1.66\Delta$. This bound is not as strong as existing randomised results, for technical reasons that are discussed in the paper. Nevertheless, it gives the first deterministic approximation schemes in this regime. We further demonstrate that in the hypergraph independent set model, approximating the partition function is NP-hard even within the uniqueness regime.

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

Keywords and phrases approximate counting, independent sets in hypergraphs, correlation decay

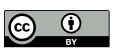
Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.45

* The full version of the paper is available at `arxiv.org/abs/1510.09193`. The theorem numbering here matches the full version.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

‡ Research supported by NSF grants CCF-0910415 and CCF-1319987.

§ This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 45; pp. 45:1–45:13

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

1 Introduction

We develop a new method for analysing correlation decays in spin systems. In particular, we take the shape of instances in the computation tree into consideration and we amortise against certain "bad" instances that are created as the recursion proceeds. This enables us to show correlation decay and to obtain an FPTAS even when strong spatial mixing fails. To the best of our knowledge, strong spatial mixing is a requirement for all previous correlation-decay based algorithms. To illustrate our technique, we focus on the computational complexity of approximately counting independent sets in *hypergraphs*, or equivalently on counting the satisfying assignments of monotone CNF formulas.

The problem of counting independent sets in *graphs* (denoted #IS) is extensively studied. A beautiful connection has been established, showing that approximately counting independent sets in graphs of maximum degree Δ undergoes a computational transition which coincides with the uniqueness phase transition from statistical physics on the infinite Δ-regular tree. The computational transition can be described as follows. Weitz [17] designed an FPTAS for counting independent sets on graphs with maximum degree at most $\Delta = 5$. On the other hand, Sly [15] proved that there is no FPRAS for approximately counting independent sets on graphs with maximum degree at most $\Delta = 6$ (unless NP = RP). The same connection has been established in the more general context of approximating the partition function of the hard-core model [17, 12, 15, 4, 5, 16] and in the even broader context of approximating the partition functions of generic antiferromagnetic 2-spin models [14, 5, 16, 8] (which includes, for example, the antiferromagnetic Ising model). As a consequence, the boundary for the existence of efficient approximation algorithms for these models has been mapped out.

Approximate counting via correlation decay is the core technique in the algorithmic developments which enabled the sharp delineation of the computational phase transition. Another standard approach for approximate counting, namely Markov Chains Monte Carlo (MCMC) simulation, is also conjectured to work up to the uniqueness threshold, but the current analysis tools that we have do not seem to be powerful enough to show that. For example, sampling independent sets via MCMC simulation is known to have fast mixing only for graphs with degree at most 4 [11, 3], rather than obtaining the true threshold of 5.

In this work, we consider counting independent sets in hypergraphs with upper-bounded vertex degree, and lower-bounded hyperedge size. A hypergraph $H = (V, \mathcal{F})$ consists of a vertex set V and a set $\mathcal{F}$ of hyperedges, each of which is a subset of V. A hypergraph is said to be *k-uniform* if every hyperedge contains exactly k vertices. Thus, a 2-uniform hypergraph is the same as a graph. We will consider the more general case where each hyperedge has arity at least k, rather than exactly k.

An independent set in a hypergraph H is a subset of vertices that does not contain a hyperedge as a subset. We will be interested in computing Z_H, which is the total number of independent sets in H (also referred to as the partition function of H). Formally, the problem of counting independent sets has two parameters – a degree upper bound Δ and a lower bound k on the arity of hyperedges. The problem is defined as follows (see also Section 2 for an equivalent formulation in terms of monotone CNF formulas).

Name #HyperIndSet(k, Δ).
Instance A hypergraph H with maximum degree at most Δ where each hyperedge has cardinality (arity) at least k.
Output The number Z_H of independent sets in H.

Previously, #HyperIndSet(k, Δ) has been studied using the MCMC technique by Borderwich, Dyer, and Karpinski [1, 2] (see also [3]). They give an FPRAS for all $k \geq \Delta + 2 \geq 5$

and for $k \geq 2$ and $\Delta = 3$. Despite equipping path coupling with optimized metrics obtained using linear programming, these bounds are not tight for small k. Liu and Lu [9] showed that there exists an FPTAS for all $k \geq 2$ and $\Delta \leq 5$ using the correlation decay technique.

Thus, the situation seems to be similar to the graph case – given the analysis tools that we have, correlation-decay brings us closer to the truth than the best-tuned analysis of MCMC simulation algorithms. On the other hand, the technique of Liu and Lu [9] does not extend beyond $\Delta = 5$. To explain why, we need to briefly describe the correlation-decay-based algorithm framework introduced by Weitz [17]. The main idea is to build a recursive procedure for computing the marginal probability that any given vertex is in the independent set. The recursion works by examining sub-instances with "boundary conditions" which require certain vertices to be in, or out, of the independent set. The recursion structure is called a "computation tree". Nodes of the tree correspond to intermediate instances, and boundary conditions are different in different branches. The computation tree allows one to compute the marginal probability exactly but the time needed to do so may be exponentially large since, in general, the tree is exponentially large. Typically, an approximate marginal probability is obtained by truncating the computation tree to logarithmic depth so that the (approximation) algorithm runs in polynomial time. If the correlation between boundary conditions at the leaves of the (truncated) computation tree and the marginal probability at the root decays exponentially with respect to the depth, then the error incurred from the truncation is small and the algorithm succeeds in obtaining a close approximation.

All previous instantiations under this framework require a property called *strong spatial mixing* (SSM), which roughly states that, conditioned on *any* boundary condition on intermediate nodes, the correlation decays (see Section 2.1 in the full version). SSM thus guards against the worst-case boundary conditions that might be created by the recursive procedure.

Let the $(\Delta - 1)$-ary k-uniform hypertree $\mathbb{T}_{k,\Delta}$ be the recursively-defined hypergraph in which each vertex has $\Delta - 1$ "descending" hyperedges, each containing $k - 1$ new vertices.

▶ **Observation 1.** Let $k \geq 2$. For $\Delta \geq 6$, strong spatial mixing *does not hold* on $\mathbb{T}_{k,\Delta}$.

Observation 1 follows from the fact that the infinite $(\Delta - 1)$-ary tree $\mathbb{T}_{2,\Delta}$ can be embedded in the hypertree $\mathbb{T}_{k,\Delta}$, and from well-known facts about the phase transition on $\mathbb{T}_{2,\Delta}$.

Observation 1 prevents the generalisation of Liu and Lu's algorithm [9] so that it applies for $\Delta \geq 6$, even with an edge-size lower bound k. The problem is that the construction of the computation tree involves constructing intermediate instances in which the arity of a hyperedge can be as small as 2. So, even if we start with a k-uniform hypergraph, the computation tree will contain instances with small hyperedges. Without strong spatial mixing, these small hyperedges cause problems in the analysis. Lu, Yang and Zhang [10] discuss this problem and say "How to avoid this effect is a major open question whose solution may have applications in many other problems." This question motivates our work.

To overcome this difficulty, we introduce a new amortisation technique in the analysis. Since lack of correlation decay is caused primarily by the presence of small-arity hyperedges within the intermediate instances, we keep track of such hyperedges. Thus, we track not only the correlation, but also combinatorial properties of the intermediate instances in the computation tree. Using this idea, we obtain the following result.

▶ **Theorem 2.** *There is an FPTAS for* $\#\mathsf{HyperIndSet}(3, 6)$.

Note that $\#\mathsf{HyperIndSet}(2, 6)$ is NP-hard to approximate due to [15], so our result is tight for $\Delta = 6$. This also shows that $\Delta = 6$ is the first case where the complexity of approximately counting independent sets differs on hypergraphs and graphs, as for $\Delta \leq 5$ both admit an

FPTAS [9]. Moreover, Theorem 2 is stronger than the best MCMC algorithm [2] when $\Delta = 6$ as [2] only works for $k \geq 8$.

We also apply our technique to large k.

▶ **Theorem 3.** *There exists a constant k_0 such that for all positive integers $k \geq k_0$ and Δ satisfying $k \geq 1.66\Delta$ there is an FPTAS for the problem* $\#\mathsf{HyperIndSet}(k, \Delta)$.

In the large k case, our result is worse than that obtained by analysis of the MCMC algorithm [2] ($k \geq 1.66\Delta$ rather than $k \geq \Delta + 2$) but it is incomparable since our algorithm is deterministic rather than randomised. The reason that our bound is worse is mainly due to technical difficulty in the analysis that we will explain shortly. In fact, we believe that, in the long run, analysis of the correlation-decay based algorithm is more likely to reveal the exact critical threshold than analysis of MCMC simulation, although other new ideas will probably be required in order to achieve sufficiently precise analysis.

The main technical difficulty in correlation-decay analysis is bounding a function that we call the "decay rate". This boils down to solving an optimization problem with $(k-1)(\Delta-1)$ variables. In previous work (e.g. [13]), this optimization has been solved using a so-called "symmetrization" argument, which reduces the problem to a univariate optimization via convexity. However, the many variables represent different branches in the computation tree. Since our analysis takes the shape of intermediate instances in the tree into consideration, the symmetrization argument does not work for us, and different branches take different values at the maximum. This problem is compounded by the fact that the shape of the sub-tree consisting of "bad" intermediate instances is heavily lopsided, and the assignment of variables achieving the maximum is far from uniform. Given these problems, there does not seem to be a clean solution to the optimization in our analysis. Instead of optimizing, we give an upper bound on the maximum decay rate. In Theorem 2, as k and Δ are small, the number of variables is manageable, and our bounds are much sharper than those in Theorem 3. On the other hand, because of this, the proof of Theorem 3 is much more accessible, and we will use Theorem 3 as a running example to demonstrate our technique.

We also provide some insight on the hardness side. Recall that for graphs it is NP-hard to approximate $\#\mathsf{IS}$ beyond the uniqueness threshold ($\Delta = 6$) [15]. We prove that it is NP-hard to approximate $\#\mathsf{HyperIndSet}(6, 22)$ (Corollary 29). In contrast, we show in the full version that uniqueness holds on the 6-uniform Δ-regular hypertree iff $\Delta \leq 28$ (Corollary 36). Thus, efficient approximation schemes cease to exist well below the uniqueness threshold on the hypertree. In fact, we show that this discrepancy grows exponentially in k: for large k, it is NP-hard to approximate $\#\mathsf{HyperIndSet}(k, \Delta)$ when $\Delta \geq 5 \cdot 2^{k/2}$ (Theorem 28 and Corollary 30), despite the fact that uniqueness holds on the hypertree for all $\Delta \leq 2^k/(2k)$ (Lemma 37 in the full version). Theorem 28 follows from a rather standard reduction to the hard-core model on graphs. Nevertheless, it demonstrates that the computational-threshold phenomena in the hypergraph case ($k > 2$) are different from those in the graph case ($k = 2$).

As mentioned earlier, there are models where efficient (randomised) approximation schemes exist (based on MCMC simulation) even though SSM does not hold. In fact, this can happen even when uniqueness does not hold. A striking example is the ferromagnetic Ising model (with external field). As [14] shows, there are parameter regimes where uniqueness holds but strong spatial mixing fails. It is easy to modify the parameters so that even uniqueness fails. Nevertheless, Jerrum and Sinclair [6] gave an MCMC-based FPRAS that applies for all parameters and for general graphs (with no degree bounds). It is still an open question to give a correlation decay based FPTAS for the ferromagnetic Ising model.

We conclude this section by giving an outline of the rest of the paper. In Section 2, we give some preliminaries. We reformulate $\#\mathsf{HyperIndSet}(k, \Delta)$ as the problem of counting

satisfying assignments in monotone CNF formulas, which will allow us to use the computation tree of Liu and Lu [9]. In Section 3, we give an overview of our proof approach, describing the main idea behind our new amortisation technique. In Section 4, we give the main ingredients which we use to prove Theorem 3 (large k). Section 5 describes the main lemma which yields Theorem 2 ($k = 3$). Section 6 gives the formal statements and proofs of the hardness results.

2 Preliminaries: monotone CNF formulas & the computation tree

The problem of counting the independent sets of a hypergraph has an equivalent formulation in terms of monotone CNF formulas. Given a hypergraph $H = (V, \mathcal{F})$, let C be a Boolean formula with variable set V. For each hyperedge, construct a clause which is the disjunction of all variables corresponding to vertices in the hyperedge. Let C be the conjunction of all such clauses. Note that C is a monotone formula – no variable is negated. Also, independent sets of H are in one-to-one correspondence with satisfying assignments of C – a variable is assigned value "true" in an assignment if and only if it is out of the corresponding independent set. Going the other direction, any monotone CNF formula can be viewed as a hypergraph. In the technical sections of this paper, we use the monotone CNF formulation and, in particular, the computation tree of Liu and Lu [9]. Below we give the relevant definitions and notation; our notation aligns as much as possible with that of [9].

Let C be an instance of a monotone CNF formula. We will denote the set of variables in C by V and set $n := |V|$. Variables in V will be denoted by $x_1, x_2, \ldots$ and clauses in C by $c_1, c_2, \ldots$. The arity of a clause c will be denoted by $|c|$, i.e., $|c|$ is the number of variables appearing in the clause c. We assume throughout that no variable appears twice in the same clause. For a variable $x \in V$, we denote by $d_x(C)$ the number of clauses where x appears. When x and C are clear from context, we will simply use d to denote $d_x(C)$. When C is clear from context, we will use Δ to denote $\max_{x \in V} d_x(C)$ and we will say that C is a formula with max degree Δ. Let $\mathcal{C}_{k,\Delta}$ be the set of all monotone CNF formulas which have max degree Δ and whose clauses have arity at least k. Note that some formulas in $\mathcal{C}_{k,\Delta}$ may have some clauses with arbitrarily large arities.

Our goal is to approximately count the number of satisfying assignments of a formula $C \in \mathcal{C}_{k,\Delta}$, which we denote by $Z(C)$. Since C is monotone, an assignment $\sigma : V \to \{0, 1\}$ is satisfying if, for every clause in C, there is at least one variable $x \in c$ with $\sigma(x) = 1$. Note that $Z(C) > 0$ since the all-1 assignment satisfies every monotone CNF formula. For convenience, we will use the simplified notation "$x = 1$" to denote (the set of) satisfying assignments of C in which x is set to 1, and we similarly use "$x = 0$". We associate the formula C with a probability distribution in which each satisfying assignment has probability mass $1/Z(C)$. We will denote probabilities with respect to this distribution by $\Pr_C(\cdot)$.

Let x be a variable in V. Define $R(C, x) := \frac{\Pr_C(x=0)}{\Pr_C(x=1)}$, this is well-defined since $\Pr_C(x = 1) > 0$ by the monotonicity of C. In fact, the monotonicity of C also implies that $0 \leq R(C, x) \leq 1$, where the upper bound follows from the fact that, for every satisfying assignment with $x = 0$, flipping the assignment of x to 1 does not affect satisfiability. Our interest in the quantity $R(C, x)$ stems from the following simple lemma (the proof follows the argument in [9, Appendix A] and is given for completeness in the full version).

▶ **Lemma 5.** *Let k and Δ be positive integers. Suppose that there is a polynomial-time algorithm (in n and $1/\varepsilon$) that takes an n-variable formula $C \in \mathcal{C}_{k,\Delta}$, a variable x of C, and an $\varepsilon > 0$ and computes a quantity $\widehat{R}(C, x)$ satisfying $|\widehat{R}(C, x) - R(C, x)| \leq \varepsilon$. Then, there exists an FPTAS which approximates $Z(C)$ for every $C \in \mathcal{C}_{k,\Delta}$.*

Liu and Lu [9] established that a computation tree approach gives a recursive procedure for *exactly* calculating $R(C, x)$ for any monotone CNF formula C and any variable $x \in C$. We next give the details of this recursive procedure (see [9, Lemma 5]). First, we introduce the following definitions (see Definitions 6 and 7 in the full version).

We call the variable x *forced* in C if x appears in a clause of arity 1 in C. We call the variable x *free* if x does not appear in any clause of C. We call the clause c *redundant* in C if there is a clause c' in C such that c is a (strict) superset of c' (note that removing c from C does not affect the set of satisfying assignments of C). We next give the details of the computation tree. The nodes in the computation tree will be pairs (C, x) such that

$$C \text{ is a monotone CNF formula and } x \text{ is a variable which is not forced in } C. \tag{1}$$

Let C, x satisfy (1). We first perform a pre-processing step on C which involves (i) initially removing all of the redundant clauses, (ii) then, removing all clauses of arity 1. Note that part (ii) of the preprocessing step removes all forced variables that were present in C; at the time of the removal, forced variables appear only in clauses of arity 1 since part (i) of the preprocessing step has already removed all redundant clauses in C (and hence all clauses of arity greater than 1 that contain forced variables). Denote the formula after the completion of the preprocessing step by $\widetilde{C}$. Note that every clause in $\widetilde{C}$ is also a clause in the initial formula C, so x is not forced in $\widetilde{C}$. Further, since removing redundant clauses does not change the set of satisfying assignments of C, and x is not forced in C, we have that $R(\widetilde{C}, x) = R(C, x)$.

If x is free in $\widetilde{C}$ (the formula after the pre-processing step), then the start node (C, x) is (declared) a leaf of the computation tree (note that in this case $R(C, x) = 1$). In the sequel, we assume that x is not free in $\widetilde{C}$. Denote by $\{c_i\}_{i\in[d]}$ the clauses where x occurs in $\widetilde{C}$ and let $w_i = |c_i| - 1$ (note that $d \geq 1$). We will use $\mathbf{w}$ to denote the vector $(w_1, \ldots, w_d)$. The variables in clause c_i other than x will be denoted by $x_{i,1}, \ldots, x_{i,w_i}$. For the pair (C, x), we next construct pairs $(C_{i,j}, x_{i,j})$ for $i \in [d]$ and $j \in [w_i]$, where $C_{i,j}$ is an appropriate subformula obtained from $\widetilde{C}$, roughly, by hard-coding (some of) the occurrences of the variables in $\widetilde{C}$ to either 1 or 0 (this will be explained below and will henceforth be referred to as pinning)[1].

Precisely, for $i \in [d]$, let C_i be the formula obtained from $\widetilde{C}$ by removing clauses $c_1, \ldots, c_{i-1}$ (note that this has the same effect as pinning the occurrences of x in these clauses to 1) and pinning the occurrences of x in $c_{i+1}, \ldots, c_d$ to 0. For $j \in [w_i]$, the formula $C_{i,j}$ is obtained from C_i by further removing clause c_i and pinning all the occurrences of $x_{i,1}, \ldots, x_{i,j-1}$ to 0.

In the full version, we prove that the pairs $(C_{i,j}, x_{i,j})$ satisfy (1) for all $i \in [d]$, $j \in [w_i]$. We next state the relation between $R(C, x)$ and the $R(C_{i,j}, x_{i,j})$'s. It is proved in [9, Lemma 5], using a Weitz-type telescopic expansion (see also the full version for the argument), that

$$R(C, x) = \prod_{i=1}^{d} \left(1 - \prod_{j=1}^{w_i} \frac{R(C_{i,j}, x_{i,j})}{1 + R(C_{i,j}, x_{i,j})}\right). \tag{2}$$

By applying (2) recursively, it is not hard to see that one can compute the quantity $R(C, x)$ *exactly*. Of course, exact computation using this scheme will typically require exponential time, so as in [9] we will stop the recursion at some (small) depth L to keep the computations feasible within polynomial time. This will yield a quantity $R(C, x, L)$ and the hope is that, by choosing L appropriately, the error $|R(C, x, L) - R(C, x)|$ will be sufficiently small.

[1] Note that our notation for i, j is different from the one in [9]; there, the roles of i, j are interchanged.

In light of (2), and analogously to [9], we define $R(C, x, L)$ for integer $L \geq 0$ as follows. Namely, for integer L we set

$$R(C,x,L) = \begin{cases} 1, & \text{if } x \text{ is free in } \widetilde{C} \text{ or } L \leq 0, \\ \prod_{i=1}^{d} \left(1 - \prod_{j=1}^{w_i} \frac{R(C_{i,j}, x_{i,j}, L-l_{w_i})}{1+R(C_{i,j}, x_{i,j}, L-l_{w_i})}\right), & \text{otherwise,} \end{cases} \tag{4}$$

where $l_{w_i} := \lceil \log_6(w_i + 1) \rceil$ (note that $l_1 = \ldots = l_5 = 1$). The choice of l_{w_i} is such that the size of the computation tree is polynomial for $L = O(\log n)$; the particular choice of the logarithm base in the definition of l_{w_i} is not important as long as it is a big enough constant.

Using correlation decay techniques together with a new method to account for the shape of the computation tree, we will show the following key lemma (proved in Section 5).

▶ **Lemma 10.** *Let $\Delta = 6$. There exist constants α, τ with $0 < \alpha < 1$ and $\tau > 0$ such that the following holds for all integers L. Let C be a monotone CNF formula whose clauses all have arity greater than or equal to 3 and, further, each variable occurs in at most Δ clauses. Then, for the quantity $R(C, x, L)$ defined recursively from (4), it holds that*

$$|R(C,x,L) - R(C,x,\infty)| \leq \tau\alpha^L.$$

In the following section, we give an overview of our approach to proving Lemma 10. We also prove an analogous lemma in the case where k, Δ are large, see Lemma 11 in Section 4.

Proof of Theorems 2 and 3 assuming Lemmas 10 and 11. Invoke Lemmas 5, 10 and 11 and the reformulation of $\#\mathsf{HyperIndSet}(k, \Delta)$ in terms of the monotone CNF problem. ◀

3 Proof Approach

To prove Lemma 10, the standard approach so far in the literature has been to show that, for a node (C, x) in the computation tree, the quantity $|R(C, x, L) - R(C, x, \infty)|$ is bounded by $\alpha \max_{i,j} |R(C_{i,j}, x_{i,j}, L-1) - R(C_{i,j}, x_{i,j}, \infty)|$ for some constant $0 < \alpha < 1$ and then, inductively, to deduce that $|R(C, x, L) - R(C, x, \infty)|$ decays exponentially in L. This approach has been extremely successful when strong spatial mixing holds [14, 8, 9, 13, 18, 10].

In our setting, this inductive approach is problematic since, inside the computation tree, we are faced with the possibility that the formula at the root of a subtree has many arity-2 clauses. For $\Delta \geq 6$, these subtrees prohibit the application of the above proof scheme since they are in non-uniqueness and hence the desired step-by-step decay is no longer present.

While arity-2 clauses are problematic, clauses with larger arity do at least lead to good decay of correlation in a single step. Thus, our approach is to do an amortised analysis. In a single step, we track both the one-step decay of correlation and the immediate creation of arity-2 clauses, which will later lead to worse decay. More formally, instead of tracking the quantity $R(C, x, L)$, we track the quantity $m(C, x, L) = \delta^{b(C)} R(C, x, L)$ where $b(C)$ denotes the number of arity-2 clauses in the formula C (the "bad" clauses) and $\delta \in (0, 1)$ is an appropriate constant. (In fact, in Section 5, we define $m(C, x, L)$ as $\delta^{b(C)}\Phi(R(C, x, L))$ for an appropriate function Φ, see (28).)

Crucially, note that the root formula C satisfies $|m(C, x, L) - m(C, x, \infty)| = |R(C, x, L) - R(C, x, \infty)|$, since by the assumption in Lemma 10 we have that $b(C) = 0$. Thus, the key step in the proof of Lemma 10 is to show that the quantity $|m(C, x, L) - m(C, x, \infty)|$ decays exponentially with L; we will show that, for some constant $\alpha \in (0, 1)$, for an arbitrary node (C, x) in the computation tree, it holds that

$$|m(C,x,L) - m(C,x,\infty)| \leq \alpha \max_{i,j} |m(C_{i,j}, x_{i,j}, L-1) - m(C_{i,j}, x_{i,j}, \infty)|. \tag{5}$$

Since arity-2 clauses are the source of problems it is important not to create too many of them. Therefore, we also have to be careful in the construction of the computational tree. We achieve this by carefully ordering the clauses that we process at each step to avoid creating arity-2 clauses as much as possible.

Unfortunately, the quantity $m(C, x, L)$ is more complicated than the plain message $R(C, x, L)$ which has been studied before since it incorporates combinatorial information about the formula C and thus it does not satisfy a simple recursion (unlike $R(C, x, L)$). Nevertheless, we are able to define a multi-variable quantity κ (see (32)) and to show that when $\kappa \leq 1$, inequality (5) holds.

The technical details of applying the approach are quite intricate in the context of Lemma 10. In Section 4, we first apply the approach to the case of large k, where the proof is (significantly) shorter. There, instead of tracking the number of clauses of arity 2, we track (roughly) the aggregate arities of the clauses in C. Other than that, the high-level proof approach is similar to what is described above. In Section 5, we give an outline of the more difficult proof of Lemma 10.

4 The case of large k

In this section, we show the following lemma. Let $c := 0.565$ and $\beta \sim 1.65115$ be the solution to $2^{0.0001}c^\beta = 2 \times 0.9997(1 - c^\beta)c^2$.

▶ **Lemma 11.** *Let $\beta \sim 1.65$ be defined as above. Let k be a sufficiently large positive integer and let Δ be a positive integer satisfying $k \geq \beta\Delta + 3$. There are real numbers α and τ satisfying $0 < \alpha < 1$ and $\tau > 0$ such that for every $C \in \mathcal{C}_{k,\Delta}$ and every integer L,*

$$|R(C, x, L) - R(C, x, \infty)| \leq \tau\alpha^L, \tag{6}$$

where the quantity $R(C, x, L)$ is defined recursively from (4)*.*

To estimate the error $|R(C, x, L) - R(C, x, \infty)|$, we will track a specific quantity $m(C, x, L)$ which is assigned to each node in the computation tree. Let C be the original monotone CNF formula and let (N, x) be a node in the computation tree. As explained earlier, each clause c' of N is obtained from a clause c of C by pinning a certain number of variables to 0 (possibly none), which effectively is the same as removing those variables. We call these 0-pinnings *deficits* and let $\max\{0, k - |c'|\}$ be number of deficits of c'. Note that a clause of arity larger than k is considered to have no deficit, although some variables of it may have been pinned to 0. Let $D(N) = \sum_{c' \in N} \max\{0, k - |c'|\}$ denote the total deficits of N. Also observe that if a clause c of C does not show up in N, it does not contribute any deficits. For any node (N, x) in the computation tree, let $m(N, x, L) := \delta^{D(N)} R(N, x, L)$ where $\delta \in (0, 1)$ is a constant that we will choose later.

Now let us calculate how the number of deficits changes in one step of the recursion. Let (N, x) be a node in the computation tree. Note that pinning any variable to 1 will remove all clauses containing it, and will therefore eliminate all deficits of these clauses. Moreover, for a clause of arity 2, pinning any of its variables to either 0 or 1 will eliminate the whole clause due to the preprocessing step. Let $b_2(N, x)$ (or simply b_2 when N and x are clear from the context) denote the number of arity-2 clauses containing x in N. In the recursion, we will always order the clauses so that arity-2 clauses come last. Thus, clauses $c_1, \ldots, c_{d-b_2}$ each have arity at least 3 and clauses $c_{d-b_2+1}, \ldots, c_d$ each have arity 2. Due to these arity-2 clauses, the deficits in every branch below (N, x) will decrease by at least $b_2(k - 2)$.

Now consider an integer i in the range $1 \leq i \leq d - b_2$. Recall that in N_i, we pin appearances of x prior to i to 1, thus eliminating deficits in clauses $c_1, \ldots, c_{i-1}$. Together with the removal

of clause c_i, these removals decrease the total deficits by $\sum_{t=1}^{i} \max\{0, k-1-w_t\}$, where $w_t = |c_t| - 1$. Let $s_i = \sum_{t=1}^{i} \max\{0, k-1-w_t\}$. We also pin all appearances of x after i to 0, increasing the total number of deficits by at most $d - i - b_2$. Here we have a "$-b_2$" because pinning a variable to 0 in a arity-2 clause does not increase the deficits. Moreover, in $N_{i,j}$, we pin all appearances of $j-1$ other variables to 0. Each of these pinnings may contribute at most $\Delta - 1$ deficits, giving a total increase of at most $(\Delta-1)(j-1)$. Next consider $i \geq d - b_2 + 1$. For such an i, clause c_i has arity 2, and it is easy to see that deficits do not increase in the branch corresponding to $N_{i,j}$. Also note that the preprocessing steps do not increase deficits. Hence we have the following upper bounds on $D(N_{i,j})$ for $i \in [d]$:

$$D(N_{i,j}) \leq \begin{cases} D(N) - b_2(k-2) - s_i + d - i - b_2 + (\Delta-1)(j-1) & \text{if } 1 \leq i \leq d - b_2, \\ D(N) - b_2(k-2) - s_{d-b_2} & \text{if } d - b_2 + 1 \leq i. \end{cases} \tag{7}$$

The key to our analysis is to bound the correlation decay of $m(C, x, L)$. We will analyse the recursion for $m(C, x, L)$ based on that of $R(C, x, L)$. Recall that the recursion for $R(C, x, L)$ depends on the function $F^{d,\mathbf{w}}(\mathbf{r})$ implicitly defined by (4), i.e.,

$$F^{d,\mathbf{w}}(\mathbf{r}) := \prod_{i=1}^{d} \left(1 - \prod_{j=1}^{w_i} \frac{r_{i,j}}{1 + r_{i,j}}\right), \tag{8}$$

where $r_{i,j} \in [0,1]$ for all $i \in [d]$, $j \in [w_i]$ (so $R(C, x, L) = F^{d,\mathbf{w}}(\{R(C_{i,j}, x_{i,j}, L - l_{w_i})\})$).

For $i \in [d]$, the following quantity $\rho_{\delta,\alpha}^{\mathbf{w},i}$ will roughly upper bound the sensitivity of $|m(C, x, L) - m(C, x, \infty)|$ to the i-th clause in which x appears in C, or more precisely, to the quantities $|m(C_{i,j}, x_{i,j}, L - l_{w_i}) - m(C_{i,j}, x_{i,j}, \infty)|$ for $j \in [w_i]$:

$$\rho_{\delta,\alpha}^{\mathbf{w},i}(\mathbf{r}) := \begin{cases} \alpha^{-l_{w_i}} \delta^{s_i + i - d + b_2} \sum_{j=1}^{w_i} \delta^{-(j-1)(\Delta-1)} \left|\frac{\partial F^{d,\mathbf{w}}}{\partial r_{i,j}}\right| & \text{if } 1 \leq i \leq d - b_2, \\ \alpha^{-l_{w_i}} \delta^{s_{d-b_2}} \left|\frac{\partial F^{d,\mathbf{w}}}{\partial r_{i,1}}\right| & \text{if } d - b_2 + 1 \leq i \leq d. \end{cases} \tag{10}$$

Note that $\rho_{\delta,\alpha}^{\mathbf{w},i}(\mathbf{r})$ depends also on $r_{i',j}$ with $i' \neq i$. The decay rate of $|m(C, x, L) - m(C, x, \infty)|$ in terms of L, for all formulas C with max degree Δ, will be captured by the aggregation of $\rho_{\delta,\alpha}^{\mathbf{w},i}(\mathbf{r})$'s, namely, $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r}) := \delta^{b_2(k-2)} \sum_{i=1}^{d} \rho_{\delta,\alpha}^{\mathbf{w},i}(\mathbf{r})$. The main technical lemma of the section is the following.

▶ **Lemma 12.** *For all sufficiently large Δ, for any $k \geq \beta\Delta + 3$ where $\beta \sim 1.65115$ is defined earlier, there exists constants $0 < \delta < 1$, $0 < \alpha < 1$, and $U > 0$ such that, for all $\mathbf{0} \leq \mathbf{r} \leq \mathbf{1}$, $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})$ is at most 1 when $d \leq \Delta - 1$ and at most U when $d = \Delta$.*

Before sketching the proof of Lemma 12, let's show that it is sufficient to imply Lemma 11.

Proof of Lemma 11 (Sketch). Let δ, α be as in Lemma 12. For $C \in \mathcal{C}_{k,\Delta}$, we have $D(C) = 0$, so $|m(C, x, L) - m(C, x, \infty)| = |R(C, x, L) - R(C, x, \infty)|$.

The lemma will thus follow by showing that for every node (N, x) of the computation tree the quantity $|m(N, x, L) - m(N, x, \infty)|$ decays roughly as α^L. The proof is by induction on L; the base cases $L \leq 0$ are easy to show since $\delta \in (0, 1)$. For the induction step, one needs the following key inequality (obtained by considering the gradient of $F^{d,\mathbf{w}}$):

$$|m(N, x, L) - m(N, x, \infty)| \leq \max_{\mathbf{r}} \left\{ \left(\sum_{i=1}^{d} \sum_{j=1}^{w_i} \delta^{D(N) - D(N_{i,j})} \alpha^{-l_{w_i}} \left| \frac{\partial F^{d,\mathbf{w}}(\mathbf{r})}{\partial r_{i,j}} \right| \right) \right\} \times \max_{i,j} \left\{ \alpha^{l_{w_i}} \left| m(N_{i,j}, x_{i,j}, L - l_{w_i}) - m(N_{i,j}, x_{i,j}, \infty) \right| \right\}. \tag{16}$$

Our bounds (7) on $D(N)-D(N_{i,j})$ for $i\in[d]$, $j\in[w_i]$ imply that the $\max_{\mathbf{r}}\{\cdot\}$ expression in (16) is upper bounded by $\max_{\mathbf{r}}\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})$ (since $\delta\in(0,1)$). Thus, by Lemma 12, one obtains step-wise decay of $m(N,x,L)$. (The case $d=\Delta$ arises only at the root formula.) ◀

Proof of Lemma 12 (Sketch). Let $\alpha=0.9999$ and δ be such that $\delta^{\Delta}=c$ (recall, $c=0.565$).

The proof has two main steps. In the first step, one obtains a bound on $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})$ that does not depend on $\mathbf{r}$ (i.e., we bound the terms which depend on the "messages" $R(C,x,L)$). We do this for each clause c_i; the bound we obtain depends only on the arity of the clause, or, for the purposes of this proof, on w_i. The second step is then to maximize over the possible values of the w_i's.

For $w_i\geq 2$, let $\mu(w_i):=\max\{2,w_i\left(2-2^{1-w_i}\right)^{-1}\}$. The result of the first step is the following bound on $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})$ (note that $\mathbf{r}$ is completely eliminated).

$$\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})\leq\delta^{b_2(k-2)}\left(\sum_{i=1}^{d-b_2}\alpha^{-l_{w_i}}\delta^{s_i+i-d+b_2-(w_i-1)(\Delta-1)}2^{-w_i}\mu(w_i)+\alpha^{-1}b_2\right),\tag{21}$$

Using the fact that $\alpha^{-l_{w_i}}\leq\alpha^{-1}w_i^{0.0001}$ for any $w_i\geq 2$, we obtain that, if $w_i>k-1$, then the right hand side of (21) increases if we replace w_i with $k-1$. Henceforth we may assume that $w_i\leq k-1$, which allows us to rewrite $s_i=s_{i-1}+k-1-w_i$. Using the fact that $\delta^{\Delta}=c$, we then obtain

$$\alpha\,\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})\leq\delta^{k+b_2(k-1)}\eta_d(\mathbf{w})+b_2\delta^{b_2(k-2)},\quad\eta_d(\mathbf{w}):=\sum_{i=1}^{d-b_2}\delta^{s_{i-1}}w_i^{0.0001}(2c)^{-w_i}\mu(w_i).\tag{22}$$

Convexity arguments show that the maximum of $\eta_d(\mathbf{w})$ is achieved at $\mathbf{w}=\mathbf{w}'$, where $w_i'=k-1$ for $1\leq i\leq t$ and $w_i'=2$ for $t+1\leq i\leq d-b_2$ for some $0\leq t\leq d-b_2-1$. We thus obtain

$$\eta_d(\mathbf{w})\leq\eta_d(\mathbf{w}')\leq\Delta(k-1)^{1.0001}(2c)^{-(k-1)}\left(2-2^{2-k}\right)^{-1}+\frac{2^{0.0001}}{2c^2(1-\delta^{k-3})}\leq 0.9998c^{-\beta},\tag{26}$$

where the last inequality follows from $\frac{2^{0.0001}}{2c^2(1-\delta^{k-3})}\leq\frac{2^{0.0001}}{2c^2}\cdot\frac{1}{1-c^{\beta}}=0.9997c^{-\beta}$ (using the definition of c,δ) and that for large Δ and k, we have $\Delta(k-1)^{1.0001}(2c)^{-(k-1)}\left(2-2^{2-k}\right)^{-1}<0.0001$ (using that $2c>1$ and $k\geq\beta\Delta+3$). Now, plug (26) into (22):

$$\alpha\,\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})\leq\delta^{b_2(k-2)}\left(0.9998\,\delta^k c^{-\beta}+b_2\right)\leq c^{\beta b_2}\left(0.9998+b_2\right)\leq\alpha,\tag{27}$$

where the last inequality follows from $c^{\beta}<1/2$. (27) yields $\kappa_{\delta,\alpha}^{d,b_2,\mathbf{w}}(\mathbf{r})\leq 1$, as claimed. ◀

5 A finer analysis to treat $k\geq 3$, $\Delta=6$

In this section, we describe the main technical ingredient to prove Lemma 10 (which in turn was critical in deducing Theorem 2 in Section 2).

Let (C,x) be a non-leaf node in the computation tree, where the root node is a monotone CNF formula with max degree $\Delta=6$ all of whose clauses have arity at least $k=3$. Denote by $b(C)$ the total number of clauses of arity 2 in C. Recall that $\widetilde{C}$ is the formula obtained from C after the preprocessing step. Also, $c_1,\ldots,c_d$ are the clauses in which x appears in $\widetilde{C}$ and $w_i=|c_i|-1$. We have $d\geq 1$ since (C,x) is a non-leaf node and $w_i\geq 1$ for all i because of property (1). Let b_3 denote the number of clauses such that $|c_i|=3$ and let b_2 denote the number of clauses such that $|c_i|=2$.

When processing the node (C, x), we will order the clauses in which x appears in $\widetilde{C}$ so that clauses with $|c_i| = 3$ are processed first. We thus have that $d, b_2, b_3 \in \mathbb{Z}$ and $\mathbf{w} = (w_1, \ldots, w_d) \in \mathbb{Z}^d$ satisfy

$$\begin{aligned} &1 \le d, \quad 0 \le b_2, b_3 \le d, \quad b_2 + b_3 \le d \\ &w_i = 2 \text{ for } i = 1, \ldots, b_3, \quad w_i \ge 3 \text{ or } w_i = 1 \text{ for } i = b_3 + 1, \ldots, d. \end{aligned} \tag{30}$$

As already described in Section 3, to prove Lemma 10, we will track a quantity $m(C, x, L)$ which is assigned to each node in the computation tree. Let $\eta := \eta(\Delta) = (1/2)^\Delta$. Define $m(C, x, L)$ by

$$m(C, x, L) := \delta^{b(C)}\, \Phi(R(C, x, L)) \tag{28}$$

where $\delta \in (0, 1]$, and $\Phi : [\eta, 1] \to \mathbb{R}$ satisfies:

$$\Phi \text{ is continuously differentiable on } [\eta, 1], \text{ and } \varphi := \Phi' \text{ satisfies } \varphi(z) > 0 \text{ for } z \in [\eta, 1]. \tag{29}$$

For $\Delta = 6$, the value of δ will be later chosen to be $\delta = 9789/10000$ and Φ will be specified in the upcoming equation (46). Also, note that for all nodes (C, x) in the computation tree the quantity $m(C, x, L)$ is well-defined; this is because of the property (1) and the lower bound $R(C, x, L) \ge \eta$ (follows from (4), see Remark 8 in the full version).

We will show Lemma 10 with $\alpha := 1 - 10^{-4}$. In particular, we aim to show that α upper bounds the decay rate of $|m(C, x, L) - m(C, x, \infty)|$, i.e., $|m(C, x, L) - m(C, x, \infty)|$ decays roughly as α^L (modulo a multiplicative constant). As in the proof of Lemma 11, this yields Lemma 10 when applied to the root formula of the tree (since it has no arity-2 clauses).

Ler $\mathbf{r}$ be a real vector with components $r_{i,j}$ for $i \in [d]$ and $j \in [w_i]$, $\mathbf{0} \le \mathbf{r} \le \mathbf{1}$. For $i \in [d]$, we will use the following analogue of the quantity $\rho^{\mathbf{w},i}_{\delta,\alpha}(\mathbf{r})$ defined in Section 4 (see (10)):

$$\rho^{\mathbf{w},i}_{\delta,\Phi,\alpha}(\mathbf{r}) := \alpha^{-l_{w_i}} \sum_{j=1}^{w_i} \left(\frac{1}{\delta}\right)^{(j-1)(\Delta-1)} \frac{1}{\varphi(r_{i,j})} \left|\frac{\partial F^{d,\mathbf{w}}}{\partial r_{i,j}}\right|, \tag{31}$$

where recall that the function $F^{d,\mathbf{w}}(\mathbf{r})$ gives the recursion (4) that $R(\cdot,\cdot,\cdot)$ satisfies (cf. (8)). Once again, the quantity $\rho^{\mathbf{w},i}_{\delta,\Phi,\alpha}(\mathbf{r})$ will roughly upper bound the sensitivity of $|m(C, x, L) - m(C, x, \infty)|$ to the quantities $|m(C_{i,j}, x_{i,j}, L) - m(C_{i,j}, x_{i,j}, \infty)|$ for $j \in [w_i]$. The decay rate of $|m(C, x, L) - m(C, x, \infty)|$ will be captured by the following quantity:

$$\kappa^{d,b_2,b_3,\mathbf{w}}_{\delta,\Phi,\alpha}(\mathbf{r}) := \varphi(F^{d,\mathbf{w}}(\mathbf{r}))\, \delta^{b_2} \left(\sum_{i=1}^{b_3} \left(\frac{1}{\delta}\right)^{b_3 - i} \rho^{\mathbf{w},i}_{\delta,\Phi,\alpha}(\mathbf{r}) + \sum_{i=b_3+1}^{d} \rho^{\mathbf{w},i}_{\delta,\Phi,\alpha}(\mathbf{r}) \right). \tag{32}$$

Finally, we specify the function Φ. For $z \in (0, 1]$, let

$$\Phi(z) := \frac{1}{\chi\psi} \log\left(\frac{z^\chi}{\psi - z^\chi}\right) \text{ with } \chi = 1/2,\ \psi = 13/10, \text{ so } \varphi(z) = \Phi'(z) = \frac{1}{z(\psi - z^\chi)}. \tag{46}$$

Our main technical lemma is the following (proved in Section 6 of the full version).

▶ **Lemma 14.** *Let $\Delta = 6$, $\chi = 1/2$, $\psi = 13/10$, $\delta = 9789/10000$, $\alpha = 1 - 10^{-4}$ and Φ be defined from (46). There exists a constant $U > 0$ such that, for all $d, b, \mathbf{w}$ satisfying (30), for all $(1/2)^{\Delta-1}\mathbf{1} \le \mathbf{r} \le \mathbf{1}$, it holds that $\kappa^{d,b_2,b_3,\mathbf{w}}_{\delta,\Phi,\alpha}(\mathbf{r})$ is at most 1 when $d \le \Delta - 1$ and at most U when $d = \Delta$.*

Lemma 10 can be derived from Lemma 14 using an argument analogous to the one used to derive Lemma 11 from Lemma 12, see Section 5 in the full version for the details.

The proof of Lemma 14 is significantly harder however than the proof of Lemma 12. For one thing, the analysis has to account for the (irrational) function φ (without which the step-wise contraction is simply not true) present in the expression for $\kappa^{d,b_2,b_3,\mathbf{w}}_{\delta,\Phi,\alpha}(\mathbf{r})$. Further, the two-step procedure that we followed in the proof of Lemma 12, that is, maximising first over $\mathbf{r}$ and then maximising over $\mathbf{w}$, is far too crude. Instead, we need to slowly reduce the number of the variables akin to the approach in [9]. In our case, however, to bound tightly the contribution of δ, we need to prove asymmetric inequalities in several variables with irrational expressions (see the key Lemmas 15, 19 and 20 in the full version). To keep the length of the paper reasonable and easier to read through, we rigorously verify certain two-variable inequalities using a computer; as the reader will notice, in some cases, even reducing to such two-variable inequalities requires a significant analytical effort (see the proofs of the aforementioned lemmas, for example).

6 Hardness Results

We prove the hardness results stated in the Introduction. We will work with the problem $\#\mathsf{HyperIndSet}(k,\Delta)$ (instead of the monotone CNF formulation). The proof is via a reduction to the independent set model on graphs which was used by Bordewich *et al.* [2]. The precise inapproximability results for the hard-core model had not yet been proved at the time [2] was written, so we carry out the details explicitly to obtain the bound that their reduction gives.

We use the inapproximability result of Sly and Sun [16] for the hard-core model. Recall, for a graph $G=(V,E)$, the partition function of G in the hard-core model with parameter $\lambda>0$ is given by $Z_G(\lambda):=\sum_I \lambda^{|I|}$ where the sum ranges over all independent sets I of G.

▶ **Theorem 27** ([16]). *For $\Delta\geq 3$, let $\lambda_c(\Delta):=(\Delta-1)^{\Delta-1}/(\Delta-2)^{\Delta}$. For all $\lambda>\lambda_c(\Delta)$, it is* NP*-hard to approximate $Z_G(\lambda)$ on Δ-regular graphs G, even within an exponential factor.*

▶ **Theorem 28.** *Let $k\geq 2$, $\Delta\geq 3$ be integers. Suppose that $2^{\lceil k/2\rceil}-1<\frac{(\Delta-2)^{\Delta}}{(\Delta-1)^{\Delta-1}}$. Then, it is* NP*-hard to approximate $\#\mathsf{HyperIndSet}(k,\Delta)$, even within an exponential factor.*

Proof (Sketch). Let $\lambda:=1/(2^{\lceil k/2\rceil}-1)$. Let G be a Δ-regular n-vertex graph. Construct a hypergraph H by replacing every vertex of G by a (distinct) set of $\lceil k/2\rceil$ vertices and place a hyperedge among the corresponding sets of vertices for each edge of G. Then $Z_H=(2^{\lceil k/2\rceil}-1)^n\,Z_G(\lambda)$, from which the theorem easily follows (see full version for details). ◀

▶ **Corollary 29.** *Let $k=6$, $\Delta=22$. It is* NP*-hard to approximate $\#\mathsf{HyperIndSet}(k,\Delta)$.*

▶ **Corollary 30.** *Let $k\geq 2$. For $\Delta\geq 5\cdot 2^{k/2}$, it is* NP*-hard to approximate $\#\mathsf{HyperIndSet}(k,\Delta)$.*

References

1 M. Bordewich, M. E. Dyer, and M. Karpinski. Path coupling using stopping times and counting independent sets and colorings in hypergraphs. *Random Struct. Algorithms*, 32(3):375-399, 2008.

2 M. Bordewich, M. E. Dyer, and M. Karpinski. Stopping Times, Metrics and Approximate Counting. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP 2006)*, LNCS 4051:108-119, 2006.

3 M. Dyer and C. Greenhill. On Markov Chains for Independent Sets. *J. Algorithms*, 35(1):17-49, 2000.

4 A. Galanis, Q. Ge, D. Štefankovič, E. Vigoda, and L. Yang. Improved inapproximability results for counting independent sets in the hard-core model. *Random Struct. Algorithms*, 45(1):78-110, 2014.

5 A. Galanis, D. Štefankovič, and E. Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. *CoRR*, abs/1203.2226, 2012. Preprint is available from the arXiv at: http://arxiv.org/abs/1203.2226.

6 M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the Ising model. *SIAM J. Comput.*, 22(5):1087–1116, 1993.

7 L. Li, P. Lu, and Y. Yin. Approximate counting via correlation decay in spin systems. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 922–940, 2012.

8 L. Li, P. Lu, and Y. Yin. Correlation Decay up to Uniqueness in Spin Systems. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 67-84, 2013.

9 J. Liu and P. Lu, FPTAS for Counting Monotone CNF. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1531–1548, 2015. Full version available from the arXiv at http://arxiv.org/abs/1311.3728.

10 P. Lu, K. Yang, and C. Zhang. FPTAS for Hardcore and Ising Models on Hypergraphs. Available from the arXiv at http://arxiv.org/abs/1509.05494

11 M. Luby and E. Vigoda. Fast convergence of the Glauber dynamics for sampling independent sets. *Random Struct. Algorithms*, 15(3-4):229-241, 1999.

12 E. Mossel, D. Weitz, and N. Wormald. On the hardness of sampling independent sets beyond the tree threshold. *Probab. Theory Related Fields*, 143(3-4):401–439, 2009.

13 A. Sinclair, P. Srivastava, D. Štefankovič, and Yitong Yin. Spatial mixing and the connective constant: Optimal bounds. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 1549–1563, 2015.

14 A. Sinclair, P. Srivastava, and M. Thurley. Approximation algorithms for two-state antiferromagnetic spin systems on bounded degree graphs. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA), 941–953, 2012.

15 A. Sly. Computational Transition at the Uniqueness Threshold. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science* (FOCS), 287–296, 2010.

16 A. Sly and N. Sun. The computational hardness of counting in two-spin models on d-regular graphs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 361–369, 2012.

17 D. Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing* (STOC), 140–149, 2006.

18 Y. Yin and C. Zhang. Spatial mixing and approximate counting for Potts model on graphs with bounded average degree. Available from the arXiv at http://arxiv.org/abs/1507.07225

A Complexity Trichotomy for Approximately Counting List H-Colourings*†‡§

Andreas Galanis[1], Leslie Ann Goldberg[2], and Mark Jerrum[3]

1 **University of Oxford, Oxford, UK**
2 **University of Oxford, Oxford, UK**
3 **Queen Mary University of London, London, UK**

Abstract

We examine the computational complexity of approximately counting the list H-colourings of a graph. We discover a natural graph-theoretic trichotomy based on the structure of the graph H. If H is an irreflexive bipartite graph or a reflexive complete graph then counting list H-colourings is trivially in polynomial time. Otherwise, if H is an irreflexive bipartite permutation graph or a reflexive proper interval graph then approximately counting list H-colourings is equivalent to #BIS, the problem of approximately counting independent sets in a bipartite graph. This is a well-studied problem which is believed to be of intermediate complexity – it is believed that it does not have an FPRAS, but that it is not as difficult as approximating the most difficult counting problems in #P. For every other graph H, approximately counting list H-colourings is complete for #P with respect to approximation-preserving reductions (so there is no FPRAS unless NP = RP). Two pleasing features of the trichotomy are (i) it has a natural formulation in terms of hereditary graph classes, and (ii) the proof is largely self-contained and does not require any universal algebra (unlike similar dichotomies in the weighted case). We are able to extend the hardness results to the bounded-degree setting, showing that all hardness results apply to input graphs with maximum degree at most 6.

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

Keywords and phrases approximate counting, graph homomorphisms, list colourings

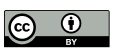
Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.46

1 Overview

We study the complexity of approximately counting the list H-colourings of a graph. List H-colourings generalise H-colourings in the same way that list colourings generalise proper vertex colourings. Fix an undirected graph H, which may have loops but not parallel edges. Given a graph G, an *H-colouring of G* is a homomorphism from G to H – that is, a mapping $\sigma : V(G) \to V(H)$ such that, for all $u, v \in V(G)$, $\{u, v\} \in E(G)$ implies $\{\sigma(u), \sigma(v)\} \in E(H)$. If we identify the vertex set $V(H)$ with a set $Q = \{1, 2, \ldots, q\}$ of "colours", then we can

* A full version of the paper containing detailed proofs is available at `http://arxiv.org/abs/1602.03985`. The numbering here matches the full version.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.
‡ This work was partially supported by the EPSRC grant EP/N004221/1.
§ This work was done in part while the authors were visiting the Simons Institute for the Theory of Computing.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 46; pp. 46:1–46:13

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

think of the mapping σ as specifying a colouring of the vertices G, and we can interpret the graph H as specifying the allowed colour adjacencies: adjacent vertices in G can be assigned colours i and j, if and only if vertices i and j are adjacent in H.

Now consider the graph G together with a collection of sets $\mathbf{S} = \{S_v \subseteq Q : v \in V(G)\}$ specifying allowed colours at each of the vertices. A *list H-colouring of* $(G, \mathbf{S})$ is an H-colouring σ of G satisfying $\sigma(v) \in S_v$, for all $v \in V$. In the literature, the set S_v is referred to as the "list" of allowed colours at vertex v, but there is no implied ordering on the elements of S_v – S_v is just a set of allowed colours.

Suppose that H is a *reflexive* graph (i.e., a graph in which each vertex has a loop). Feder and Hell [4] considered the complexity of determining whether a list H-colouring exists, given an input $(G, \mathbf{S})$. They showed that the problem is in FP if H is an interval graph, and that it is NP-complete, otherwise. Feder, Hell and Huang [5] studied the same problem in the case where H is *irreflexive* (i.e., H has no loops). They showed that the problem is in FP if H is a circular arc graph of clique covering number two (which is the same as being the complement of an interval bigraph [12]), and that it is NP-hard, otherwise. Finally, Feder, Hell and Huang [6] generalised this result to obtain a dichotomy for all H. They introduced a new class of graphs, called bi-arc graphs, and showed that the problem is in FP if H is a bi-arc graph, and NP-complete, otherwise.

We are concerned with the computational complexity of counting list H-colourings. Specifically we are interested in how the complexity of the following computational problem depends on H.

Name #List-H-Col.
Instance A graph G and a collection of colour sets $\mathbf{S} = \{S_v \subseteq Q : v \in V(G)\}$, where $Q = V(H)$.
Output The number of list H-colourings of $(G, \mathbf{S})$.

Note that it is of no importance whether we allow or disallow loops in G – a loop at vertex $v \in V(G)$ can be encoded within the set S_v – so we adopt the convention that G is loop-free. As in the case of the decision problem, H is a parameter of the problem – it does not form part of the problem instance. Sometimes we obtain sharper results by introducing an additional parameter Δ, which is an upper bound on the degrees of the vertices of G. Thus #List-H-Col(Δ) is the special case of #List-H-Col in which the graph G has degree at most Δ. Although #List-H-Col and #List-H-Col(Δ) are the main objects of study in this paper, we occasionally need to discuss the more basic versions of these problems without lists.

Name #H-Col.
Instance A graph G.
Output The number of H-colourings of G.

Once again, #H-Col(Δ) is the special case of #H-Col in which the degree of G is at most Δ. To illustrate the definitions, let K_2' be the first graph illustrated in Figure 1, consisting of two connected vertices with a loop on vertex 2. #K_2'-Col is the problem of counting independent sets in a graph since the vertices mapped to colour 1 by any homomorphism form an independent set. Let K_3 be the complete irreflexive graph on three vertices. Then #K_3-Col is the problem of counting the proper 3-colourings of a graph.

The computational complexity of computing exact solutions to #H-Col and #H-Col(Δ) was determined by Dyer and Greenhill [3]. Dyer and Greenhill showed that #H-Col is in FP if H is a complete reflexive graph or a complete bipartite irreflexive graph, and #H-Col

is #P-complete otherwise. Their dichotomy also extends to the bounded-degree setting. In particular, they showed that if H is not a complete reflexive graph or a complete bipartite irreflexive graph then there is an integer Δ_H such that, for all $\Delta \geq \Delta_H$, #H-Col(Δ) is #P-complete.

Since the polynomial-time cases in Dyer and Greenhill's dichotomy clearly remain solvable in polynomial-time in the presence of lists, their dichotomy for #H-Col carries over to #List-H-Col without change. In other words, there is no difference between the complexity of #H-Col and #List-H-Col as far as exact computation is concerned. However, this situation changes if we consider approximate counting, and this is the phenomenon that we explore in this paper.

With a view to reaching the statement of the main results as quickly as possible, we defer precise definitions of the relevant concepts to Section 2, and provide only indications here. From graph theory we import a couple of well studied hereditary graph classes, namely bipartite permutation graphs and proper interval graphs. These classes each have several equivalent characterisations, and we give two of these, namely, excluded subgraph and matrix characterisations, in Section 2. It is sometimes useful to restrict the definition of proper interval graphs to simple graphs. However, in this paper, as in [4], we consider reflexive proper interval graphs.

From complexity theory we need the definitions of a Fully Polynomial Randomised Approximation Scheme (FPRAS), of approximation-preserving (AP-) reducibility, and of the counting problems #SAT and #BIS. An FPRAS is a randomised algorithm that produces approximate solutions within specified relative error with high probability in polynomial time. An AP-reduction from problem Π to problem Π' is a randomised Turing reduction that yields close approximations to Π when provided with close approximations to Π'. It meshes with the definition of an FPRAS in the sense that the existence of an FPRAS for Π' implies the existence of an FPRAS for Π. The problem of counting satisfying assignments of a Boolean formula is denoted by #SAT. Every counting problem in #P is AP-reducible to #SAT, so #SAT is said to be complete for #P with respect to AP-reductions. It is known that there is no FPRAS for #SAT unless RP = NP. The problem of counting independent sets in a bipartite graph is denoted by #BIS. The problem #BIS appears to be of intermediate complexity: there is no known FPRAS for #BIS (and it is generally believed that none exists) but there is no known AP-reduction from #SAT to #BIS. Indeed, #BIS is complete with respect to AP-reductions for a complexity class #RHII$_1$ which is discussed further in the full version.

We say that a problem Π is #SAT-hard if there is an AP-reduction from #SAT to Π, that it is #SAT-easy if there is an AP-reduction from Π to #SAT, and that it is #SAT-equivalent if both are true. Note that all of these labels are about the difficulty of *approximately* solving Π, not about the difficulty of exactly solving it. Similarly, Π is said to be #BIS-hard if there is an AP-reduction from #BIS to Π, #BIS-easy if there is an AP-reduction from Π to #BIS, and #BIS-equivalent if there are both.

Our main result is a trichotomy for the complexity of approximating #List-H-Col.

▶ **Theorem 1.** *Suppose that H is a connected undirected graph (possibly with loops).*

(i) *If H is an irreflexive complete bipartite graph or a reflexive complete graph then* #List-H-Col *is in* FP*.*

(ii) *Otherwise, if H is an irreflexive bipartite permutation graph or a reflexive proper interval graph then* #List-H-Col *is* #BIS*-equivalent.*

(iii) *Otherwise,* #List-H-Col *is* #SAT*-equivalent.*

▶ Remarks. **1.** The assumption that H is connected is made without loss of generality,

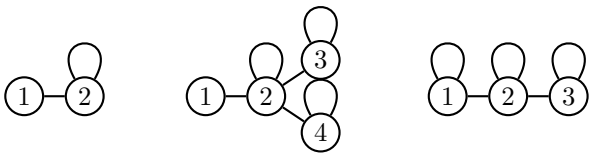


Figure 1 K_2', 2-wrench and P_3^*.

since the complexity of #List-H-Col is determined by the maximum complexity of #List-H'-Col over all connected components H' of H.

2. Part (ii) of Theorem 2 can be strengthened. For the graphs H covered by this part of the theorem, #List-H-Col is actually complete for the complexity class $\#\mathrm{RH}\Pi_1$. See the full version for a definition of $\#\mathrm{RH}\Pi_1$ and a proof of membership of #List-H-Col in $\#\mathrm{RH}\Pi_1$.

Theorem 1 also extends to the bounded-degree case.

▶ **Theorem 2.** *Suppose that H is a connected undirected graph (possibly with loops).*

(i) *If H is an irreflexive complete bipartite graph or a reflexive complete graph then, for all Δ, #List-H-Col(Δ) is in* FP.

(ii) *Otherwise, if H is an irreflexive bipartite permutation graph or a reflexive proper interval graph then, for all $\Delta \geq 6$, #List-H-Col(Δ) is* #BIS*-equivalent.*

(iii) *Otherwise, for all $\Delta \geq 6$, #List-H-Col(Δ) is* #SAT*-equivalent. Further, if H is reflexive or irreflexive, #List-H-Col(Δ) is* #SAT*-equivalent for $\Delta \geq 3$.*

▶ Remarks.

1. The condition $\Delta \geq 6$ is necessary for any hardness result that holds for all graphs H. In particular, there is a graph H that is not an irreflexive complete bipartite graph or a reflexive complete graph but for which #List-H-Col(Δ) has an FPTAS. An example is the graph $H = K_2'$ for which Weitz's self-avoiding walk algorithm [22] gives an FPTAS for #List-H-Col(Δ) for $\Delta \leq 5$.
2. In general, the lowest value of the degree bound Δ such that #List-H-Col(Δ) is computationally hard depends on the particular graph H.

Proof of Theorems 1 and 2. Part (i) is trivial. Part (ii) follows from Lemmas 13 and 14. Part (iii) follows from Lemmas 7, 9, and 11. ◀

The most obvious issue raised by our theorems is the computational complexity of approximately counting H-colourings (in the absence of lists). This question was extensively studied by Kelk [14] and others, and appears much harder to resolve, even when there are no degree bounds. It is known [7] that #H-Col is #BIS-hard for every connected undirected graph H that is neither an irreflexive bipartite permutation graph nor a reflexive proper interval graph. It is not known for which connected H the problem is #BIS-easy and for which it is #SAT-equivalent, and whether one or the other always holds. In fact, there are specific graphs H, two of them with as few as four vertices, for which the complexity of #H-Col is unresolved. It is far from clear that a trichotomy should be expected, and in fact there may exist an infinite sequences (H_t) of graphs for which #H_t-Col is reducible to #H_{t+1}-Col but not vice versa. Some partial results and speculations can be found in [14].

As we noted, #H-Col and #List-H-Col have the same complexity as regards exact computation. However, for approximate computation they are different, assuming (as is widely believed) that there is no AP-reduction from #SAT to #BIS. An example is provided

by the 2-wrench (see Figure 1). It is known [2, Theorem 21] that #2-WRENCH-COL is #BIS-equivalent, but we know from Theorem 1 that the list version #LIST-2-WRENCH-COL is #SAT-equivalent since the 2-wrench is neither irreflexive nor reflexive. One way to see that #LIST-2-WRENCH-COL is #SAT-equivalent is to note that the 2-wrench contains K_2' as an induced subgraph, and that this induced subgraph can be "extracted" using the list constraints $S_v = \{1, 2\}$, for all $v \in V(G)$. But #LIST-K_2'-COL is already known to be #SAT-equivalent [2, Theorem 1]. Indeed, systematic techniques for extracting hard induced subgraphs form the main theme of the paper. It is for this reason that the theory of hereditary graph classes comes into play, just as in [6].

Another recent research direction, at least in the unbounded-degree case, is towards weighted versions of list colouring. Here, the graph H is augmented by edge-weights, specifying for each pair of colours i, j, the cost of assigning i and j to adjacent vertices in G. The computational complexity of obtaining approximate solutions was studied by Chen, Dyer, Goldberg, Jerrum, Lu, McQuillan and Richerby [1], and by Goldberg and Jerrum [11]. There is a trichotomy for the case in which the input has no degree bound, but this is obtained in a context where *individual* spins at vertices are weighted and not just the interactions between *pairs* of adjacent spins. In this paper we have restricted the class of problems under consideration to ones having 0,1-weights on interactions, but at the same time we have restricted the problem instances to ones having 0,1-weights on individual spins. So we have a different tradeoff and the results from the references that we have just discussed do not carry across, even in the unbounded-degree setting. Indeed, towards the end of the paper, in Section 5, we give an example to show that Theorem 1 is not simply the restriction of earlier results to 0,1-interactions (not merely because the proofs differ, but, in a stronger sense, because the results themselves are different).

Two things are appealing about our theorems. First, unlike the weighted classification theorems [1], here the truth is pleasingly simple. The trichotomies for #LIST-H-COL and #LIST-H-COL(Δ) have a simple, natural formulation in terms of hereditary graph classes. Second, the proofs of the theorems are largely self-contained. The proofs do not rely on earlier works such as [1], which require multimorphisms and other deep results from universal algebra. The proof of Theorem 1 is self-contained apart from some very elementary and well-known starting points, which are collected together in Lemma 6. The proof of Theorem 2 is similarly self-contained, though it additionally relies on recent results [18, 8] about approximating the partition function of the anti-ferromagnetic Ising model on bounded degree graphs (these are also contained in Lemma 6).

2 Complexity- and graph-theoretic preliminaries

As the complexity of computing exact solutions of #LIST-H-COL is well understood, we focus on the complexity of computing approximations. The framework for this has already been explained in many papers, so we provide an informal description only here and direct the reader to Dyer, Goldberg, Greenhill and Jerrum [2] for precise definitions.

The standard notion of efficient approximation algorithm is that of a *Fully Polynomial Randomised Approximation Scheme* (or FPRAS). This is a randomised algorithm that is required to produce a solution within relative error specified by a tolerance $\varepsilon > 0$, in time polynomial in the instance size and ε^{-1}. Evidence for the non-existence of an FPRAS for a problem Π can be obtained through *Approximation-Preserving* (or AP-) *reductions*. These are randomised polynomial-time Turing reductions that preserve (closely enough) the error tolerance. The set of problems that have an FPRAS is closed under AP-reducibility.

Every problem in #P is AP-reducible to #SAT, so #SAT is complete for #P with respect to AP-reductions. The same is true of the counting version of any NP-complete decision problem. It is known that these problems do not have an FPRAS unless RP = NP. On the other hand, using the bisection technique of Valiant and Vazirani [20, Corollary 3.6], we know that #SAT can be approximated (in the FPRAS sense) by a polynomial-time probabilistic Turing machine equipped with an oracle for the decision problem SAT.

In the statement and proofs of our theorems we refer to two hereditary graph classes. A class of undirected graphs is said to be *hereditary* if it is closed under taking induced subgraphs. The classes of bipartite permutation graphs and proper interval graphs have been widely studied and many equivalent characterisations of them are known. We are concerned with the excluded subgraph and matrix characterisations.

A graph is a *bipartite permutation graph* if and only if it contains none of the following as an induced subgraph: X_3, X_2, T_2 or a cycle C_ℓ of length ℓ not equal to four. (Refer to Figure 2 for specifications of X_3, X_2 and T_2.) This characterisation was noted by Köhler [15], who observed that it follows from an excluded subgraph characterisation of Gallai [9, 10]. The argument is given by Hell and Huang [12], in the proof of their Theorem 3.4, in particular parts (iv) and (vi).

A graph is a *proper interval graph* if and only if it contains none of the following as an induced subgraph: the claw, the net, S_3 or a cycle C_ℓ of length ℓ at least four. (Refer to Figure 3 for specifications of the claw, the net and S_3.) This characterisation is due to Wegner [21] and Roberts [17], and is stated is by Jackowski [13] as his Theorem 1.4, specifically the equivalence of (i) and (iii). In this context, note that a *chordal* graph is one that contains no induced cycles of length other than three.

The two graph classes also have matrix characterisations. Say that a 0,1-matrix $A = (A_{i,j} : 1 \leq i \leq n, 1 \leq j \leq m)$ has *staircase form* if the 1s in each row are contiguous and the following condition is satisfied: letting $\alpha_i = \min\{j : A_{i,j} = 1\}$ and $\beta_i = \max\{j : A_{i,j} = 1\}$, we require that the sequences (α_i) and (β_i) are non-decreasing. It is automatic that the columns share the contiguity and monotonicity properties, so the property of having staircase form is in fact invariant under matrix transposition.

A graph is a bipartite permutation graph if the rows and columns of its biadjacency matrix can be (independently) permuted so that the resulting biadjacency matrix has staircase form. This characterisation is presented by Spinrad, Brandstädt and Stewart [19], specifically the equivalence of (i) and (ii) in their Theorem 1.

A graph is a proper interval graph if the rows and columns of its adjacency matrix can be (simultaneously) permuted so that the resulting adjacency matrix has staircase form. This fact comes directly from the characterisation of proper interval graphs that gives the class its name, namely, that they are graphs which have an interval intersection model in which no interval is a proper subset of another. The ordering of intervals by left endpoint (which is the same as the ordering by right endpoint) gives the required permutation of rows and columns.

As we mentioned in Section 1, an appealing feature of our theorems is that our proofs are largely self-contained. The only pre-requisites for the proof are complexity results classifying some very well-known approximation problems. These are collected in Lemma 6, which is proved in the full version. For this, we will use the graph K_2' defined in Section 1 (see Figure 1), the path P_4 of length three (with four vertices) and the problem #1P1NSAT of counting the satisfying assignments of a CNF formula in which each clause has at most one negated literal and at most one unnegated literal. We will also use the following definition.

▶ **Definition 4.** Let $0 < \lambda < 1$ be a rational number and let Δ be a positive integer. Define
Name AntiFerroIsing$_\lambda(\Delta)$.
Instance *A graph G of maximum degree at most Δ.*
Output *The partition function of the antiferromagnetic Ising model with parameter λ evaluated on instance G, i.e., $Z_\lambda(G) = \sum_{\sigma:V\to\{\pm1\}} \prod_{\{u,v\}\in E(G)} \lambda^{\delta(\sigma(u),\sigma(v))}$, where $\delta(i,j)$ is 1 if $i = j$ and 0 otherwise.*

▶ **Lemma 6.** *The following problems are* #SAT*-equivalent: #K_2'-Col(Δ) for any $\Delta \geq 6$, and* AntiFerroIsing$_\lambda(\Delta)$ *for any $\Delta \geq 3$ and $0 < \lambda < (\Delta-2)/\Delta$. The following problems are* #BIS*-equivalent: #P_4-Col(Δ) for $\Delta \geq 6$ and* #1p1nSat.

3 #SAT-equivalence

The aim of this section is to establish the #SAT-equivalence parts of Theorems 1 and 2.

▶ **Lemma 7.** *Suppose that H is a connected undirected graph. If H is neither reflexive nor irreflexive then, for all $\Delta \geq 6$,* #List-H-Col(Δ) *is* #SAT*-equivalent. Hence,* #List-H-Col *is* #SAT*-equivalent.*

Proof. Let $\Delta \geq 6$. Since H is connected, it must contain K_2' as an induced subgraph. So #K_2'-Col(Δ) is AP-reducible to #List-H-Col(Δ). By Lemma 6, #K_2'-Col(Δ) is #SAT-equivalent. ◀

The gadgets that we use in our reductions in the upcoming lemmas are of a particularly simple kind, namely paths.[1] Let the vertex set of the L-vertex path be $\{1, 2, \ldots, L\}$, where the vertices are numbered according to their position on the path. The end vertices 1 and L are *terminals*, which make connections with the rest of the construction. For each vertex $1 \leq k \leq L$ there is a set of allowed colours S_k. We can describe a gadget by specifying L and specifying the sets $(S_1, S_2, \ldots, S_L)$. In our application, each set S_i has cardinality 2, and $S_1 = S_L$.

Fix a connected graph H (note that H may have loops). Our strategy for proving that #List-H-Col(Δ) is #SAT-equivalent is to find a gadget $(\{i_1, j_1\}, \{i_2, j_2\}, \ldots, \{i_L, j_L\})$ such that

(i) the sequence $(i_1, \ldots, i_L)$ is a path in H, and likewise $(j_1, \ldots, j_L)$;
(ii) it is never the case that both $\{i_k, j_{k+1}\} \in E(H)$ and $\{j_k, i_{k+1}\} \in E(H)$; and
(iii) $i_1 = j_L$ and $j_1 = i_L$.

If we achieve these conditions then, as we shall see, the colours at the terminals will be negatively correlated, and from there we will be able to encode instances of AntiFerroIsing$_\lambda(\Delta)$ for some integer $\Delta \geq 3$ and $\lambda \in (0, \frac{\Delta-2}{\Delta})$, and this is #SAT-equivalent (Lemma 6). Note that although the ordering of elements within the sets S_i is irrelevant to the workings of the gadget, we write the pairs in a specific order to bring out the path structure that we have just described.

Fix H and let $A = A_H$ be the adjacency matrix of H. Denote by $A_{(i,j),(i',j')}$ the 2×2 submatrix of A indexed by rows i and j and columns i' and j'. We regard the indices in the notation $A_{(i,j),(i',j')}$ as ordered; thus the first row of this 2×2 matrix comes from row i of A and the second from row j.

[1] We were also able to make use of path gadgets in [11], though, as noted (see Section 1) the results unfortunately do not carry over to our setting. Here the use of structural graph theory makes the discovery of such gadgets pleasingly straightforward.

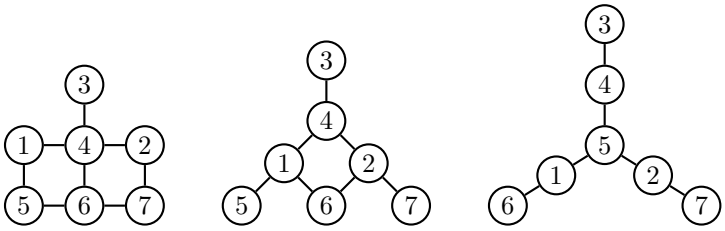


Figure 2 X_3, X_2 and T_2.

Given a gadget, i.e., sequence $(\{i_1, j_1\}, \{i_2, j_2\}, \ldots, \{i_L, j_L\})$, consider the product of 2×2 submatrices of A:

$$D' = A_{(i_1,j_1),(i_2,j_2)} A_{(i_2,j_2),(i_3,j_3)} \cdots A_{(i_{L-1},j_{L-1}),(i_L,j_L)}. \tag{1}$$

If conditions (i)–(iii) for gadget construction are satisfied then each of the 2×2 matrices in the product has 1s on the diagonal; also, all of them have at least one off-diagonal entry that is 0. Thus, each matrix has determinant 1, from which it follows that $\det D' = 1$.

Now consider the matrix D that is obtained by swapping the two columns of D'. This swap rectifies the "twist" that occurs in the passage from (i_1, j_1) to $(i_L, j_L) = (j_1, i_1)$, but it also flips the sign of the determinant, leaving $\det D = -1$. Let $r = i_1 = j_L$ and $s = j_1 = i_L$. The matrix D can be interpreted as giving the number of list H-colourings of the gadget when the k'th vertex of the gadget (for $k \in \{1, \ldots, L\}$ is assigned the list $\{i_k, j_k\}$, so the terminals are restricted to colours in $\{r, s\}$. Thus the entry in the first row and column of D is the number of colourings with both terminals receiving colour r, the entry in the first row and second column is the number of colourings with terminal 1 receiving colour r and terminal L receiving colour s, the entry in the second row and first column is the number of colourings with terminal 1 receiving colour s and terminal L receiving colour r and finally the entry in the second row and second column is the number of colourings with both terminals receiving colour s. We call $D = D(\Gamma)$ the *interaction matrix* associated with the gadget Γ. Since $\det D < 0$ the gadget provides a negative correlation between the colours at the terminals, which, as we will see, will allow a reduction from $\text{AntiFerroIsing}_\lambda(\Delta)$.

In the full version, Lemma 8 first applies the technique to get the #SAT-equivalences in the unbounded-degree case. For these arguments, we intentionally keep the construction of the gadgets as simple as possible. While this would also lead to a value of Δ such that #List-H-Col(Δ) is #SAT-equivalent, the value of Δ would be much larger than 6. Nevertheless, we are able to refine the constructions to obtain the bounded-degree results of Theorem 2. Lemma 9 here combines Lemmas 8 and 9 of the full version and illustrates the key ideas.

▶ **Lemma 9.** *Suppose that H is a connected undirected graph. If H is irreflexive but it is not a bipartite permutation graph, then for all $\Delta \geq 3$, #List-H-Col(Δ) is #SAT-equivalent (so #List-H-Col is also #SAT-equivalent).*

Proof (One Case). Graphs that are not bipartite permutation graphs contain one of the following as an induced subgraph: X_3, X_2, T_2, or a cycle of length other than 4. (Refer to Figure 2.) Here we present the case X_3. The remaining cases are similar and can be found in the full version.

We first show that the unbounded problem #List-X_3-Col is #SAT-equivalent. The gadget in this case is $\Gamma = \big(\{1, 2\}, \{4, 7\}, \{3, 6\}, \{4, 5\}, \{2, 1\}\big)$. The conditions (i)–(iii) for

gadget construction are easy to check. Explicit calculation using (1) yields

$$D' = A_{(1,2),(4,7)}A_{(4,7),(3,6)}A_{(3,6),(4,5)}A_{(4,5),(2,1)} = \left(\begin{smallmatrix}1&0\\1&1\end{smallmatrix}\right)\left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right)\left(\begin{smallmatrix}1&0\\1&1\end{smallmatrix}\right)\left(\begin{smallmatrix}1&1\\0&1\end{smallmatrix}\right) = \left(\begin{smallmatrix}2&3\\3&5\end{smallmatrix}\right).$$

Swapping the columns of D' yields the interaction matrix $D = \left(\begin{smallmatrix}3&2\\5&3\end{smallmatrix}\right)$. As we explained earlier, $\det D = -1$. Obtaining a matrix D with negative determinant is moving in the right direction, but in order to encode antiferromagnetic Ising we ideally want the matrix $D = (D_{i,j})$ to also satisfy $D_{1,1} = D_{2,2}$ and $D_{1,2} = D_{2,1}$. Observe that the graph X_3 has an automorphism of order two, $\pi = (1,2)(5,7)$, that transposes vertices 1 and 2, which are the terminals of the gadget Γ. Consider the gadget obtained from Γ by letting π act on the colour sets, namely

$$\begin{aligned}\Gamma^\pi &= \big(\{\pi(1),\pi(2)\},\{\pi(4),\pi(7)\},\{\pi(3),\pi(6)\},\{\pi(4),\pi(5)\},\{\pi(2),\pi(1)\}\big)\\ &= \big(\{2,1\},\{4,5\},\{3,6\},\{4,7\},\{1,2\}\big),\end{aligned}$$

The interaction matrix $D^\pi = \left(\begin{smallmatrix}3&5\\2&3\end{smallmatrix}\right)$ corresponding to Γ^π is the same as D, except that the rows and columns are swapped. Placing Γ and Γ^π in parallel, identifying the terminals, yields a composite gadget Γ^* whose interaction matrix is $D^* = \left(\begin{smallmatrix}D_{1,1}D_{2,2} & D_{1,2}D_{2,1}\\ D_{2,1}D_{1,2} & D_{2,2}D_{1,1}\end{smallmatrix}\right) = \left(\begin{smallmatrix}9&10\\10&9\end{smallmatrix}\right)$. Note that the gadget Γ^* has maximum degree 2 (this observation will be important for the bounded-degree case). Also, $\det D^* = D_{1,1}^2D_{2,2}^2 - D_{1,2}^2D_{2,1}^2 = (D_{1,1}D_{2,2}+D_{1,2}D_{2,1})\det D < 0$. So we have an AP-reduction from AntiFerroIsing$_\lambda$ with $\lambda = D_{1,1}D_{2,2}/(D_{1,2}D_{2,1})$ to #List-H-Col: given an instance G of AntiFerroIsing$_\lambda$, simply replace each edge $\{u,v\}$ of G with a copy of the gadget Γ^*, identifying the two terminals of Γ^* with the vertices u and v, respectively. (Since Γ^* is symmetric, it does not matter which is u and which is v.) The problem AntiFerroIsing$_\lambda$ is #SAT-equivalent by Lemma 6. So for this case ($H = X_3$), we have $\lambda = \frac{9}{10}$.

We next show that, for $\Delta \geq 3$, #List-X_3-Col(Δ) is #SAT-equivalent. The smallest Δ such that AntiFerroIsing$_{\frac{9}{10}}(\Delta)$ is #SAT-equivalent is $\Delta = 21$, so the argument above would only give that #List-X_3-Col(Δ) is #SAT-equivalent for $\Delta \geq 21$ (in fact, $\Delta \geq 2\cdot 21 = 42$, since the terminals of Γ^* have degree 2). To improve this, we will implement thickenings of the gadget Γ^* using carefully chosen list colourings to keep the degree of the gadget small. More precisely, for integer $t \geq 0$, we will construct inductively gadgets Γ_t^* such that:

- **(i)** The allowed colours of the terminals of Γ_t^* will be $\{1,2\}$ for odd t and $\{5,7\}$ for even t.
- **(ii)** The two terminals of Γ_t^* will each have degree 1, and all other vertices of Γ_t^* will have degree at most 3.
- **(iii)** The interaction matrix of Γ_t^* will be $D_t^* = \left(\begin{smallmatrix}9^{2^t} & 10^{2^t}\\ 10^{2^t} & 9^{2^t}\end{smallmatrix}\right)$.

By taking t sufficiently large, the reduction above, using Γ_t^* as gadget instead of Γ^*, yields that #List-X_3-Col(Δ) is #SAT-equivalent for $\Delta \geq 3$. It remains to build the gadgets Γ_t^*.

Γ_0^* is obtained from Γ^* by connecting each terminal of Γ^* to a new vertex whose allowed set of colours is $\{5,7\}$ (recall that the allowed colours of the terminals of Γ^* are in $\{1,2\}$). The terminals of Γ_0^* are the two new vertices. Γ_0^* clearly satisfies properties (i) and (ii) for $t = 0$. To find the interaction matrix of Γ_0^*, note that colour 1 is adjacent to colour 5 in X_3 but not to colour 7. Similarly, colour 2 is adjacent to colour 7 in X_3 but not to colour 5. Thus, $D(\Gamma_0^*) = \left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)\left(\begin{smallmatrix}9&10\\10&9\end{smallmatrix}\right)\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right) = D_0^*$, proving that Γ_0^* satisfies all properties (i)–(iii), as desired.

To construct Γ_{t+1}^* from Γ_t^*, take two copies of Γ_t^*, place them in parallel, identifying their terminals in the natural way. Now, analogously to the construction of Γ_0^*, connect each (doubled-up) terminal of Γ_t^* to a new vertex whose allowed set of colours is $\{1,2\}$ if t is even and $\{5,7\}$ if t is odd. The final graph is the gadget Γ_{t+1}^* and the new vertices

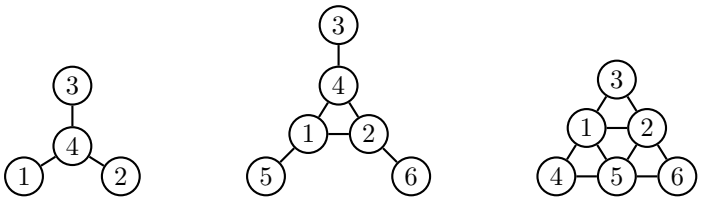


Figure 3 The claw, the net and S_3.

introduced in the second step of the construction are its terminals. It is clear that Γ^*_{t+1} satisfies property (i) and, using the fact that Γ^*_t satisfies property (ii), we have that Γ^*_{t+1} satisfies property (ii) as well. Arguing analogously as for Γ^*_0, the interaction matrix of Γ^*_{t+1} is given by $D(\Gamma^*_{t+1}) = \left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)\begin{pmatrix}(9^{2^t})^2 & (10^{2^t})^2\\ (10^{2^t})^2 & (9^{2^t})^2\end{pmatrix}\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right) = D^*_{t+1}$, where the squared entries account for the 2-thickening of the gadget Γ^*_t. This proves that Γ^*_{t+1} has property (iii), concluding the proof for $H = X_3$. The other cases are similar, and are given in the full version (where more of the general principles are explained). ◀

The remaining cases of parts (iii) of Theorems 1 and 2 are covered by Lemmas 10 and 11 of the full version, which we combine here.

▶ **Lemma 11.** *Suppose that H is a connected undirected graph. If H is a reflexive graph that is not a proper interval graph, then, for $\Delta \geq 3$, #List-H-Col(Δ) is #SAT-equivalent (so #List-H-Col is #SAT-equivalent).*

Proof Sketch. The line of argument is similar to those used in Lemma 9. Graphs that are not proper interval graphs contain one of the following as an induced subgraph: the claw, the net, S_3, or a cycle of length at least four. (Refer to Figure 3 but note that loops are omitted.) We show that #List-H-Col is #SAT-equivalent when H is any of these (and the bounded-degree analogue). The details can be found in the full version. ◀

4 #BIS-equivalence

We now deal with the #BIS-equivalent cases in Theorems 1 and 2.

▶ **Lemma 13.** *Suppose that H is a connected undirected graph. If H is not a reflexive complete graph or an irreflexive complete bipartite graph then, for all $\Delta \geq 6$, #List-H-Col(Δ) is #BIS-hard. Hence, #List-H-Col is #BIS-hard.*

Proof Sketch. In the full version, we show that any graph covered by the lemma contains one of the following as an induced subgraph: K'_2, P^*_3, P_4 or an odd cycle. These are (at least) #BIS-hard: by Lemma 6 for K'_2 and P_4, by Lemma 12 of the full version for P^*_3, and by Lemma 9 for an odd cycle. ◀

▶ **Lemma 14.** *Suppose that H is a connected undirected graph. If H is an irreflexive bipartite permutation graph or a reflexive proper interval graph, then #List-H-Col is #BIS-easy.*

Proof Sketch. The reduction is done in a more general weighted setting by Chen, Dyer, Goldberg, Jerrum, Lu, McQuillan and Richerby [1]: see the proofs of Lemmas 45 and 46 of that article. However, in the current context, we can simplify the reduction significantly (eliminating the need for multimorphisms and other concepts from universal algebra), and we can also extract (see the full paper) the slightly stronger statement that #List-H-Col

is in #RHΠ$_1$. The target problem for our reduction is #1P1NSAT which is #BIS-equivalent by Lemma 6.

We will treat the case where H is an irreflexive bipartite permutation graph. The other case is similar (as explained in the full version). Without loss of generality, suppose that H is connected and that its biadjacency matrix B has q_1 rows and q_2 columns and is in staircase form. Let A be the adjacency matrix $\left(\begin{smallmatrix} B & 0 \\ 0 & B^T \end{smallmatrix}\right)$, which is formally defined as follows.

$$A_{i,j} = \begin{cases} B_{i,j}, & \text{if } 1 \le i \le q_1,\ 1 \le j \le q_2 \\ B_{j-q_2,i-q_1}, & \text{if } q_1+1 \le i \le q_1+q_2,\ q_2+1 \le j \le q_2+q_1 \\ 0, & \text{otherwise.} \end{cases}$$

Let $q = q_1 + q_2$. For each $i \in \{1,\dots,q\}$, let $\alpha_i = \min\{j : A_{i,j} = 1\}$ and let $\beta_i = \max\{j : A_{i,j} = 1\}$. Since B is in staircase form, so is A, so the sequences (α_i) and (β_i) are non-decreasing. Let $r_1,\dots,r_q$ be the colours associated with the rows of A and $c_1,\dots,c_q$ be the colours associated with the columns of A, in order. Note that $\{r_1,\dots,r_q\}$ and $\{c_1,\dots,c_q\}$ are different permutations of the vertices of H,

Suppose that $(G,\mathbf{S})$ is an instance of #LIST-H-COL. Assume without loss of generality that G is bipartite. Otherwise, it has no H-colourings. Let $V_1(G) \cup V_2(G)$ be the bipartition of $V(G)$. We will construct an instance Ψ of #1P1NSAT such that the number of satisfying assignments to Ψ is equal to the number of list H-colourings of $(G,\mathbf{S})$.

The variable set of Ψ is $\mathbf{x} = \{x_i^u : u \in V(G) \text{ and } 0 \le i \le q\}$. For each vertex $u \in V(G)$ we introduce the clauses (x_0^u) and $(\neg x_q^u)$. Also, for each $j \in \{1,\dots,q\}$ we introduce the clause $\mathrm{IMP}(x_j^u, x_{j-1}^u)$. Denote by $\Psi_V(\mathbf{x})$ the formula obtained by taking the conjunction of all these clauses.

We will interpret the assignment to the variables in $\mathbf{x}$ as an assignment σ of colours to the vertices of G according to the following rule. If $u \in V_1(G)$ then $x_i^u = 1$ if and only if $\sigma(u) = r_j$ for some $j > i$. If $u \in V_2(G)$ then $x_i^u = 1$ if and only if $\sigma(u) = c_j$ for some $j > i$. Note that there is a one-to-one correspondence between assignments to $\mathbf{x}$ that satisfy the clauses in $\Psi_V(\mathbf{x})$ and assignments σ of colours to the vertices of G.

We now introduce further clauses to enforce the constraint on colours received by adjacent vertices. For each edge $\{u,v\} \in E(G)$ with $u \in V_1(G)$ and $v \in V_2(G)$, and for each $i \in \{1,\dots,q\}$, we add the clauses $\mathrm{IMP}(x_{i-1}^u, x_{\alpha_i-1}^v)$ and $\mathrm{IMP}(x_{\beta_i}^v, x_i^u)$. Denote by $\Psi_E(\mathbf{x})$ the formula obtained by taking the conjunction of all of these clauses.

We next argue that there is a bijection between H-colourings of G and satisfying assignments to $\Psi_V(\mathbf{x}) \wedge \Psi_E(\mathbf{x})$. In one direction, suppose σ is an H-colouring of G. We wish to show that all clauses in $\Psi_E(\mathbf{x})$ are satisfied. Consider an edge $\{u,v\} \in E(G)$ with $u \in V_1(G)$ and $v \in V_2(G)$ and the corresponding clause $\mathrm{IMP}(x_{i-1}^u, x_{\alpha_i-1}^v)$. The clause is satisfied unless $x_{i-1}^u = 1$, so suppose $x_{i-1}^u = 1$. Then by the interpretation of assignments, $\sigma(u) = r_j$ for some $j \ge i$. Since σ is an H-colouring, this implies that $\sigma(v) = c_k$ for some $k \ge \alpha_i$. But by the interpretation of assignments, this means that $x_{\alpha_i-1}^v = 1$, so the clause is satisfied. The argument for the other clause $\mathrm{IMP}(x_{\beta_i}^v, x_i^u)$ coresponding to the edge $\{u,v\}$ is similar – see the full version.

In the other direction, suppose $\Psi_V(\mathbf{x}) \wedge \Psi_E(\mathbf{x})$ is satisfied. Consider an edge $\{u,v\} \in E(G)$ with $u \in V_1(G)$ and $v \in V_2(G)$ and suppose that $\sigma(u) = r_i$. In the corresponding assignment $x_{i-1}^u = 1$ so by the clause $\mathrm{IMP}(x_{i-1}^u, x_{\alpha_i-1}^v)$ we have $x_{\alpha_i-1}^v = 1$ so $\sigma(v) = c_k$ for some $k \ge \alpha_i$. In the corresponding assignment $x_i^u = 0$ so by the clause $\mathrm{IMP}(x_{\beta_i}^v, x_i^u)$, $x_{\beta_i}^v = 0$, so $\sigma(v) = c_k$ for some $k \le \beta_i$. We conclude that the colours $\sigma(u)$ and $\sigma(v)$ are adjacent in H. This holds for every edge, so σ is an H-colouring of G.

Finally, we add clauses to deal with lists. A colour assignment $\sigma(u) = r_i$ with $u \in V_1(G)$ is uniquely characterised by $x^u_{i-1} = 1$ and $x^u_i = 0$. So we can eliminate the possibility of $\sigma(u) = r_i$ by introducing the clause $\mathrm{IMP}(x^u_{i-1}, x^u_i)$. A similar clause will forbid a vertex $v \in V_2(G)$ to receive colour c_j. Let $\Psi_L(\mathbf{x})$ be the conjunction of all such clauses, arising from the lists in $\mathbf{S}$. Let $\Psi(\mathbf{x}) = \Psi_V(\mathbf{x}) \wedge \Psi_E(\mathbf{x}) \wedge \Psi_L(\mathbf{x})$. Then the list H-colourings of $(G, \mathbf{S})$ are in bijection with the satisfying assignments to $\Psi(\mathbf{x})$, as required ◀

5 A counterexample

The situation that we have studied in this paper is characterised by having hard interactions between pairs of adjacent spins (a pair is either allowed or it is disallowed) and hard constraints on individual spins (again, a spin is either allowed at a particular vertex or it is disallowed). Our results apply both in the degree-bounded case and in the unbounded-degree case. In the unbounded case, earlier work treated the situation with weighted interactions and weighted spins. The characterisations derived in these weighted scenarios (see, e.g. [11, Thm 1]) have a similar feel to the trichotomy that we have presented in Theorem 1. We may wonder whether, in the unbounded case, at least, there is a common generalisation. That is, in the unbounded case, does the trichotomy of [11] survive if weights on spins are replaced by lists? The answer is no. There are examples of weighted spin systems with just $q = 2$ spins whose partition function is #SAT-hard to approximate with vertex weights but efficiently approximable (in the sense that there is an FPRAS) with lists instead of weights.

Here is one such example. Following Li, Lu and Yin [16], define the interaction matrix $A = (a_{ij} : 0 \le i, j \le 1)$ by $A = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 2 \end{smallmatrix}\right)$, and the partition function associated with an instance G by $Z_A(G) = \sum_{\sigma : V(G) \to \{0,1\}} \prod_{\{u,v\} \in E(G)} a_{\sigma(u),\sigma(v)}$. This is the partition function of a variant of the independent set model, which instead of defining the interaction between spin 1 and itself (two vertices that are out of the independent set) to be 1, defines this interaction weight to be 2.

Li, Lu and Yin [16, Theorem 21] show that Weitz's self-avoiding walk algorithm [22] gives an FPTAS for $Z_A(G)$. Also, Weitz's correlation decay algorithm [22] can accommodate lists. Indeed, the construction of the self-avoiding walk tree relies on being able to "pin" colours at individual vertices. So the partition function remains easy to approximate (in the sense that there is an FPTAS) even in the presence of lists. In contrast, the approximation problem becomes #SAT-hard if arbitrary weights are allowed. Indeed, by weighting spin 0 at each vertex $u \in V(G)$ by $2^{d(u)}$, where $d(u)$ is the degree of u, we recover the usual independent set partition function, which is #SAT-equivalent (Lemma 6). (The same fact can be read off from general results in many papers, including [11, Thm 1].) Thus, even in the unbounded case, the dichotomies presented in [11, Thm 1] and [1, Thm 6] do not hold with lists in place of weights. So even in the unbounded-degree case, it was necessary to explicitly analyse list homomorphisms in order to derive precise characterisations quantifying the problem of approximately counting these.

References

1 Xi Chen, Martin Dyer, Leslie Ann Goldberg, Mark Jerrum, Pinyan Lu, Colin McQuillan, and David Richerby. The complexity of approximating conservative counting CSPs. *J. Comput. System Sci.*, 81(1):311–329, 2015.

2 Martin Dyer, Ann Leslie Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2003.

3 Martin Dyer and Catherine Greenhill. The complexity of counting graph homomorphisms. *Random Structures Algorithms*, 17(3-4):260–289, 2000.

4 Tomas Feder and Pavol Hell. List homomorphisms to reflexive graphs. *Journal of Combinatorial Theory, Series B*, 72(2):236 – 250, 1998.

5 Tomás Feder, Pavol Hell, and Jing Huang. List homomorphisms and circular arc graphs. *Combinatorica*, 19(4):487–505, 1999.

6 Tomas Feder, Pavol Hell, and Jing Huang. Bi-arc graphs and the complexity of list homomorphisms. *J. Graph Theory*, 42(1):61–80, 2003.

7 Andreas Galanis, Leslie Ann Goldberg, and Mark Jerrum. Approximately counting H-colourings is #BIS-hard. *SIAM Journal on Computing (to appear)*, 2016.

8 Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic Ising and hard-core models. *Combinatorics, Probability and Computing (to appear)*, 2016.

9 T. Gallai. Transitiv orientierbare Graphen. *Acta Math. Acad. Sci. Hungar*, 18:25–66, 1967.

10 Tibor Gallai. A translation of T. Gallai's paper: "Transitiv orientierbare Graphen" [Acta Math. Acad. Sci. Hungar. **18** (1967), 25–66; MR0221974 (36 #5026)]. In *Perfect graphs*, Wiley-Intersci. Ser. Discrete Math. Optim., pages 25–66. Wiley, Chichester, 2001. Translated from the German and with a foreword by Frédéric Maffray and Myriam Preissmann.

11 Leslie Ann Goldberg and Mark Jerrum. A complexity classification of spin systems with an external field. *Proceedings of the National Academy of Sciences*, 112(43):13161–13166, 2015.

12 Pavol Hell and Jing Huang. Interval bigraphs and circular arc graphs. *J. Graph Theory*, 46(4):313–327, 2004.

13 Zygmunt Jackowski. A new characterization of proper interval graphs. *Discrete Math.*, 105(1-3):103–109, 1992.

14 Steven Kelk. *On the relative complexity of approximately counting H-colourings.* PhD thesis, Warwick University, 2003.

15 Ekkehard G. Koehler. *Graphs without asteroidal triples.* PhD thesis, Technische Universität Berlin, 1999.

16 Liang Li, Pinyan Lu, and Yitong Yin. Approximate counting via correlation decay in spin systems. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 922–940, 2012.

17 F.S Roberts. *Representations of indifference relations.* PhD thesis, Stanford University, Stanford, CA, 1968.

18 Allan Sly. Computational transition at the uniqueness threshold. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 287–296, 2010.

19 Jeremy Spinrad, Andreas Brandstädt, and Lorna Stewart. Bipartite permutation graphs. *Discrete Appl. Math.*, 18(3):279–292, 1987.

20 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.

21 Gerd Wegner. *Eigenschaften der Nerven homologisch-einfacher Familien im $\mathbb{R}^n$.* PhD thesis, Universität Göttingen, Göttingen, Germany, 1967.

22 Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, pages 140–149, 2006.

Parity Separation: A Scientifically Proven Method for Permanent Weight Loss*

Radu Curticapean

Simons Institute for the Theory of Computing, UC Berkeley, USA; and
Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
radu.curticapean@gmail.com

Abstract

Given an edge-weighted graph G, let PerfMatch(G) denote the weighted sum over all perfect matchings M in G, weighting each matching M by the product of weights of edges in M. If G is unweighted, this plainly counts the perfect matchings of G.

In this paper, we introduce parity separation, a new method for reducing PerfMatch to unweighted instances: For graphs G with edge-weights 1 and -1, we construct two unweighted graphs G1 and G2 such that PerfMatch(G) = PerfMatch(G1) − PerfMatch(G2). This yields a novel weight removal technique for counting perfect matchings, in addition to those known from classical #P-hardness proofs. Our technique is based upon the Holant framework and matchgates. We derive the following applications:

Firstly, an alternative #P-completeness proof for counting unweighted perfect matchings.

Secondly, C=P-completeness for deciding whether two given unweighted graphs have the same number of perfect matchings. To the best of our knowledge, this is the first C=P-completeness result for the "equality-testing version" of any natural counting problem that is not already #P-hard under parsimonious reductions.

Thirdly, an alternative tight lower bound for counting unweighted perfect matchings under the counting exponential-time hypothesis #ETH.

1998 ACM Subject Classification G.2.1 Combinatorics, F.1.3 Complexity Measures and Classes

Keywords and phrases perfect matchings, counting complexity, structural complexity, exponential-time hypothesis

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.47

1 Introduction

The problem of counting perfect matchings has played a central role in counting complexity since Valiant [27] introduced the class #P and established #P-completeness of counting perfect matchings in unweighted bipartite graphs. This problem was previously already considered in statistical physics [26, 21, 22] and Valiant's computational hardness result explains the lack of progress encountered in this area for finding efficient algorithms for counting perfect matchings.

As complexity theorists, we can appreciate this seminal #P-completeness result from another perspective: The problem of counting perfect matchings in *unweighted* graphs presented the first example of a natural hard counting problem with an easy decision version, since Edmond's classical algorithm [14] allows to decide in polynomial time whether a graph

* Part of this work was carried out while the author was a PhD student at the Department of Computer Science at Saarland University, Saarbrücken, Germany. It also appears in his PhD thesis [11] and [10].

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 47; pp. 47:1–47:14


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

contains *at least one* perfect matching. This showed exemplarily that the complexity-theoretic study of counting problems amounts to more than merely checking whether NP-hardness proofs for decision problems carry over to their counting versions.

For instance, a fundamental peculiarity of counting problems that is not shared by decision problems are *cancellations*: In (weighted) counting problems, witness structures may cancel each other out, and this can have strong effects on the complexity of the problem. The most prominent example of this phenomenon might be the situation of the *determinant* and the *permanent*, both summing over the same permutations, however with different weights. This results in the permanent being #P-complete by Valiant's result, whereas the determinant can be computed in polynomial time. The *accidental* and *holographic* algorithms introduced by Valiant [28, 29] provide examples for further and more unexpected cancellations that render counting problems easy.

However, cancellations are also crucial for negative results: In many #P-hardness proofs, such as [4, 2, 3], we first define an intermediate variant of the target problem on weights ± 1. Examples for this strategy include the original reduction from #SAT to counting unweighted perfect matchings [27]: In this setting, let G be a graph with edge-weights $w : E(G) \to \{-1, 1\}$, let $\mathcal{PM}[G]$ denote its set of perfect matchings, and define

$$\text{PerfMatch}(G) := \sum_{M \in \mathcal{PM}[G]} \prod_{e \in M} w(e). \tag{1}$$

Given an instance to this weighted problem, that is, a graph G derived from a 3-CNF formula, its space of witness structures $\mathcal{PM}[G]$ can then be partitioned into "good" structures that correspond to satisfying assignments, and "bad" structures that could be called combinatorial noise. By careful construction of such a graph G on edge-weights ± 1, we can ensure that bad structures come in pairs of weight $+1$ and -1, thus canceling out, whereas good structures all have weight $+1$.

To conclude #P-completeness of counting unweighted perfect matchings, it remains to simulate the weight -1 from the intermediate problem. This can be achieved by several techniques, which we survey in the next part of the introduction. Let us however first point out that the main contribution of this paper is a novel technique for precisely this part of the reduction: Using a method we call *parity separation*, we reduce the computation of $\text{PerfMatch}(G)$ for a ± 1-weighted graph G to the difference of PerfMatch for two unweighted graphs, that is, to the difference of two numbers of perfect matchings.

▶ **Lemma 1** (Parity Separation). *Let G be a graph on n vertices and m edges that is weighted by a function $w : E(G) \to \{-1, 1\}$. Then we can construct in time $\mathcal{O}(n + m)$ two unweighted graphs G_1 and G_2, each on $\mathcal{O}(n + m)$ vertices and edges, such that*

$$\text{PerfMatch}(G) = \text{PerfMatch}(G_1) - \text{PerfMatch}(G_2). \tag{2}$$

Intuitively speaking, this allows us to "collect" positive and negative terms of $\text{PerfMatch}(G)$ for ± 1-weighted graphs. This way, we can reduce the effect of cancellations incurred *within* PerfMatch to a mere difference *outside* of PerfMatch.

In the remainder of this introduction, we present parity separation in more detail and demonstrate three applications that can be derived from it: Firstly, and not surprisingly, we obtain a new #P-completeness proof for counting perfect matchings. Secondly, we can show $\mathsf{C}_{=}\mathsf{P}$-completeness of deciding whether two graphs have the same number of perfect matchings. Thirdly, we also obtain tight lower bounds under the exponential-time hypothesis.

1.1 #P-completeness via parity separation

To put parity separation into context, we first recapitulate Valiant's #P-hardness result for counting perfect matchings in more detail. Let us denote the problem of evaluating PerfMatch on graphs with edge-weights from $A \subseteq \mathbb{Q}$ by PerfMatch^A. For consistency with [13], we write PerfMatch^{0,1} for the problem of counting perfect matchings in unweighted graphs. That is, we explicitly include $0 \in A$ for PerfMatch^A, although zero-weight edges could be simply deleted.

First step: From #SAT to PerfMatch$^{-1,0,1}$

It is shown in [27, Lemma 3.1] that PerfMatch^W is #P-hard for $W := \{-1, 0, 1, 2, 3\}$. More precisely, from a 3-CNF formula φ, a number $t(\varphi) \in \mathbb{N}$ and a bipartite graph $G = G(\varphi)$ on weights W are constructed in polynomial time, such that

$$\#\mathsf{SAT}(\varphi) = \frac{\mathrm{PerfMatch}(G)}{4^{t(\varphi)}}, \tag{3}$$

This however only yields hardness for a weighted generalization of counting perfect matchings. To obtain a useful reduction source for further problems, it is crucial to reduce PerfMatch^W to PerfMatch^{0,1}, as reductions from PerfMatch to other problems would otherwise need to take care of the weights in W, which is particularly problematic for the edge-weight -1 in the case of unweighted reduction targets.

In fact, the weight -1 is the only problem we encounter: Edges e of positive integer edge-weight w can be simulated easily by replacing e with w parallel edges of unit weight, possibly subdividing edges twice to obtain simple graphs. This trick however clearly does not apply for the weight -1, so we need a different strategy.

Second step: From PerfMatch$^{-1,0,1}$ to PerfMatch^{0,1}

By now, two different strategies are known for removing the weight -1, which we briefly survey in the following. Let G be a graph with n vertices and $m > 0$ edges, all on weights -1 and 1.

- **Modular arithmetic**: Variations of the following approach were originally used by Valiant [27] and later by Zanko [32] and Ben-Dor and Halevi [1]: Write $M = 2^m + 1$ and observe that $\mathrm{PerfMatch}(G) < M$. We can hence replace the weight -1 by the positive integer $M - 1$ to obtain a graph G' satisfying $\mathrm{PerfMatch}(G) \equiv \mathrm{PerfMatch}(G')$ modulo M. The weight $M - 1$ can be simulated by a gadget as in the previous paragraph, and using a more involved construction [32], it can be seen that a gadget on $\mathcal{O}(m)$ vertices and edges suffices, yielding a total number of $\mathcal{O}(nm)$ vertices and $\mathcal{O}(m^2)$ edges in G'. Then we compute $\mathrm{PerfMatch}(G')$ modulo M and obtain $\mathrm{PerfMatch}(G)$, as we may assume from (3) that $\mathrm{PerfMatch}(G) \geq 0$. In total, we obtain one reduction image for PerfMatch^{0,1} on $\mathcal{O}(nm)$ vertices and $\mathcal{O}(m^2)$ edges.
- **Polynomial interpolation:** An alternative technique for removing the edge-weight -1 from G is to replace it by an indeterminate x. This gives rise to a graph G_x on edge-weights $\{1, x\}$ for which $\mathrm{PerfMatch}(G_x)$ is a polynomial $p(x) \in \mathbb{Z}[x]$ of degree at most $n/2$. We can evaluate $p(i)$ for $i \in \{0, \ldots, n/2\}$ by substituting $x \leftarrow i$ in G_x and simulating this positive weight by a gadget as discussed before. This allows us to recover $p(-1) = \mathrm{PerfMatch}(G)$ via Lagrangian interpolation. In total, using gadgets as in [32, 13], we obtain $\mathcal{O}(n)$ reduction images for PerfMatch^{0,1} on $\mathcal{O}(m \log n)$ vertices and $\mathcal{O}(m \log n + m)$ edges each.

Both weight removal techniques allow to reduce PerfMatch$^{-1,0,1}$ to PerfMatch^{0,1} and thus complete the #P-completeness proof of the latter problem. Note however that both approaches map weighted graphs G with m edges to unweighted graphs with a super-linear number of edges. Using parity separation, we obtain a third way of performing the weight removal step, which differs substantially from both approaches mentioned before and features only constant blowup:

- **Parity separation:** Using Lemma 1, compute two unweighted graphs G_1 and G_2 from G such that PerfMatch(G) is the mere difference of PerfMatch(G_1) and PerfMatch(G_2). In total, we obtain 2 reduction images for PerfMatch^{0,1} on $\mathcal{O}(n+m)$ vertices and edges.

Together with the first step, this implies an alternative #P-completeness proof for the problem PerfMatch^{0,1} of counting perfect matchings in unweighted graphs. For the sake of completeness, we also include a self-contained reduction from #SAT to PerfMatch$^{-1,0,1}$.

▶ **Theorem 2.** PerfMatch^{0,1} *is* #P*-complete under polynomial-time Turing reductions.*

1.2 $\mathsf{C}_=\mathsf{P}$-completeness via parity separation

Apart from an alternative #P-completeness proof, Lemma 1 also yields implications for the structural complexity of PerfMatch: We show that deciding whether two unweighted graphs have the same number of perfect matchings is complete for the complexity class $\mathsf{C}_=\mathsf{P}$, which was introduced in [25, 31] and further elaborated in [16, 15].

To define $\mathsf{C}_=\mathsf{P}$, let us associate the following language $\mathsf{A}_=$ with each counting problem $\mathsf{A} \in \#\mathsf{P}$: The inputs to $\mathsf{A}_=$ are pairs (x, y) of instances to A, and we are asked to determine whether $\mathsf{A}(x) = \mathsf{A}(y)$ holds. We can then define[1] the class $\mathsf{C}_=\mathsf{P} := \{\mathsf{A}_= \mid \mathsf{A} \in \#\mathsf{P}\}$.

For instance, it is clear that $\#\mathsf{SAT}_=$, the problem that asks whether two 3-CNF formulas have the same number of satisfying assignments, is $\mathsf{C}_=\mathsf{P}$-complete under polynomial-time many-one reductions. In fact, $\mathsf{C}_=\mathsf{P}$-completeness holds for every problem $\mathsf{A}_=$ whose counting version #A is #P-complete under parsimonious reductions. We recall the notion of parsimonious (and other) reductions in Definition 5.

The relationship between $\mathsf{C}_=\mathsf{P}$ and other complexity classes has been studied in structural complexity theory, and several results are surveyed in [15]. For instance, we clearly have $\mathsf{coNP} \subseteq \mathsf{C}_=\mathsf{P}$, and using the witness isolation technique [30], we see that NP is contained in $\mathsf{C}_=\mathsf{P}$ under randomized reductions. Let us also observe that $\mathsf{NP}^{\#\mathsf{P}} \subseteq \mathsf{NP}^{\mathsf{C}_=\mathsf{P}}$: Whenever we issue an oracle call to #P, we may instead guess the output number, and then check whether we guessed correctly by using the $\mathsf{C}_=\mathsf{P}$ oracle.

To the best of the author's knowledge, no *natural* $\mathsf{C}_=\mathsf{P}$-complete problem $\mathsf{A}_=$ is known whose counting version A is *not* #P-complete under parsimonious reductions.[2] It is clear that the problem PerfMatch^{0,1} of counting unweighted perfect matchings cannot be #P-complete under parsimonious reductions, unless $\mathsf{P} = \mathsf{NP}$. Therefore, the following completeness

[1] We deviate here from the standard definition of $\mathsf{C}_=\mathsf{P}$, according to which we have $L \in \mathsf{C}_=\mathsf{P}$ iff the following holds: There is a polynomial-time nondeterministic Turing machine M such that $x \in L$ iff the numbers of accepting and rejecting computation paths of $M(x)$ are equal. It can be verified easily that this is equivalent to our definition.

[2] Here, we stressed *natural*, because we can easily construct artificial $\mathsf{C}_=\mathsf{P}$-complete problems $\mathsf{A}_=$ whose counting version #A admits no parsimonious reduction from #SAT: Consider as an example the counting problem $\#\mathsf{SAT}'$ that asks to count satisfying assignments, incremented by 1. If $\#\mathsf{SAT}'$ had a parsimonious reduction from #SAT, then every CNF-formula would be satisfiable. On the other hand, the reduction from $\#\mathsf{SAT}_=$ to $\#\mathsf{SAT}'_=$ is trivial.

result for $\text{PerfMatch}^{0,1}_{=}$ seems relevant for structural complexity theory, as it establishes a $\mathsf{C}_{=}\mathsf{P}$-variant of Valiant's result.

▶ **Theorem 3.** $\text{PerfMatch}^{0,1}_{=}$ *(deciding whether two unweighted graphs have the same number of perfect matchings) is* $\mathsf{C}_{=}\mathsf{P}$*-complete under polynomial-time many-one reductions.*

To prove this theorem, we first reduce instances (φ, φ') for $\#\mathsf{SAT}_{=}$ to ± 1-weighted graphs G that satisfy $\text{PerfMatch}(G) = 0$ iff $\#\mathsf{SAT}(\varphi) = \#\mathsf{SAT}(\varphi')$. This requires a modification of the first step in the $\#\mathsf{P}$-hardness reduction, which is however supported easily by our alternative proof. Then we apply Lemma 1 on the graph G to obtain unweighted graphs G_1 and G_2 satisfying (2). In particular, their numbers of perfect matchings agree iff $\text{PerfMatch}(G)$ vanishes, that is, iff (φ, φ') is a yes-instance for $\#\mathsf{SAT}_{=}$.

To conclude this subsection, we note that the complexity of a similar problem was posed as an open question in [8]: Given two directed acyclic graphs, decide whether their numbers of topological orderings agree. It was shown in [5] that counting topological orderings is $\#\mathsf{P}$-complete under Turing reductions, but the decision version is trivial for DAGs. Our result for $\text{PerfMatch}^{0,1}_{=}$ might be useful to prove $\mathsf{C}_{=}\mathsf{P}$-completeness for this and other problems. For instance, a reduction was recently found [24] from $\text{PerfMatch}^{0,1}_{=}$ to deciding whether two formulas in 2-CNF are satisfied by the same number of assignments.

1.3 Tight lower bounds via parity separation

We turn our attention to conditional *quantitative* lower bounds: A relatively new subfield in computational complexity makes use of assumptions stronger than $\mathsf{P} \neq \mathsf{NP}$ or $\mathsf{FP} \neq \#\mathsf{P}$ to prove tight (exponential) lower bounds on the running times needed to solve computational problems. A popular such assumption is the exponential-time hypothesis ETH, introduced by Impagliazzo et al. [19, 20], which states that the satisfiability of n-variable formulas φ in 3-CNF cannot be decided in time $2^{o(n)}$. For counting problems, an analogous variant $\#\mathsf{ETH}$ was introduced by Dell et al. [13], and it postulates the same for the problem of *counting* satisfying assignments to φ.

Assuming ETH, it was shown for a vast body of popular decision problems that the known exponential-time exact algorithms are somewhat optimal: For instance, there is a trivial $2^{\mathcal{O}(m)}$ time algorithm for finding a Hamiltonian cycle (or various other structures) in an m-edge graph, but $2^{o(m)}$ time algorithms would refute ETH. See [23] for a nice survey.

Similar lower bounds were shown for counting problems under $\#\mathsf{ETH}$, see [17, 18, 13], and a very recent paper [9] introduced *block interpolation*, an approach to make the technique of polynomial interpolation (as seen in the second step of Section 1.1) compatible with tight lower bounds under $\#\mathsf{ETH}$. For several problems, that of counting perfect matchings being among them, block interpolation gave the first tight $2^{\Omega(m)}$ lower bounds under $\#\mathsf{ETH}$.

When applying this framework to $\text{PerfMatch}^{0,1}$, we would first reduce $\#\mathsf{SAT}$ on n-variable 3-CNFs φ to instances $G = G(\varphi)$ for $\text{PerfMatch}^{-1,0,1}$ with $\mathcal{O}(n)$ edges as in the first step of the $\#\mathsf{P}$-hardness proof. Then we apply the block interpolation technique to reduce G to $2^{o(n)}$ unweighted instances G' for $\text{PerfMatch}^{0,1}$ with $\mathcal{O}(n)$ edges. While this sub-exponential number of instances is compatible with the goal of proving tight lower bounds, it leaves open the natural question whether the same reduction could be achieved with only polynomially many oracle calls on graphs with $\mathcal{O}(n)$ edges.

Using Lemma 1, we obtain a strong positive answer to this question: Replacing the application of block interpolation by one of parity separation, we obtain a reduction to merely *two* instances of $\text{PerfMatch}^{0,1}$. And as a synthesis of structural and quantitative complexity, we also obtain a tight lower bound for the equality-testing problem $\text{PerfMatch}^{0,1}_{=}$.

▶ **Theorem 4.** *Unless* #ETH *fails, the problem* $\text{PerfMatch}^{0,1}$ *admits no algorithm with running time* $2^{o(m)}$ *on simple graphs with* m *edges. Furthermore, the same applies to* $\text{PerfMatch}^{0,1}_{=}$ *under the decision version* ETH*.*

Organization of this paper

The remainder of this paper is structured as follows: In Section 2, we introduce the Holant framework and matchgates, concepts that are crucial to our constructions. These are put to use in Section 3, where we prove Lemma 1, our main result. Its applications, as discussed above, are shown in Section 4.

2 Preliminaries

Graphs in this paper may be edge- or vertex-weighted. Given a graph G and $v \in V(G)$, denote the edges incident with v by $I(v)$. If the context of an argument unambiguously determines a graph G, we write $n = |V(G)|$ and $m = |E(G)|$.

We denote the Hamming weight of strings $x \in \{0,1\}^*$ by $\text{hw}(x)$. Given a statement φ, we let $[\varphi] = 1$ if φ is true, and $[\varphi] = 0$ otherwise. For convenience, we recall that several reduction notions are distinguished in the study of counting complexity: The most restrictive notion is that of *parsimonious* (many-one) reductions, which can be slightly relaxed to *weakly* parsimonious reductions. The most permissive notion is that of *Turing* reductions.

▶ **Definition 5.** Let A and B be counting problems. Let $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \{0,1\}^* \to \mathbb{Q}$ be polynomial-time computable functions. If $\mathsf{A}(x) = g(x) \cdot \mathsf{B}(f(x))$ holds for all $x \in \{0,1\}^*$, then we call (f, g) a *weakly parsimonious (polynomial-time) reduction* from A to B and write $\mathsf{A} \leq_p \mathsf{B}$. If additionally $g(x) = 1$ holds for all $x \in \{0,1\}^*$, then we call f *parsimonious* and write $\mathsf{A} \leq_p^{pars} \mathsf{B}$.

If $\mathbb{T}$ is a deterministic polynomial-time algorithm that solves A with an oracle for B, then we call $\mathbb{T}$ a *Turing reduction* from A to B and write $\mathsf{A} \leq_p^T \mathsf{B}$.

2.1 Weighted sums of (perfect) matchings

The quantity PerfMatch on edge-weighted graphs, as defined in (1) and [29], will be the central object of investigation in this paper. For intermediate steps, we also consider the quantity MatchSum introduced in [29].

▶ **Definition 6.** For vertex-weighted graphs G with $w : V(G) \to \mathbb{Q}$, let $\mathcal{M}[G]$ denote the set of (not necessarily perfect) matchings in G. Recall that $\mathcal{PM}[G] \subseteq \mathcal{M}[G]$ denotes the perfect matchings in G. For $M \in \mathcal{M}[G]$, let $\text{usat}(M)$ denote the set of unmatched vertices in M. Then we define

$$\text{MatchSum}(G) = \sum_{M \in \mathcal{M}[G]} \prod_{v \in \text{usat}(M)} w(v).$$

Given $W \subseteq \mathbb{Q}$, we write PerfMatch^W for the problem of evaluating $\text{PerfMatch}(G)$ on graphs G with weights $w : E(G) \to W$. Likewise, write MatchSum^W on graphs with weights $w : V(G) \to W$. Please note that an edge of weight 0 in PerfMatch can be treated as if it were not present, whereas weight 0 at a vertex v in MatchSum signifies that v must be matched. We can easily reduce PerfMatch^W for finite $W \subseteq \mathbb{Q}$ to $\text{PerfMatch}^{-1,0,1}$:

▶ **Lemma 7** (folklore). *Let G be edge-weighted by $w : E(G) \to \mathbb{Q}$. Let $q \in \mathbb{N}$ denote the least common denominator of the weights in G, and let $T = \max_{e \in E(G)} q \cdot w(e)$. Then we can compute a number $B \in \mathbb{N}$ and an edge-weighted graph G' on $\mathcal{O}(n + Tm)$ vertices and edges, all of weight ± 1, such that* $\text{PerfMatch}(G) = q^{-B} \cdot \text{PerfMatch}(G')$.

2.2 Holant problems

We give an introduction to the *Holant framework*, summarizing ideas from [29, 6, 7]. A more detailed introduction to our notation can be found in [11].

▶ **Definition 8** (adapted from [29]). A *signature graph* is an edge-weighted graph Ω, which may feature parallel edges, with a *vertex function* $f_v : \{0,1\}^{I(v)} \to \mathbb{Q}$ at each $v \in V(\Omega)$.

The *Holant* of Ω is a particular sum over edge assignments $x \in \{0,1\}^{E(\Omega)}$. We sometimes identify x with the set $x^{-1}(1)$ of indices that have value 1 under x. Given $S \subseteq E(\Omega)$, we write $x|_S$ for the restriction of x to S, which is the unique assignment in $\{0,1\}^S$ that agrees with x on S. Then we define

$$\text{Holant}(\Omega) := \sum_{x \in \{0,1\}^{E(\Omega)}} \left(\prod_{e \in x} w(e) \right) \left(\prod_{v \in V(\Omega)} f_v(x|_{I(v)}) \right). \tag{4}$$

As a first example, we can reformulate $\text{PerfMatch}(G)$ easily as the Holant of a signature graph $\Omega = \Omega(G)$ by declaring $f_v : \{0,1\}^{I(v)} \to \{0,1\}$ for $v \in V(G)$ to be the vertex function that maps $x \in \{0,1\}^*$ to 1 iff $\text{hw}(x) = 1$ and to 0 else.

When considering signature graphs Ω in the following, we will always assume that $I(v)$ for each $v \in V(\Omega)$ is ordered in a fixed (usually implicit) way. This way, if v is a vertex of degree $d \in \mathbb{N}$, we can simply view f_v as a function $f_v : \{0,1\}^d \to \mathbb{Q}$, and we call this representation a *signature*.

▶ **Example 9.** The following are signatures of arity $k \in \mathbb{N}$ on inputs $x \in \{0,1\}^{[k]}$ with $x = (x_1, \ldots, x_k)$.

$$\begin{aligned}
\texttt{EQ} &: x \mapsto [x_1 = \ldots = x_k] \\
\texttt{HW}_{=1} &: x \mapsto [\text{hw}(x) = 1] \\
\texttt{HW}_{\leq 1} &: x \mapsto [\text{hw}(x) \leq 1] \\
\texttt{ODD} &: x \mapsto x_1 \oplus \ldots \oplus x_k \\
\texttt{EVEN} &: x \mapsto 1 \oplus x_1 \oplus \ldots \oplus x_k.
\end{aligned}$$

We may write, say, $\texttt{EQ}_4$ to denote the arity-4 signature $\texttt{EQ}$. Note that these signatures are symmetric, as they depend only upon the Hamming weight on the input.

Similarly as for PerfMatch, we can also express MatchSum as a Holant problem.

▶ **Lemma 10.** *Let G be a graph with vertex-weights $w : V(G) \to \mathbb{Q}$. Then* $\text{MatchSum}(G) = \text{Holant}(\Omega)$ *holds with the signature graph Ω that is derived from G by placing $\texttt{VTX}_{w(v)}$ at $v \in V(G)$ and assigning weight 1 to all edges. Here, $\texttt{VTX}_w$ for $w \in \mathbb{Q}$ is defined as*

$$\texttt{VTX}_w : \quad x \mapsto \begin{cases} w & \text{if } \text{hw}(x) = 0, \\ 1 & \text{if } \text{hw}(x) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

We can easily reduce edge-weighted Holant problems to unweighted versions as follows.

▶ **Lemma 11.** *Let Ω' be defined as follows from Ω: Subdivide each edge $e \in E(\Omega)$ into two edges, assign weight 1 to the obtained subdivision edges, and equip the obtained subdivision vertices with the signature $\texttt{EDGE}_{w(e)}$, where*

$$\texttt{EDGE}_w : \quad x \mapsto \begin{cases} w & \text{if } x = 11, \\ 0 & \text{if } x \in \{01, 10\}, \\ 1 & \text{if } x = 00. \end{cases}$$

Then Ω' features only the edge-weight 1, and we have $\mathrm{Holant}(\Omega) = \mathrm{Holant}(\Omega')$.

Finally, a signature is called *even* if its support contains only bitstrings of even Hamming weight. The problem #SAT can be rephrased as a Holant problem with even signatures:

▶ **Lemma 12.** *For $n, m, d \in \mathbb{N}$, let φ be a d-CNF formula on variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$. We construct a signature graph Ω as follows:*

- *For each $i \in [n]$, let $r(i)$ denote the number of occurrences of x_i (as a positive or negative literal) in φ. Create a* variable vertex *v_i in Ω, with signature $\texttt{EQ}_{2r(i)}$.*
- *For each $j \in [m]$, let $x_{i_1}, \ldots, x_{i_d}$ be the variables that clause c_j depends upon. We create a* clause vertex *w_j in Ω, and for $\kappa \in [d]$, we add two parallel edges of weight 1 between w_j and v_{i_κ} as the $2\kappa - 1$-th and 2κ-th edges in the ordering of $I(w_j)$.*
- *For each $j \in [m]$, consider clause c_j as a Boolean function on variables $z_1, \ldots, z_d$, where z_i for $i \in [d]$ represents the i-th variable in c_j. Define a function c'_j on variables $y_1, \ldots, y_{2d}$ that outputs $c_j(y_1, y_3, \ldots, y_{2d-1})$ if $y_{2i} = y_{2i-1}$ for all $i \in [d]$. On all other inputs, the value of c'_j is defined to be zero. Assign such a signature c'_j to the vertex w_j.*

Then we have $\#\mathsf{SAT}(\varphi) = \mathrm{Holant}(\Omega)$. Note that Ω is a signature graph with $n + m$ vertices and $2dm$ edges that features only even signatures and the edge-weight 1.

▶ Remark. The degree of clause vertices above is $2d$ rather than d to ensure that their signatures are even. The need for this will become clear in the next subsection.

2.3 Gates and matchgates

Given a signature graph Ω, we can sometimes simulate vertex functions by gadgets or *gates*, which are signature graphs with so-called *dangling edges* that feature only one endpoint. These notions are borrowed from the $\mathcal{F}$-gates in [7]. Matchgates were first considered in [29].

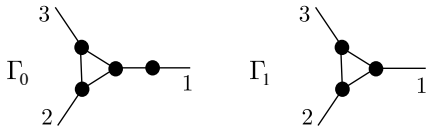
▶ **Definition 13** (adapted from [7])**.** For disjoint sets A, B, and for assignments $x \in \{0,1\}^A$ and $y \in \{0,1\}^B$, we write $xy \in \{0,1\}^{A \cup B}$ for the assignment that agrees with x on A, and with y on B. We also say that the assignment xy *extends* x. A *gate* is a signature graph Γ containing a set $D \subseteq E(\Gamma)$ of dangling edges, all of which have edge-weight 1. The *signature realized by* Γ is the function $\mathrm{Sig}(\Gamma) : \{0,1\}^D \to \mathbb{Q}$ that maps x to

$$\mathrm{Sig}(\Gamma, x) = \sum_{y \in \{0,1\}^{E(\Gamma) \setminus D}} \left(\prod_{e \in xy} w(e) \right) \left(\prod_{v \in V(\Gamma)} f_v(xy|_{I(v)}) \right). \tag{5}$$

A gate Γ is a *matchgate* if it features only the signature $\texttt{HW}_{=1}$.

In the following, we consider the dangling edges D of gates Γ to be labelled as $1, \ldots, |D|$. This way, we can view $\mathrm{Sig}(\Gamma)$ as a function of type $\{0,1\}^{|D|} \to \mathbb{Q}$ instead of $\{0,1\}^D \to \mathbb{Q}$. We will use gates to realize required signatures as "gadgets" consisting of other (usually simpler) signatures. Consider the following example, which appeared in [29].

▶ **Example 14.** It can be verified that $\texttt{EVEN}_3$ and $\texttt{ODD}_3$ are realized by the matchgates Γ_0 and Γ_1 below, where all vertices are assigned $\texttt{HW}_{=1}$ and all edges have weight 1.



Using this, we can realize the signatures $\texttt{ODD}_k$ and $\texttt{EVEN}_k$ for any arity $k \geq 3$ as matchgates, noted also in [29, Theorem 3.3]. This will be required in Section 3.

▶ **Example 15.** For all $k \geq 3$, there exists a gate $\Gamma_{\texttt{EVEN}}$ with $\text{Sig}(\Gamma_{\texttt{EVEN}}) = \texttt{EVEN}_k$. It consists of vertices $v_1, \ldots, v_{k-2}$ equipped with $\texttt{EVEN}_3$, edges $e_1, \ldots, e_{k-3}$ of weight 1, and dangling edges $[k]$.

We can likewise realize $\texttt{ODD}_k$ by a gate $\Gamma_{\texttt{ODD}}$ that is constructed as above, but with $\texttt{ODD}_3$ rather than $\texttt{EVEN}_3$ at v_{k-2}.

In the following, we formalize the operation of *inserting* a gate Γ into a signature graph so as to simulate a desired signature. A more detailed version of this operation can be found in Definition 2.10 and Lemma 2.11 of [11].

▶ **Lemma 16.** *Let Ω be a signature graph, let $v \in V(\Omega)$ with $D = I(v)$ and let Γ be a gate with dangling edges D. We can* insert Γ at v *by deleting v and keeping D as dangling edges, and then placing Γ into Ω and identifying each dangling edge $e \in D$ across Γ and Ω. If Ω' is derived from Ω by inserting a gate Γ with $\text{Sig}(\Gamma) = f_v$ at v, then $\text{Holant}(\Omega) = \text{Holant}(\Omega')$.*

By an argument presented in the author's PhD thesis [11], also used in [12], we can realize every even signature f by some matchgate $\Gamma = \Gamma(f)$. If the image of f is W, then Γ contains $W \cup \{\pm 1, 1/2\}$ as edge-weights. For sake of completeness, we include a self-contained proof in the full version.

▶ **Lemma 17** ([11, 12])**.** *Let Ω be a signature graph on n vertices and m edges, with even vertex functions $\{f_v\}_{v \in V(\Omega)}$ that map into $W \subseteq \mathbb{Q}$. Let $s = \max_{v \in V(\Omega)} |\text{supp}(f_v)|$. Then we can construct, in linear time, a graph G on $\mathcal{O}(n + sm)$ vertices and edges such that $\text{Holant}(\Omega) = \text{PerfMatch}(G)$. The edge-weights of G are $W \cup \{\pm 1, 1/2\}$.*

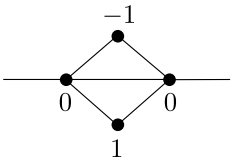
3 The parity separation technique

We are ready to prove Lemma 1, our main result. The proof proceeds by establishing, with several intermediate steps, the reduction chain

$$\text{PerfMatch}^{-1,0,1} \leq_p \text{MatchSum}^{-1,0,1} \leq_p^T \text{PerfMatch}^{0,1}. \tag{6}$$

For the first reduction in (6), we apply a gadget Γ realizing the signature $\texttt{EDGE}_{-1}$ from Lemma 11 to all edges of weight -1.

▶ **Lemma 18.** *We have $\texttt{EDGE}_{-1} = \text{Sig}(\Gamma)$, where Γ is the gate below. In Γ, each vertex features the signature $\texttt{VTX}_w$ for the number $w \in \{-1, 0, 1\}$ it is annotated with in the drawing.*



This allows us to transform an instance for PerfMatch$^{-1,0,1}$ to one for MatchSum$^{-1,0,1}$.

▶ **Lemma 19.** *Let G be a graph with n vertices and m edges, all of weight ± 1. Then we can compute a graph G' on $\mathcal{O}(n+m)$ edges, with vertices of weight $\{-1,0,1\}$, such that* PerfMatch(G) = MatchSum(G').

Proof. We assume that $|V(G)|$ is even, as otherwise PerfMatch$(G) = 0$. First, let Ω be the signature graph constructed by assigning $\texttt{HW}_{=1}$ to all vertices of G, and then applying the signature $\texttt{EDGE}_{-1}$ as in Lemma 11. We obtain PerfMatch(G) = Holant(Ω).

Then realize each occurrence of $\texttt{EDGE}_{-1}$ by the gate Γ from Lemma 18. Note that Γ features no edge-weights, and only $\texttt{VTX}_w$ for $w \in \{-1,0,1\}$. We obtain a signature graph Ω' whose signatures are all of the type $\texttt{VTX}_w$ for $w \in \{-1,0,1\}$, and which satisfies Holant(Ω) = Holant(Ω'). Note that $\texttt{HW}_{=1} = \texttt{VTX}_0$, so this indeed covers all vertices of Ω'.

By Lemma 10, we may equivalently consider Holant(Ω') = MatchSum(G'), where G' is a vertex-weighted graph obtained from Ω' as follows: Keep all vertices and edges of Ω' intact, and if $v \in V(\Omega')$ features the signature $\texttt{VTX}_w$, for $w \in \{-1,0,1\}$, then assign the vertex weight w to v in G'. ◀

For the second reduction in (6), we perform the actual act of parity separation: We will split the vertex-weighted graph G' into an even part G_0 and an odd part G_1, both unweighted, such that the *perfect* matchings of the even (resp. odd) part correspond bijectively to the matchings of G' with an even (resp. odd) number of unmatched vertices of weight -1.[3] Since $(-1)^{even} = 1$ and $(-1)^{odd} = -1$, this clearly implies that MatchSum(G) is the difference of PerfMatch(G_0) and PerfMatch(G_1).

To proceed, we first use the signatures EVEN and ODD from Example 9 to obtain an alternative reformulation of MatchSum$^{-1,0,1}$ as the difference of two Holants.

▶ **Lemma 20.** *Let G' be a graph with vertex-weights $\{-1,0,1\}$. For $a,b \in \{0,1\}$, let $\Phi_{ab} = \Phi_{ab}(G')$ be the signature graph obtained as follows:*

1. *Assign the signature $\texttt{HW}_{=1}$ to all vertices of G'.*
2. *For $x \in \{-1,0,1\}$, let $V_x \subseteq V(G')$ denote the set of vertices of weight x in G'. For $x \in \{-1,1\}$, add a vertex u_x connected to V_x.*
   - *Assign to u_{-1} the signature EVEN if $a = 0$, and assign ODD if $a = 1$.*
   - *Assign to u_1 the signature EVEN if $b = 0$, and assign ODD if $b = 1$.*

   Then we have MatchSum(G') = Holant(Φ_{00}) − Holant(Φ_{11}).

The second reduction in (6) follows by realizing the signatures ODD and EVEN appearing in Φ_{00} and Φ_{11} via matchgates that feature neither edge- nor vertex-weights. Note that the only other appearing signature $\texttt{HW}_{=1}$ is trivially realized by such a matchgate.

Proof of Lemma 1. Follows from Lemma 19 (to reduce PerfMatch$^{-1,0,1}$ to MatchSum$^{-1,0,1}$) with Lemma 20 (to reformulate MatchSum$^{-1,0,1}$ as a Holant problem) and Example 15 (to realize the ODD and EVEN signatures in the Holant problem by unweighted matchgates). ◀

[3] This step is inspired by a reduction [29, Theorem 3.3] from certain instances of MatchSum on planar graphs to PerfMatch on planar graphs.

4 Parity separation in action

In the final section of this paper, we cover the three applications of parity separation that we discussed in the introduction.

4.1 Completeness for #P

We can easily show the #P-completeness of $\text{PerfMatch}^{0,1}$ via parity separation. To this end, we first express #SAT as a Holant problem on even signature graphs, as seen in Lemma 12. Together with Lemma 17, this yields $\#\mathsf{SAT} \leq_p \text{PerfMatch}^B$ with $B = \{-1, 0, 1/2, 1\}$. We use Lemma 7 to remove the edge-weight $1/2$, and finally remove the weight -1 by parity separation as in Lemma 1. Altogether, we obtain the following lemma.

▶ **Lemma 21.** *Let φ be a 3-CNF formula with n variables and m clauses. Then we can compute a number $T \in \mathbb{N}$ and construct two unweighted graphs G_1 and G_2 on $\mathcal{O}(n+m)$ vertices and edges, all in time $\mathcal{O}(n+m)$, such that $2^T \cdot \#\mathsf{SAT}(\varphi) = \text{PerfMatch}(G_1) - \text{PerfMatch}(G_2)$.*

This readily implies Theorem 2, the desired #P-completeness result.

4.2 Completeness for $\mathsf{C}_=\mathsf{P}$

For our next application, we apply the parity separation technique to prove Theorem 3. That is, we prove $\mathsf{C}_=\mathsf{P}$-completeness of the problem $\text{PerfMatch}^{0,1}_=$ that asks, given two unweighted graphs G_1 and G_2, whether their numbers of perfect matchings agree. We call graphs satisfying this property *equipollent graphs* and will likewise speak of *equipollent formulas* if their numbers of satisfying assignments agree.

Proof of Theorem 3. The problem $\text{PerfMatch}^{0,1}_=$ is clearly contained in $\mathsf{C}_=\mathsf{P}$. For the hardness part, we reduce from the $\mathsf{C}_=\mathsf{P}$-complete problem $\#\mathsf{SAT}_=$ that asks, given 3-CNF formulas φ and φ', to determine whether they are equipollent. To this end, we construct unweighted graphs G and G' that are equipollent if and only if φ and φ' are.

Assume that φ and φ' are defined on the same set of variables $x_1, \ldots, x_n$ and feature the same number m of clauses. This can be achieved by renaming variables, and by adding dummy variables and clauses. If, say, φ has less variables than φ', then we can add dummy variables to φ', together with clauses that ensure that every dummy variable has the same assignment as x_1. We can also duplicate clauses.

Let $C_1, \ldots, C_m$ and $C'_1, \ldots, C'_m$ denote the clauses in φ and φ', respectively. We introduce a *selector* variable x^* and define a formula ψ on the variable set $\mathcal{X} = \{x^*, x_1, \ldots, x_n\}$, which has clauses $D_1, \ldots, D_m$ and $D'_1, \ldots, D'_m$, where $D_i := (x^* \vee C_i)$ and $D'_i := (\neg x^* \vee C'_i)$ for $i \in [m]$. If $a(x^*) = 0$ holds in an assignment $a \in \{0,1\}^{\mathcal{X}}$, then all clauses $D'_1, \ldots, D'_m$ are satisfied by $\neg x^*$, but in order for a to satisfy ψ, the clauses $D_1, \ldots, D_m$ have to be satisfied by $x_1, \ldots, x_n$. In other words, if a satisfies ψ and $a(x^*) = 0$, then the restriction of a to $x_1, \ldots, x_n$ satisfies φ. Likewise, if a satisfies ψ and $a(x^*) = 1$, then the restriction of a to $x_1, \ldots, x_n$ satisfies φ'. Hence, we can define the following quantity

$$S := \sum_{a \in \{0,1\}^{\mathcal{X}}} (-1)^{a(x^*)} \cdot [\psi \text{ satisfied by } a]$$

and we observe that $S = \#\mathsf{SAT}(\varphi) - \#\mathsf{SAT}(\varphi')$. It is clear that $S = 0$ if and only if φ and φ' are equipollent. As in Lemma 12, we then express $S = \text{Holant}(\Omega)$ for a signature graph

$\Omega = \Omega(\psi)$, with one modification: At the vertex v^* corresponding to the variable x^*, we do not use the signature $\texttt{EQ}$, but rather a modified signature

$$\texttt{EQ}_{-} : \quad y \mapsto \begin{cases} -1 & \text{if } y = 1 \dots 1, \\ 1 & \text{if } y = 0 \dots 0, \\ 0 & \text{otherwise.} \end{cases}$$

We realize Ω via Lemma 17 to obtain a graph G on edge-weights $1/2, \pm 1$, simulate the edge-weight $1/2$ via Lemma 7, and obtain an edge-weighted graph H with weights ± 1 together with a number $T \in \mathbb{N}$ such that

$$S = \text{Holant}(\Omega) = 2^{-T} \cdot \text{PerfMatch}(H). \tag{7}$$

Using Lemma 1, we then obtain unweighted graphs G and G' such that

$$\text{PerfMatch}(H) = \text{PerfMatch}(G) - \text{PerfMatch}(G'). \tag{8}$$

Then G and G' are equipollent iff $S = 0$, which in turn holds iff φ and φ' are equipollent. ◀

4.3 Tight lower bounds under #ETH

By the exponential-time hypothesis #ETH, there is no $2^{o(n)}$ time algorithm for counting satisfying assignments to 3-CNF formulas φ with n variables. Applying the counting version of the so-called sparsification lemma, shown in [13], we may additionally assume that φ features $m = \mathcal{O}(n)$ clauses. Then Lemma 21 clearly implies the lower bound for $\text{PerfMatch}^{0,1}$ claimed in Theorem 4.

Concerning $\text{PerfMatch}^{0,1}_{=}$, it is even easier to prove lower bounds under ETH than to prove its $\mathsf{C}_{=}\mathsf{P}$-completeness, as we may (i) reduce from SAT rather than $\mathsf{SAT}_{=}$, and (ii) use the more permissive notion of Turing (rather than many-one) reductions: With Lemma 21, we can construct unweighted graphs G_1 and G_2 on $\mathcal{O}(m)$ vertices and edges that are equipollent iff φ is *unsatisfiable*, thus a $2^{o(m)}$ time algorithm would contradict ETH. This proves Theorem 4.

5 Conclusion and future work

We have added a new method to the known techniques (modular arithmetic and polynomial interpolation) for removing the edge-weight -1 from $\text{PerfMatch}^{-1,0,1}$. This method is based on matchgates and the simple observation that $(-1)^{even} = 1$ and $(-1)^{odd} = -1$. We obtained non-trivial applications that could not be obtained via the previously known techniques.

Our work leaves some interesting questions open for further investigations. For instance, we could not find a way to show #P-completeness of $\text{PerfMatch}^{0,1}$ on *bipartite* graphs by parity separation. Is there a complexity-theoretic explanation for this? On another note, can we prove $\mathsf{C}_{=}\mathsf{P}$-completeness for other "equality-testing" versions of counting problems?

Acknowledgments. The author wishes to thank Markus Bläser, Mingji Xia, Meena Mahajan and Jin-Yi Cai for interesting discussions. Furthermore, thanks to Patrick Scharpfenecker for pointing out [24].

References

1 Amir Ben-Dor and Shai Halevi. Zero-one permanent is *#P*-complete, A simpler proof. In *Second Israel Symposium on Theory of Computing Systems, ISTCS 1993, Natanya, Israel, June 7-9, 1993. Proceedings*, pages 108–117, 1993.

2 Markus Bläser and Radu Curticapean. The complexity of the cover polynomials for planar graphs of bounded degree. In *Mathematical Foundations of Computer Science 2011 – 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*, pages 96–107, 2011. doi:10.1007/978-3-642-22993-0_12.

3 Markus Bläser and Radu Curticapean. Weighted counting of k-matchings is #W[1]-hard. In *Parameterized and Exact Computation – 7th International Symposium, IPEC 2012, Ljubljana, Slovenia, September 12-14, 2012. Proceedings*, pages 171–181, 2012. doi:10.1007/978-3-642-33293-7_17.

4 Markus Bläser and Holger Dell. Complexity of the cover polynomial. In *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, pages 801–812, 2007. doi:10.1007/978-3-540-73420-8_69.

5 Graham Brightwell and Peter Winkler. Counting linear extensions is #P-complete. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 175–181, 1991. doi:10.1145/103418.103441.

6 Jin-Yi Cai and Pinyan Lu. Holographic algorithms: From art to science. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, pages 401–410, New York, NY, USA, 2007. ACM. doi:http://doi.acm.org/10.1145/1250790.1250850.

7 Jin-Yi Cai, Pinyan Lu, and Mingji Xia. Holographic algorithms by Fibonacci gates and holographic reductions for hardness. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 644–653. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.34.

8 Mike Chen. The complexity of checking whether two DAG have the same number of topological sorts, November 2010. URL: http://cstheory.stackexchange.com/questions/3105.

9 Radu Curticapean. Block interpolation: A framework for tight exponential-time counting complexity. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 380–392, 2015. doi:10.1007/978-3-662-47672-7_31.

10 Radu Curticapean. Parity separation: A scientifically proven method for permanent weight loss. *CoRR*, abs/1511.07480, 2015. URL: http://arxiv.org/abs/1511.07480.

11 Radu Curticapean. *The simple, little and slow things count: on parameterized counting complexity*. PhD thesis, Saarland University, August 2015.

12 Radu Curticapean and Dániel Marx. Tight conditional lower bounds for counting perfect matchings on graphs of bounded treewidth, cliquewidth, and genus. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1650–1669, 2016. doi:10.1137/1.9781611974331.ch113.

13 Holger Dell, Thore Husfeldt, Dániel Marx, Nina Taslaman, and Martin Wahlen. Exponential time complexity of the permanent and the Tutte polynomial. *ACM Transactions on Algorithms*, 10(4):21, 2014. doi:10.1145/2635812.

14 Jack Edmonds. Paths, trees, and flowers. In *Classic Papers in Combinatorics*, Modern Birkhauser Classics, pages 361–379. Birkhauser Boston, 1987. doi:10.1007/978-0-8176-4842-8_26.

15 Lane A. Hemaspaandra and Mitsunori Ogihara. *The Complexity Theory Companion*. Springer, 2002.

16 Lane A. Hemaspaandra and Heribert Vollmer. The satanic notations: counting classes beyond #P and other definitional adventures. *SIGACT News*, 26(1):2–13, 1995. doi:10.1145/203610.203611.

17 Christian Hoffmann. Exponential time complexity of weighted counting of independent sets. In *Parameterized and Exact Computation – 5th International Symposium, IPEC*

2010, Chennai, India, December 13-15, 2010. Proceedings, pages 180–191, 2010. doi:10.1007/978-3-642-17493-3_18.

18 Thore Husfeldt and Nina Taslaman. The exponential time complexity of computing the probability that a graph is connected. In *Parameterized and Exact Computation – 5th International Symposium, IPEC 2010, Chennai, India, December 13-15, 2010. Proceedings*, pages 192–203, 2010. doi:10.1007/978-3-642-17493-3_19.

19 Russel Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

20 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

21 Pieter W. Kasteleyn. The statistics of dimers on a lattice: I. The number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209–1225, 1961. doi:10.1016/0031-8914(61)90063-5.

22 Pieter W. Kasteleyn. Graph Theory and Crystal Physics. In *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, 1967.

23 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011. URL: http://albcom.lsi.upc.edu/ojs/index.php/beatcs/article/view/96.

24 Patrick Scharpfenecker and Jacobo Torán. Solution-graphs of boolean formulas and isomorphism. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:24, 2016. URL: http://eccc.hpi-web.de/report/2016/024.

25 J. Simon. *On Some Central Problems in Computational Complexity.* PhD thesis, Cornell University, 1975.

26 H. N. V. Temperley and Michael E. Fisher. Dimer problem in statistical mechanics – an exact result. *Philosophical Magazine*, 6(68):1478–6435, 1961.

27 Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

28 Leslie G. Valiant. Accidental algorithms. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 509–517, 2006. doi:10.1109/FOCS.2006.7.

29 Leslie G. Valiant. Holographic algorithms. *SIAM Journal on Computing*, 37(5):1565–1594, 2008. doi:10.1137/070682575.

30 Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

31 Klaus W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23(3):325–356, 1986. doi:10.1007/BF00289117.

32 Viktoria Zanko. #P-completeness via many-one reductions. *International Journal of Foundations of Computer Science*, 2(1):77–82, 1991. doi:10.1142/S0129054191000066.

On the Hardness of Partially Dynamic Graph Problems and Connections to Diameter

Søren Dahlgaard*

University of Copenhagen, Denmark
soerend@di.ku.dk

Abstract

Conditional lower bounds for dynamic graph problems has received a great deal of attention in recent years. While many results are now known for the fully-dynamic case and such bounds often imply worst-case bounds for the partially dynamic setting, it seems much more difficult to prove amortized bounds for incremental and decremental algorithms. In this paper we consider partially dynamic versions of three classic problems in graph theory. Based on popular conjectures we show that:

- No algorithm with amortized update time $O(n^{1-\varepsilon})$ exists for incremental or decremental maximum cardinality bipartite matching. This significantly improves on the $O(m^{1/2-\varepsilon})$ bound for sparse graphs of Henzinger et al. [STOC'15] and $O(n^{1/3-\varepsilon})$ bound of Kopelowitz, Pettie and Porat[1]. Our linear bound also appears more natural. In addition, the result we present separates the node-addition model from the edge insertion model, as an algorithm with total update time $O(m\sqrt{n})$ exists for the former by Bosek et al. [FOCS'14].
- No algorithm with amortized update time $O(m^{1-\varepsilon})$ exists for incremental or decremental maximum flow in directed and weighted sparse graphs. No such lower bound was known for partially dynamic maximum flow previously. Furthermore no algorithm with amortized update time $O(n^{1-\varepsilon})$ exists for directed and unweighted graphs or undirected and weighted graphs.
- No algorithm with amortized update time $O(n^{1/2-\varepsilon})$ exists for incremental or decremental $(4/3-\varepsilon')$-approximating the diameter of an unweighted graph. We also show a slightly stronger bound if node additions are allowed. The result is then extended to the static case, where we show that no $O((n\sqrt{m})^{1-\varepsilon})$ algorithm exists. We also extend the result to the case when an additive error is allowed in the approximation. While our bounds are weaker than the already known bounds of Roditty and Vassilevska Williams [STOC'13], it is based on a weaker conjecture of Abboud et al. [STOC'15] and is the first known reduction from the 3SUM and APSP problems to diameter. Showing an equivalence between APSP and diameter is a major open problem in this area (Abboud et al. [SODA'15]), and thus showing even a weak connection in this direction is of interest.

1998 ACM Subject Classification G.2.2 Graph Theory, E.1 Data Structures, F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Conditional lower bounds, Maximum cardinality matching, Diameter in graphs, Hardness in P, Partially dynamic problems, Maximum flow

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.48

* Part of this work was done while the author was visiting Stanford University. Research partly supported by Mikkel Thorup's Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research under the Sapere Aude research career programme.

[1] Kopelowitz et al. showed this result at SODA'16, and after posting their result online it was improved in an online version of the paper by Henzinger et al. Kopelowitz et al. also showed a slightly stronger bound of $O(n^{0.39-\varepsilon})$ if node insertions are allowed.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 48; pp. 48:1–48:14

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

1 Introduction

Arguably one of the most important goals of computer science is to understand the complexity of natural computational problems. For many such problems we know of polynomial time algorithms, but getting matching unconditional lower bounds seem far beyond the scope of our current techniques. Therefore a recent and very active line of research at top-level conferences concerns itself with hardness results in the class **P** [26, 6, 8, 17, 21, 3, 14, 4, 13, 12, 10, 2, 1]. Such results are obtained by reducing from classic problems like 3SUM, APSP and CNF-SAT, for which there exist very popular conjectures about the running time. We call such a hardness result a *conditional lower bound (CLB)* as it is based (conditioned) on the truthfulness of some popular conjecture. The main goal of CLBs is to explain barriers in algorithm development and provide "warning signs" that improving an algorithm for some problem has major and surprising consequences for a classic problem like the ones mentioned above, which researchers have worked on for decades, and trying to do so may be ill-advised.

One particular area that has received a lot of attention from this perspective is dynamic graph problems [29, 26, 6, 8, 17, 21]. In dynamic graph problems we are asked to maintain some property about a graph such as reachability or shortest paths distances as the graph undergoes changes (typically edge insertions and deletions). One may also consider the partially dynamic cases where only edge insertions are allowed (incremental) or edge deletions (decremental) or cases where node insertion and deletion is allowed. Several conditional lower bounds are known for both partially and fully dynamic problems such as shortest paths [29, 17], maximum bipartite matching [6, 17, 21], maximum flow [8], reachability [26, 6, 17], and many more.

1.1 Difficulties of partially dynamic

Most of the research on CLBs for dynamic graph problems has been focused on the fully dynamic case, however such results do not translate well into CLBs for incremental or decremental algorithms. A typical reduction works by 1) building a structured base graph, 2) for each element in some subset of the input perform a series of insertions and queries to decide whether this element is in a possible solution, 3) perform a series of deletions returning the graph to its base state. From a partially dynamic perspective we may use the above procedure to get similar *worst-case* bounds, by keeping track of the data structure state and simulating step 3 by *rolling back* the insertions, however this kills any hope of good amortized bounds. As noted in [6, 17, 21] it seems more difficult to obtain good bounds in this case, and specialized reductions are often needed.

1.2 Bounds under weaker assumptions

While proving higher lower bounds is the main goal of CLBs, a simultaneous goal is to prove similar CLBs under weaker assumptions, thus lending more credibility to the belief that a problem is difficult or even impossible. Several recent papers concerns themselves with this be either replacing a conjecture with a weaker version as done by Abboud et al. in [4] or by showing similar reductions under several conjectures [32, 5, 7, 8, 17]. As an example Abboud, Vassilevska Williams, and Yu [8] showed that 3SUM, APSP and CNF-SAT can all be reduced to the same problem of finding triangles in a node-colored graph and showed several interesting results based on the following conjecture:

▶ **Conjecture 1** ([8]). *At least one of the following is true:*

1. *There is no algorithm for the 3-SUM problem running in $O(n^{2-\varepsilon})$ for any $\varepsilon > 0$.*

2. *There is no algorithm for the APSP problem on weighted graphs running in* $O(n^{3-\varepsilon})$ *for any* $\varepsilon > 0$.
3. *For every* $\delta > 0$ *there is an integer* $k \geq 3$ *such that* k*-SAT on* n *variables and* $O(n)$ *clauses cannot be solved in* $2^{(1-\delta)n}\operatorname{poly}(n)$ *time.*

The third item in Conjecture 1 is what is known as the strong exponential time hypothesis (SETH) [18] and the $O(n)$ bound on the number of clauses follows from the sparsification lemma of Impagliazzo, Paturi, and Zane [19].

1.3 Our results

In this paper we consider three of the perhaps most classic problems in graph theory, namely maximum flow, maximum bipartite matching and diameter in the partially dynamic setting. For maximum flow and maximum bipartite matching we show new, stronger, and more natural conditional lower bounds. For diameter we show a new reduction from Conjecture 1 to both the partially dynamic version of diameter and, perhaps more interestingly, the static case. This is the first known connection from APSP and 3SUM to diameter in graphs and addresses one of the main open problems in the area as stated in [3].

Maximum bipartite matching. In dynamic maximum cardinality bipartite matching we wish to maintain the size of a maximum matching in a dynamic graph G. One can trivially do this in $O(m)$ time by finding an augmenting path. Sankowski [30] gave a fully dynamic algorithm with update time $O(n^{1.495})$ by using fast matrix multiplication. In the incremental setting, one may consider a node-addition version in which the right-hand side of the bipartite graph is given and the left-hand side arrives one node at a time with all its incident edges. In this model Bosek et al. [11] gave an algorithm with total running time of $O(m\sqrt{n})$. From a hardness perspective, Abboud and Vassilevska Williams [6] gave reductions from 3SUM, triangle detection and boolean matrix multiplication to fully-dynamic maximum cardinality bipartite matching. In particular, they showed that a $O(n^{2-\varepsilon})$ algorithm would imply a faster *combinatorial* boolean matrix multiplication algorithm. Their reductions, however, only imply worst-case bounds in the case of partially dynamic algorithms. This was addressed by Kopelowitz, Pettie and Porat [21] who revisited Pătraşcu's reductions from [26] and showed that any $O(n^{1/3-\varepsilon})$ algorithm for incremental MCM would imply a truly subquadratic algorithm for 3SUM. They also showed the same result for $O(n^{0.39-\varepsilon})$ algorithms when node insertions are allowed. Subsequently, in an online version of [17], it was shown how to obtain a CLB of $O(m^{1/2-\varepsilon})$ in sparse graphs by reducing from the online matrix-vector multiplication (OMv) problem.

In this paper we show the following theorem:

▶ **Theorem 2.** *There is no algorithm for solving incremental (or decremental) maximum cardinality bipartite matching with amortized time* $O(n^{1-\varepsilon})$ *per insertion (or deletion) and* $O(n^{2-\varepsilon})$ *time per query unless the OMv conjecture of [17] is false.*

One thing to note about Theorem 2 is that it separates the node-addition model from the edge-insertion model as it implies a total running time of $O(mn^{1-o(1)})$ in contrast to the $O(m\sqrt{n})$ running time of the algorithm from [11]. Furthermore, the reduction used to prove Theorem 2 also rules out any efficient incremental (or decremental) approximation algorithm that works by ruling out the existence of short augmenting paths. Ruling out such paths is a popular way of ensuring a good approximation ratio [25].

Maximum flow. Single-source single-sink maximum flow (*st* Max-Flow) is one of the most classic problems in graph theory. In recent years there have been several breakthrough results for *st* maximum flow using the powerful tools of Laplacian system solvers and interior point methods [24, 31, 20, 22]. These algorithms seem to take near-line time in practice, and the limits of our current analysis might be the bottleneck in proving such upper bounds. Proving super-linear conditional lower bounds for this problem may thus be difficult if not impossible. Therefore, Abboud et al. [8] considered different variants of the problem such as single-source maximum flow and *ST* maximum flow. They also showed that any algorithm solving the fully-dynamic version of *st* maximum flow with amortized update and query time $O(n^{1-\varepsilon})$ for any $\varepsilon > 0$ would refute Conjecture 1. Finally, we note that it is possible to modify the $m^{1-o(1)}$ CLB for fully dynamic #SSR of Abboud and Vassilevska Williams [6] to obtain a $m^{1-o(1)}$ CLB for fully-dynamic *st* max-flow in sparse graphs.

In this paper we show that even in the incremental and decremental case *st* maximum flow exhibit the same kind of CLB, but based solely on SETH. This is summarized in the following theorem:

▶ **Theorem 3.** *There is no algorithm for solving incremental (or decremental) max st flow on a weighted and directed graph with n nodes and $\tilde{O}(n)$ edges with amortized time $O(m^{1-\varepsilon})$ per operation for any $\varepsilon > 0$ unless SETH is false.*

Our bound shows that we cannot hope to get incremental maximum flow in offline time as is the case for other problems. We note that the above result only holds for directed and weighted graphs. We show similar results for other types of graphs:

▶ **Theorem 4.** *There is no algorithm for solving incremental (or decremental) max st flow on unweighted directed graphs or weighted undirected graphs on n nodes with amortized time $O(n^{1-\varepsilon})$ per operation for any $\varepsilon > 0$ unless the OMv conjecture is false.*

This result follows directly from Theorem 2 by using textbook reductions from maximum bipartite matching to directed flow (see e.g. [15]) and from directed flow to undirected flow (see e.g. [23]).

Diameter. The diameter problem asks us to compute the longest shortest-path distance in a graph G. Efficiently computing or approximating the diameter is a basic problem in graphs [3, 9, 14, 16, 28]. One can trivially compute the diameter in the same time as computing APSP, however in general no better algorithm is known. It remains a major open problem whether a reduction exists in the other direction [3] – that is, can we compute all distances in the same time as the longest? One can, however, approximate the diameter faster. Roditty and Vassilevska Williams [28] showed how to compute a 3/2-approximation in time $\tilde{O}(m\sqrt{n})$ randomized, and Chechik et al. [14] showed how to obtain the same guarantee deterministically in time $\tilde{O}(\min(m^{3/2}, mn^{2/3}))$. More recently, it was shown by Cairo, Grossi and Rizzi [13] how to obtain a $(2 - \frac{1}{2^k})$-approximation in time $\tilde{O}(mn^{\frac{1}{k+1}})$. From a hardness perspective it is known that any algorithm able to distinguish between diameter 3 and 2 in time $O(m^{2-\varepsilon})$ for sparse graphs would refute SETH [28]. Chechik et al. [14] showed that approximating within a $4/3 - \varepsilon$ factor with additive error $\beta = O(m^{\delta})$ in time $O(m^{2-2\delta-\varepsilon'})$ for sparse graphs would also refute SETH, and this bound was improved in [13] to rule out any $3/2 - \varepsilon$ approximation with the same additive error and time bounds based on SETH (also for sparse graphs). From the perspective of dynamic algorithms Abboud and Vassilevska Williams [6] showed that any algorithm for $4/3 - \varepsilon$-approximating the diameter in a fully dynamic graph with amortized update time $O(m^{2-\varepsilon'})$ would refute SETH. We also

note, that the above static reductions rules out any $O(m^{1-\varepsilon'})$ amortized update time for incremental algorithms.

We note that all the reductions mentioned above are based on SETH. Similar to the work of [8, 4] we seek to replace this assumption by a weaker one. In this paper we show the first reduction from 3SUM and APSP to the diameter problem. That is, we show that a fast algorithm for approximating the diameter implies a faster algorithm for the APSP and 3SUM problems. The bounds we achieve are not as strong as the known bounds based on SETH [28, 14, 13], however they are based on a weaker conjecture and hold even if SETH turns out to be false, thus giving more credibility to the difficulty of the problem. For the partially dynamic case we show the following theorem:

▶ **Theorem 5.** *There exists no incremental (or decremental) algorithm that approximates the diameter of an unweighted graph within a factor of $4/3-\varepsilon$ running in amortized time $O(n^{1/2-\varepsilon'})$ for any $\varepsilon,\varepsilon'>0$ unless Conjecture 1 is false. Furthermore, if we allow node insertions in the incremental case the bound is $O(n^{0.618-\varepsilon'})$.*

In order to achieve the result for node insertions, we use the technique of Kopelowitz et al. [21] leveraging rollback with our standard incremental bound. By doing this we obtain a graph with fewer nodes and thus a better bound. More interestingly, we are able to generalize our results from the incremental case to the following result for static graphs:

▶ **Theorem 6.** *There exists no static $4/3-\varepsilon$ approximation to the diameter on unweighted graphs running in $O((n\sqrt{m})^{1-\varepsilon'})$ time for any $\varepsilon,\varepsilon'>0$ and any number of edges m unless Conjecture 1 is false.*

As mentioned, this is the first known reduction from APSP to diameter and shows at least some weak connection in this direction. An interesting property of Theorem 6 is that it holds *for any* m as a function of n and thus an algorithm need not exist *for all* m. As a corollary of Theorem 6 we see that no algorithm can $(4/3-\varepsilon)$-approximate the diameter of static unweighted graph in time $O(n^{2-\varepsilon'})$ for any ε,ε' unless Conjecture 1 is false. This is reminiscent of the bounds from [28, 14, 13], however not quite as strong as it does not hold for sparse graphs, for which we get a bound of $O(m^{3/2-\varepsilon'})$.

Similar to [14, 13] we also extend the above bound to the case of $(4/3-\varepsilon)$-approximations with additive error $O(m^\delta)$. We show the following

▶ **Corollary 7.** *There exists no static $4/3-\varepsilon$ approximation with additive error $O(m^\delta)$ with running time $O(m^{\frac{3}{2}(1-\delta)-\varepsilon'})$ or incremental/decremental algorithm with amortized time $O(m^{\frac{1}{2}-\frac{3\delta}{2}-\varepsilon'})$ for any $\varepsilon,\varepsilon'>0$ unless Conjecture 1 is false.*

1.4 A note on the decremental results and preprocessing

We will in general only describe the reductions in the incremental case and note that the decremental results are obtained by removing the edges in the reverse order of insertions. This requires an assumption on the beginning graph, and we will thus assume any suitable graph on $\tilde{O}(n)$ edges in the sparse case and the complete graph in the dense case.

Furthermore, we do not assume that any of the algorithms are allowed to preprocess the graph. It is often an assumption in the design of amortized partially dynamic algorithms that one starts with the empty (or complete) graph in order for the analysis to work. Thus, our results hold for this case.

2 Preliminaries

Notation. Throughout the paper we assume that matrices are boolean. Thus the output of a vector-matrix-vector multiplication will always be a single bit. We use $[n]$ to denote the set $\{0, \ldots, n-1\}$.

Online vector-matrix-vector multiplication. We will consider the online vector-matrix-vector multiplication problem of [17]:

▶ **Definition 8** (OuMv problem [17]). Let M be a binary $n \times n$ matrix than can be preprocessed. After preprocessing n vector pairs $(u^1, v^1), \ldots, (u^n, v^n)$ arrive one at a time and the task is to compute $(u^i)^T M v^i$ before being presented with the $i+1$th vector pair for every i.

In [17] they showed that the OMv problem can be reduced to the OuMv problem. They also came up with the following conjecture:

▶ **Conjecture 9** ([17]). *There is no algorithm for the OMv problem (and thus the OuMv problem) running in time $O(n^{3-\varepsilon})$ for any $\varepsilon > 0$.*

Triangle collection. We will also consider the triangle collection problem of [8]:

▶ **Definition 10** (Triangle collection [8]). Given a node-colored graph G, is it true that for every triplet of colors a, b, c there exists a triangle (u, v, w) in G where u has color a, v has color b and w has color c?

In fact, we will consider the more structured triangle collection* (TC*) problem which they also used in [8]

▶ **Definition 11** (Triangle collection* [8]). Let n, Δ, p be parameters and let G be an undirected node-colored tripartite graph with partitions A, B, C. Let G be any graph with the following structure:

- Each partition has its own n colors and we denote these by the numbers of $[n]$ for each partition.
- A contains nodes of the form a^i_j, where $i \in [n]$ is the color of the node and $j \in [\Delta]$.
- B and C contains nodes of the form $b^i_{j,x}$ and $c^i_{j,x}$ where $i \in [n]$ is the color of the node and $j \in [\Delta], x \in [p]$.

And the edges of G are as follows:

- For each $i, i' \in [n]$ and $j \in [\Delta]$ there is an edge from a^i_j to $b^{i'}_{j,x}$ for *exactly one* x. Similarly there is an edge from a^i_j to $c^{i'}_{j,y}$ for exactly one y (note that y and x need not be the same for the same j and i').
- There may be an edge between nodes $b^i_{j,x}$ and $c^{i'}_{j',y}$ *only if* $j = j'$.

We ask the following question: Does there exist a triple of colors (one color per partition) such that G does not contain a triangle with these colors?

In [8] it was shown that this problem does not have a truly subcubic algorithm unless Conjecture 1 is false.

It will be important that the reductions from these problems to TC* hold even when Δ and p are bounded by polylog(n).

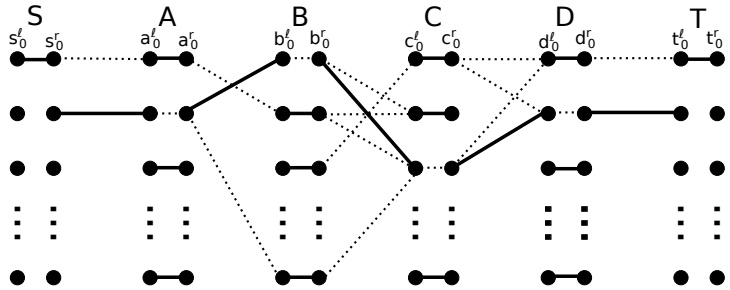


Figure 1 Reduction to incremental maximum matching.

3 Incremental maximum matching

We will reduce from the OuMv problem of Definition 8. Observe that the OuMv problem is equivalent to the following statement: For each vector pair u^i, v^i determine whether indices j, k exist, such that $u^i_j = v^i_k = M_{jk} = 1$. In order to model this as an incremental maximum matching problem we construct the following graph: Create 6 copies of $2n$ nodes and name these S, A, B, C, D, T. Partition A into n pairs of nodes $a^\ell_1, a^r_1, \ldots, a^\ell_n, a^r_n$. Do the same for S, B, C, D, T. Add the edges (a^ℓ_i, a^r_i) for each i and do the same for B, C, D. Now for each i, j add the edge (b^r_i, c^ℓ_j) if $M_{ij} = 1$. Observe that this graph has a unique maximum matching each (ℓ, r) pair. Observe also that the graph is bipartite. Now we do the following n phases – one for each u^i, v^i vector pair.

1. For each j such that $u^i_j = 1$ add the edge (a^r_i, b^ℓ_j).
2. For each j such that $v^i_j = 1$ add the edge (c^r_j, d^ℓ_i).
3. Add the edges (s^r_i, a^ℓ_i) and (d^r_i, t^ℓ_i).
4. Query the size of a maximum matching.
5. Add the edges (s^ℓ_i, s^r_i) and (t^ℓ_i, t^r_i).

This is illustrated in Figure 1.

▶ **Lemma 12.** *Let the setting be as above and let the phases be numbered* $0, 1, \ldots, n-1$. *Then the size of the maximum matching during the* i*th phase is exactly* $4n + 2i + 1$ *if the resulting vector-matrix-vector product is* 1 *and* $4n + 2i$ *otherwise.*

Proof. Note that prior to any of the i phases the size of the maximum matching is exactly $4n + 2i$, which is also a perfect matching of the graph induced by the edges. To see this observe that each $s^\ell_0, \ldots, s^\ell_{i-1}$ must be matched to its corresponding $s^r_0, \ldots, s^r_{i-1}$, and this is the only edge incident to the ℓ-nodes. As a consequence of this, each a^ℓ_j must be matched with a^r_j, and so on.

Now consider the ith phase. Adding any edge (a^r_i, b^ℓ_j) or (c^r_j, d^ℓ_i) cannot increase the size of the maximum matching, as the size of the subgraph induced by the edges of the graph does not increase – i.e. all nodes with edges incident to them are already matched.

Assume that adding the edges (s^r_i, a^ℓ_i) and (t^ℓ_i, d^r_i) increases the matching. The matching can increase by at most 1, as only two more nodes can be matched. Furthermore the matching must now contain edges as follows

$$(s^r_i, a^\ell_i), (a^r_i, b^\ell_j), (b^r_j, c^\ell_k), (c^r_k, d^\ell_x), (d^r_x, t^\ell_y) \ .$$

Now observe that each t^ℓ_y for $y < i$ *must* be matched to t^r_y, as the right nodes have no other incident edges and all nodes have to be matched for the size of the matching to increase.

Thus we must have $y = i$ in the list above, but this means that we have exactly found a pair j, k such that $u_j^i = v_k^i = M_{jk} = 1$ and the vector-matrix-vector product is thus 1.

Conversely, assume that the vector-matrix-vector product is 1, then such an index pair j, k must exist and we can find the following matching of size $4n + 2i + 1$: Match the edges

$$(s_i^r, a_i^\ell), (a_i^r, b_j^\ell), (b_j^r, c_k^\ell), (c_k^r, d_i^\ell), (d_i^r, t_i^\ell) \ .$$

For all $x < i$ add the edges (s_x^ℓ, s_x^r) and (t_x^ℓ, t_x^r) to the matching. For all $x \neq i$ add the edges (a_x^ℓ, a_x^r) and (d_x^ℓ, d_x^r) to the matching. And for all $x \neq j$ and $y \neq k$ add the edges (b_x^ℓ, b_x^r) and (c_y^ℓ, c_y^r) to the matching. This matches all nodes incident to an edge and has size $4n + 2i + 1$. This is also exactly the matching illustrated in Figure 1 for $i = 1$. ◀

It follows from Lemma 12 that we can solve the OuMv problem correctly via this reduction. The reduction creates a graph with $O(n)$ nodes and $O(n^2)$ edges. We perform $O(n^2)$ insertions and $O(n)$ queries giving the result in Theorem 2

4 Maximum flow

In order to show Theorem 3 we will use a similar graph construction as have been used numerous times before [27, 28, 14, 6]: First partition the variables of the SAT problem into two groups A and B of $n/2$ variables each. For each possible assignment to the variables in A we create a node in our graph G (and likewise for B). Furthermore, for each clause of the SAT formula, we create a node as well. We denote the corresponding sets of nodes by A, B, C. Set $N = 2^{n/2} = |A| = |B|$. For each pair of nodes $a \in A, c \in C$ we add the directed edge (a, c) with capacity N if the partial assignment a *does not* satisfy the clause c. Similarly, for each pair of nodes $b \in B, c \in C$ we add the directed edge (c, b) with capacity 1 if b does not satisfy c. Finally we add two nodes s, t and add edges (b, t) with capacity 1 for each $b \in B$.

We now continue in phases with a phase for each $a \in A$. Denote these nodes by $a_1, a_2, \ldots, a_N$:

1. Add the edge (s, a_i) with capacity N.
2. Query the maximum flow between s and t.
3. Add the edge ("shortcut") (a_i, t) with capacity N.

▶ **Lemma 13.** *Let the setup be as described above. If the st flow returned during any of the i phases is $< i \cdot N$, then the SAT formula is satisfiable. Otherwise the formula is not satisfiable.*

Proof. Observe, that prior to the ith phase, the flow is exactly $(i-1) \cdot N$, as we can use the paths $(s, a_j), (a_j, t)$ for each $j < i$, which has capacity N and exactly $(i-1) \cdot N$ flow leaves s.

Now assume that the partial formula corresponding to a_i can be completed to a satisfying assignment. In this case, there must be some node $b \in B$, for which there is no path from a_i to b. This follows because such a path has to go through a node $c \in C$, but then both a_i and b do not satisfy the clause c, which is a contradiction. However, the only way to send flow from a_i to t is through the nodes $b \in B$ and thus it is not possible to send all N units of flow from a_i to t.

Now assume that the flow is $< i \cdot N$, then there must be some $b \in B$ such that there is no path from a_i to b. Otherwise, we could route N units of flow from a_i to t via the nodes of B and the remaining $(i-1) \cdot N$ units through the "shortcuts". It now follows that a_i and b together satisfy all clauses (otherwise there would be a path) and thus the CNF formula is satisfiable.

Since this is true for all of the i phases, the statement of the lemma follows. ◀

As a consequence of Lemma 13 we may use the above procedure to solve the given SAT problem. By the sparsification lemma of [19] it follows that we can assume the graph has $O(N)$ nodes and $\tilde{O}(N)$ edges and we perform a total of $\tilde{O}(N)$ insertions and queries. The result of Theorem 3 thus follows directly.

5 Diameter

In this section we show how to obtain conditional lower bounds for the problem of approximating the diameter of an unweighted graph within a factor of $4/3 - \varepsilon$.

5.1 A graph construction

We will first describe the graph structure we use.

▶ **Definition 14.** Let G be an instance of the TC* problem as defined above. We will define the graph $H_{\gamma,k}(G)$. The idea is that $H_{\gamma,k}(G)$ "corresponds" to the colors $\{kn^\gamma, \dots, (k+1)n^\gamma - 1\}$ of A. Thus k is a number in $[n^{1-\gamma}]$. The nodes of this graph are as follows:

- The nodes B and C of G.
- For each color $i \in \{kn^\gamma, (k+1)n^\gamma - 1\}$ of A we add the nodes $a^i_0, \dots, a^i_{n-1}$ and $t^i_0, \dots, t^i_{n-1}$.
- We also add several special nodes: A "master node" u, n^γ "skip nodes" v_i and three "connector nodes" w_1, w_2, w_3.

For a color $i \in \{kn^\gamma, (k+1)n^\gamma - 1\}$ we denote the nodes $a^i_0, \dots a^i_{n-1}$ by A_i and the collection of all A_is by A. We do the same for T_i and T.

The edges of $H_{\gamma,k}(G)$ are as follows:

- Add the edges between B and C in G.
- Connect the node w_1 to each node of A and w_2.
- Connect w_2 to each node of B and C as well as w_3 and the master node u.
- Connect w_3 to each node of T.
- Connect u to all nodes v_i.
- For each $i \in \{kn^\gamma, (k+1)n^\gamma - 1\}$ do as follows:
  - Connect v_i to all nodes of $T \setminus T_i$ and to all nodes of A_i.
  - For each $i' \in [n]$ and each edge $(a^i_j, b^{i'}_{j,x}) \in G$ add the edge $(a^i_{i'}, b^{i'}_{j,x})$.
  - For each $i' \in [n]$ and each edge $(a^i_j, c^{i'}_{j,x}) \in G$ add the edge $(c^{i'}_{j,x}, t^i_{i'})$.

An overview of the graph $H_{\gamma,k}(G)$ is illustrated in Figure 2 and a more detailed view in Figure 3.

The idea is that length three paths between A_i and T_i correspond to triangles in G containing the color i of A. Each of the n nodes in A_i thus correspond to picking a color from B and each of the n nodes in T_i correspond to picking a color from C. If two such nodes don't have a length three path there is no triangle in G of the corresponding triplet of colors. In this case the connector nodes ensure that there is a length four path between the nodes. The master and skip nodes ensure that all other nodes have distance at most 3. This is captured by the following lemma:

▶ **Lemma 15.** *Let G be an instance to the TC* problem and let $H_{\gamma,k}(G)$ be as defined above. Let $i \in \{kn^\gamma, (k+1)n^\gamma - 1\}$ be a color of A and let $\alpha, \beta \in [n]$ be colors of B and C respectively. Then the distance from a^i_α to t^i_β in $H_{\gamma,k}(G)$ is 3 if the colors i, α, β have a triangle in G and 4 otherwise.*

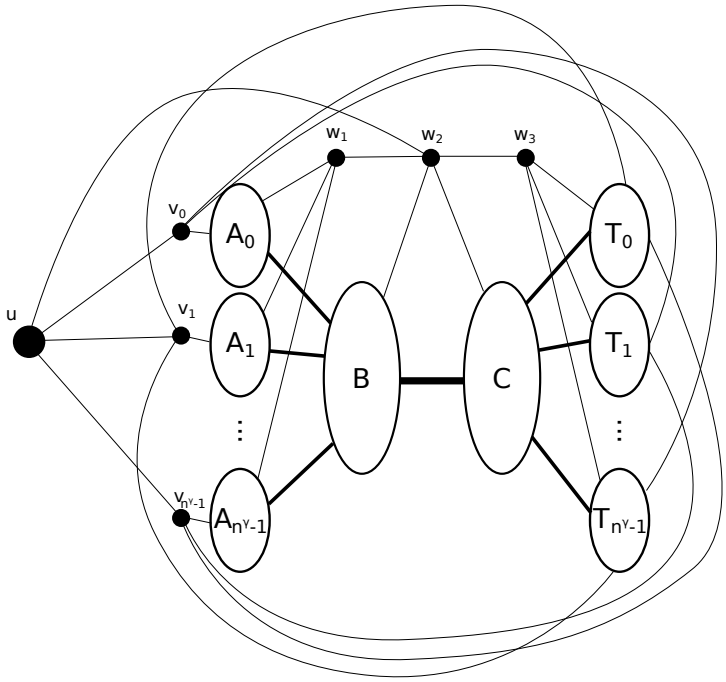


Figure 2 Diameter structure.

Proof. Assume first that there is a triangle $a^i_j, b^\alpha_{j,x}, c^\beta_{j,y}$ for some j in G (note that such a triangle can only occur if j is the same for all the three nodes). In this case there is a path $a^i_\alpha, b^\alpha_{j,x}, c^\beta_{j,y}, t^i_\beta$ in $H_{\gamma,k}(G)$ and thus the distance is at most 3. Observe also, that no node is connected to both A_i and T_i and thus the distance is strictly greater than 2.

Now assume that the distance from a^i_α to t^i_β is 3. Such a path has to go from A_i to B to C to T_i as any node w_ℓ, v_ℓ or u either has distance 3 to one of a^i_α or t^i_β or it has distance 2 to both of them. Now consider a shortest path $a^i_\alpha, b, c, t^i_\beta$, where b and c are the nodes of B and C on this path. Clearly the node b must have color α in G as it would not have an edge to a^i_α otherwise, and similarly c must have color β in G. Thus the path consists of nodes $a^i_\alpha, b^\alpha_{j,x}, c^\beta_{j',y}, t^i_\beta$. Since no edge in G goes between nodes with different j-values we must have $j' = j$. It is now clear that the edge $(a^i_\alpha, b^\alpha_{j,x})$ corresponds to the edge $(a^i_j, b^\alpha_{j,x})$ in G and the edge $(c^\beta_{j,y}, t^i_\beta)$ corresponds to the edge $(a^i_j, c^\beta_{j,y})$ in G. Thus, these three nodes form a triangle of the correct color triple in G. ◀

Furthermore it is easy to see that the longest distance in $H_{\gamma,k}(G)$ is at most 4, thus the diameter is 4 exactly when one of the corresponding color triplets do not have a triangle in G and 3 otherwise.

5.2 Dynamic

We will first consider the problem without node additions. For simplicity we only consider the incremental case and note that the decremental case follows by deleting edges until we obtain the same graph[2].

[2] Under the assumption that the algorithm starts with some suitable graph

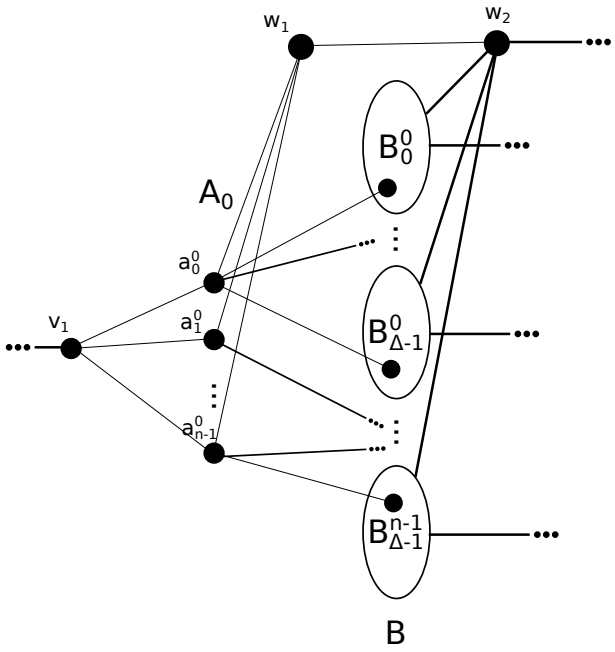


Figure 3 Diameter structure.

Given an instance to the TC* problem we create the graph $H_{1,0}(G)$ (that is, the graph representing all colors of A). This graph is created by adding edges incrementally and has $\tilde{O}(n^2)$ nodes and edges. It follows that an edge insertion must take $n^{1/2-o(1)}$ time unless Conjecture 1 is false.

Next, we consider the problem with node additions. It was shown in [21] that if we allow node additions in the problem of incremental maximum matching, it is possible to show stronger lower bounds by leveraging the amortized running time with the widely used rollback technique. We here apply the same argument to the problem of incremental diameter approximation.

The goal is again to construct (a subgraph of) $H_{1,0}(G)$ but we do not start with all nodes in the graph. We will assume that the amortized running time of an insert operation is n^α for some α. The goal is to get a bound on α by expressing the total running time in terms of α and using the assumption on running time for TC*. We let $\hat{n}$ denote the current number of nodes in the graph G. We continue as follows:

1. Insert all nodes of B and C into the dynamic graph. Also insert the nodes w_1, w_2, w_3 and u. We also insert all the edges induced by these nodes in $H_{1,0}(G)$ into the graph.
2. For each color $i \in [n]$ of A we do a phase:
   - We insert the nodes of A_i, T_i, v_i into the dynamic graph and all the edges induced by these nodes and the current state of the dynamic graph in $H_{1,0}(G)$.
   - Query the diameter of the graph.
   - Assume we inserted k edges+nodes in this phase. If the total running time of all these insertions was greater than $2k\hat{n}^\alpha$ we keep the nodes in the graph. Otherwise we rollback all operations of this phase.

We answer the question of the TC* problem according to whether the diameter was 3 all the time or not similar to the proof of the case without node additions.

The goal is now to bound α by using the method of [21]. We will do this by carefully

counting the number of "amortized credit units" the data structure has and using this to bound the total number of nodes added to the graph (i.e. not rolled back).

Observe that after the first step, we have added $\tilde{O}(n^2)$ edges to the graph and $\tilde{O}(n)$ nodes. Thus the data structure has at most $\tilde{O}(n^{2+\alpha})$ credit at this point (this happens if almost all operations were $O(1)$). Now consider the total time spent by the algorithm. This can be bounded by $\tilde{O}(n^2 \cdot N^\alpha)$ where N is the number of nodes at the end of all phases. This is the case since $N \geq N_0$, where $N_0 = \tilde{O}(n)$ is the number of nodes after the first step and there are at most $\tilde{O}(n^2)$ total operations. Note that this would not be the case if we did not have a bound on the cost of the rolled back operations, but we only rollback the cheap operations, so this is okay. We wish to express N in terms of n and α in order to express the total running time in terms of these.

Observe, that every time we keep the added nodes in the graph, the data structure spent at least twice the amortized cost. Since we started out with $\tilde{O}(n^{2+\alpha})$ credit it must be true that

$$\sum_{i=N_0}^{N} i^\alpha \leq \text{cost of non-rollbacked operations} = \tilde{O}(n^{2+\alpha}) \ .$$

The worst case is if N is polynomially larger than N_0, and thus $\sum_{i=N_0}^{N} i^\alpha = \Omega(N^{1+\alpha})$. It follows that $N = \tilde{O}(n^{\frac{2+\alpha}{1+\alpha}})$. Thus the total running time is $\tilde{O}(n^2 \cdot n^{\frac{2+\alpha}{1+\alpha}\alpha})$. Now, by Conjecture 1 we must have $\frac{2+\alpha}{1+\alpha}\alpha = 1 - o(1)$. Solving this for α gives $\alpha = \frac{\sqrt{5}-1}{2} < 0.618$.

5.3 Static

Proof of Theorem 6. Let G be an instance of the TC* problem with parameters n, Δ, p with Δ and p bounded by $\tilde{O}(1)$ and $m = \tilde{O}(n^2)$ as in [8].

For a parameter $0 < \gamma \leq 1$ we create the graphs $H_{\gamma,0}(G), \ldots, H_{\gamma,n^{1-\gamma}-1}(G)$ and solve the diameter problem on these graphs up to a $4/3 - \varepsilon$ approximation. This is sufficient to distinguish between diameters 4 and 3 in all of the graphs. Now, if the diameter is 4 in just one of the graph we answer that there exists a triplet of colors such that there is no triangle in G. This follows from Lemma 15.

We note that the graphs $H_{\gamma,k}(G)$ each have $N = \tilde{O}(n^{1+\gamma})$ nodes and $M = \tilde{O}(n^2)$ edges. Assume now that that any algorithm approximating the diameter within a factor of $4/3 - \varepsilon$ in time $O(N\sqrt{M}^{1-\varepsilon'}) = O(n^{2+\gamma-\varepsilon'})$ for any $\varepsilon, \varepsilon' > 0$ exists. Since we create $n^{1-\gamma}$ instances of the problem this would imply an $O(n^{3-\varepsilon''})$ algorithm for the TC* problem for some $\varepsilon'' > 0$. ◀

5.4 Additive error

To see Corollary 7 we fix m^α and consider TC* on a graph G with N nodes and $M = \tilde{O}(N^2)$ edges such that $M = m^{1-\alpha}$. We then create $H_{1,0}(G)$ and subdivide each edge into m^α nodes. This graph now has m nodes and edges and any algorithm solving $4/3 - \varepsilon$ diameter with additive error $O(m^\alpha)$ in time $M^{3/2-\varepsilon'} = m^{\frac{3}{2}(1-\alpha)-\varepsilon''}$ time thus violates Conjecture 1.

Acknowledgements. I would like to thank Amir Abboud and Virginia Vassilevska Williams for helpful discussions and observations. I would also like to thank an anonymous referee for pointing out the reduction from incremental matching to undirected flow.

References

1. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In *Proc. 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 98–117, 2015.
2. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *Proc. 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 59–78, 2015.
3. Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, APSP and diameter. In *Proc. 26th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 1681–1697, 2015.
4. Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends or: A polylog shaved is a lower bound made. In *Proc. 48th ACM Symposium on Theory of Computing (STOC)*, 2016. To appear.
5. Amir Abboud and Kevin Lewi. Exact weight subgraphs and the k-sum conjecture. In *Proc. 40th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 1–12, 2013.
6. Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 434–443, 2014.
7. Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 39–51, 2014.
8. Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proc. 47th ACM Symposium on Theory of Computing (STOC)*, pages 41–50, 2015.
9. Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
10. Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proc. 47th ACM Symposium on Theory of Computing (STOC)*, pages 51–58, 2015.
11. Bartlomiej Bosek, Dariusz Leniowski, Piotr Sankowski, and Anna Zych. Online bipartite matching in offline time. In *Proc. 55th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 384–393, 2014.
12. Karl Bringmann and Marvin Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *Proc. 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 79–97, 2015.
13. Massimo Cairo, Roberto Grossi, and Romeo Rizzi. New bounds for approximating extremal distances in undirected graphs. In *Proc. 27th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 363–376, 2016.
14. Shiri Chechik, Daniel H. Larkin, Liam Roditty, Grant Schoenebeck, Robert Endre Tarjan, and Virginia Vassilevska Williams. Better approximation algorithms for the graph diameter. In *Proc. 25th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 1041–1052, 2014.
15. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
16. Marek Cygan, Harold N. Gabow, and Piotr Sankowski. Algorithmic applications of baur-strassen's theorem: Shortest cycles, diameter, and matchings. *Journal of the ACM*, 62(4):28, 2015.

17 Monika Henzinger, Sebastian Krinninger, Danupon Nanongkai, and Thatchaphol Saranurak. Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture. In *Proc. 47th ACM Symposium on Theory of Computing (STOC)*, pages 21–30, 2015.

18 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

19 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

20 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proc. 25th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 217–226, 2014.

21 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3sum conjecture. In *Proc. 27th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 1272–1287, 2016.

22 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(vrank) iterations and faster algorithms for maximum flow. In *Proc. 55th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2014.

23 Aleksander Madry. *From Graphs to Matrices, and Back: New Techniques for Graph Algorithms*. PhD thesis, Massachusetts Institute of Technology, 6 2011.

24 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *Proc. 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 253–262, 2013.

25 Ofer Neiman and Shay Solomon. Simple deterministic algorithms for fully dynamic maximal matching. In *Proc. 45th ACM Symposium on Theory of Computing (STOC)*, pages 745–754, 2013.

26 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proc. 42nd ACM Symposium on Theory of Computing (STOC)*, pages 603–610, 2010.

27 Mihai Pătraşcu and Ryan Williams. On the possibility of faster SAT algorithms. In *Proc. 21st ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 1065–1075, 2010.

28 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. 45th ACM Symposium on Theory of Computing (STOC)*, pages 515–524, 2013.

29 Liam Roditty and Uri Zwick. On dynamic shortest paths problems. *Algorithmica*, 61(2):389–401, 2011. See also ESA'04.

30 Piotr Sankowski. Faster dynamic matchings and vertex connectivity. In *Proc. 18th ACM/SIAM Symposium on Discrete Algorithms (SODA)*, pages 118–126, 2007.

31 Jonah Sherman. Nearly maximum flows in nearly linear time. In *Proc. 54th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 263–269, 2013.

32 Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proc. 41st ACM Symposium on Theory of Computing (STOC)*, pages 455–464, 2009.

Incremental 2-Edge-Connectivity in Directed Graphs*

Loukas Georgiadis[1], Giuseppe F. Italiano[2], and Nikos Parotsidis[3]

1 **University of Ioannina, Ioannina, Greece**
loukas@cs.uoi.gr
2 **University of Rome Tor Vergata, Rome, Italy**
giuseppe.italiano@uniroma2.it
3 **University of Rome Tor Vergata, Rome, Italy**
nikos.parotsidis@uniroma2.it

Abstract

We present an algorithm that can update the 2-edge-connected blocks of a directed graph with n vertices through a sequence of m edge insertions in a total of $O(mn)$ time. After each insertion, we can answer the following queries in asymptotically optimal time:

- Test in constant time if two query vertices v and w are 2-edge-connected. Moreover, if v and w are not 2-edge-connected, we can produce in constant time a "witness" of this property, by exhibiting an edge that is contained in all paths from v to w or in all paths from w to v.
- Report in $O(n)$ time all the 2-edge-connected blocks of G.

This is the first dynamic algorithm for 2-connectivity problems on directed graphs, and it matches the best known bounds for simpler problems, such as incremental transitive closure.

1998 ACM Subject Classification E.1 Graphs and networks – Trees, F.2.2 Computations on discrete structures, G.2.2 Graph algorithms – Trees

Keywords and phrases 2-edge connectivity on directed graphs; dynamic graph algorithms; incremental algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.49

1 Introduction

A dynamic graph algorithm aims at updating efficiently the solution of a problem after an update, such as an edge insertion or an edge deletion, faster than recomputing it from scratch. A problem is said to be *fully dynamic* if the update operations include both insertions and deletions of edges, and it is said to be *partially dynamic* if only one type of update, either insertions or deletions, is allowed. More specifically, a problem is said to be *incremental* (resp., *decremental*) if only insertions (resp., deletions) are allowed. In this paper, we present new incremental algorithms for 2-edge connectivity problems on directed graphs (digraphs). Before defining the problem, we first review some definitions. Let $G = (V, E)$ be a digraph. G is *strongly connected* if there is a directed path from each vertex to every other vertex. The *strongly connected components* (in short *SCC's*) of G are its maximal strongly connected subgraphs. Vertices $u, v \in V$ are *strongly connected* if they are in the same SCC of G. An edge of G is a *strong bridge* if its removal increases the number of SCC's. Let G be strongly connected: G is 2-*edge-connected* if it has no strong bridges. The 2-*edge-connected components* of G are its maximal 2-edge-connected subgraphs. Vertices $u, v \in V$ are said to

* A full version of the paper is available at http://arxiv.org/abs/1607.07073.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 49; pp. 49:1–49:15

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

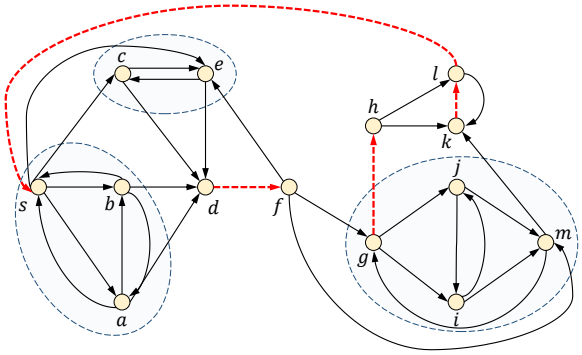


Figure 1 The 2-edge-connected blocks of a digraph G. Strong bridges of G are shown red and dashed. (Better viewed in color.)

be 2-*edge-connected*, denoted by $u \leftrightarrow_{2e} v$, if there are two edge-disjoint directed paths from u to v and two edge-disjoint directed paths from v to u. A 2-*edge-connected block* of a digraph $G = (V, E)$ is a maximal subset $B \subseteq V$ such that $u \leftrightarrow_{2e} v$ for all $u, v \in B$ (see Figure 1).

We remark that in digraphs 2-connectivity has a much richer and more complicated structure than in undirected graphs. To see this, observe that, while in undirected graphs blocks are exactly the same as components, in digraphs there is a substantial difference between those two notions. In particular, two vertices that are 2-edge-connected (i.e., in the same 2-edge-connected block) may lie in different 2-edge-connected components (e.g., vertices i and j in Figure 1, each of them being in a 2-edge-connected component by itself). As a result, 2-connectivity problems on digraphs appear to be much harder than on undirected graphs. For undirected graphs it has been known for over 40 years how to compute 2-edge- and 2-vertex- connected components in linear time [33]. For digraphs, however, only $O(mn)$ algorithms were known (see e.g., [25, 26, 28, 30]). It was shown only recently how to compute the 2-edge- and 2-vertex- connected blocks in linear time [13, 14], and the best current bound for computing the 2-edge- and the 2-vertex- connected components is $O(n^2)$ [18].

Our Results. We initiate the study of the dynamic maintenance of 2-edge-connectivity relationships in directed graphs. We present an algorithm that can update the 2-edge-connected blocks of a digraph G with n vertices through a sequence of m edge insertions in a total of $O(mn)$ time. After each insertion, we can answer the following queries in asymptotically optimal time:

- Test in constant time if two query vertices v and w are 2-edge-connected. Moreover, if v and w are not 2-edge-connected, we can produce in constant time a "witness" of this property, by exhibiting an edge that is contained in all paths from v to w or in all paths from w to v.
- Report in $O(n)$ time all the 2-edge-connected blocks of G.

Ours is the first dynamic algorithm for 2-connectivity problems on digraphs, and it matches the best known bounds for simpler problems, such as incremental transitive closure [23]. This is a substantial improvement over the $O(m^2)$ simple-minded algorithm, which recomputes the 2-edge-connected blocks from scratch after each edge insertion.

Related Work. Many efficient algorithms for several dynamic graph problems have been proposed in the literature, including dynamic connectivity [20, 22, 31, 37], minimum spanning trees [8, 11, 21, 22], edge/vertex connectivity [8, 22] on undirected graphs, and transitive closure [7, 19, 27] and shortest paths [6, 27, 38] on digraphs. Once again, dynamic problems on digraphs appear to be harder than on undirected graphs. Indeed, most of the dynamic algorithms on undirected graphs have polylog update bounds, while dynamic algorithms on digraphs have higher polynomial update bounds. The hardness of dynamic algorithms on digraphs has been recently supported also by conditional lower bounds [1].

Our Techniques. Known algorithms for computing the 2-edge-connected blocks of a digraph G [13, 16] hinge on properties that seem very difficult to dynamize. The algorithm in [13] uses very complicated data structures based on 2-level auxiliary graphs. The loop nesting forests used in [16] depends heavily on an underlying dfs tree of the digraph, and the incremental maintenance of dfs trees on general digraphs is still an open problem (incremental algorithms are known only for the special case of DAGs [10]). Despite those inherent difficulties, we find a way to bypass loop nesting forests by suitably combining the approaches in [13, 16] in a novel framework, which is amenable to dynamic implementations. Another complication is that, although our problem is incremental, strong bridges may not only be deleted but also added (when a new SCC is formed). As a result, our data structures undergo a fully dynamic repertoire of updates, which is known to be harder. By organizing carefully those updates, we are still able to obtain the desired bounds. For lack of space, some technical details and proofs are omitted from this extended abstract and will be given in the full paper.

2 Dominator trees and 2-edge-connected blocks

Given a rooted tree, we denote by $T(v)$ the subtree of T rooted at v (we also view $T(v)$ as the set of descendants of v). Given a digraph $G = (V, E)$, and a set of vertices $S \subseteq V$, we denote by $G[S]$ the subgraph induced by S. We introduce next some of the building blocks of our new incremental algorithm.

Flow graphs, dominators, and bridges. A *flow graph* is a digraph with a distinguished *start vertex* s such that every vertex is reachable from s. Let $G = (V, E)$ be a strongly connected graph. The *reverse digraph* of G, denoted by $G^R = (V, E^R)$, is obtained by reversing the direction of all edges. Let s be a fixed but arbitrary start vertex of a strongly connected digraph G. Since G is strongly connected, all vertices are reachable from s and reach s, so we can view both G and G^R as flow graphs with start vertex s. To avoid ambiguities, throughout the paper we will denote those flow graphs respectively by G_s and G_s^R. Vertex u is a *dominator* of vertex v (u *dominates* v) in G_s if every path from s to v in G_s contains u. The dominator relation can be represented by a tree D rooted at s, the *dominator tree* of G_s: u dominates v if and only if u is an ancestor of v in D. For any $v \neq s$, we denote by $d(v)$ the parent of v in D. Similarly, we can define the dominator relation in the flow graph G_s^R, and let D^R denote the dominator tree of G_s^R, and $d^R(v)$ the parent of v in D^R. Dominators and dominator trees can be computed in linear time [2, 5, 9, 12]. An edge (u, v) is a *bridge* of a flow graph G_s if all paths from s to v include (u, v).[1] Let s be an arbitrary start vertex of G.

[1] Throughout the paper, to avoid confusion we use consistently the term *bridge* to refer to a bridge of a flow graph and the term *strong bridge* to refer to a strong bridge in the original graph.

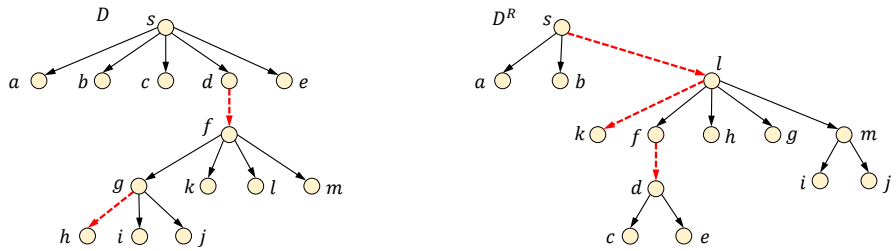


Figure 2 The dominator trees of flow graphs G_s and G_s^R. Strong bridges of G are shown red and dashed. (Better viewed in color.)

As shown in [24], an edge $e = (u, v)$ is strong bridge of G if and only if it is either a bridge of G_s or a bridge of G_s^R.

As a consequence, all the strong bridges of G can be obtained from the bridges of the flow graphs G_s and G_s^R, and thus there can be at most $2(n-1)$ strong bridges overall. Figure 2 illustrates the dominator trees D and D^R of the flow graphs G_s and G_s^R that correspond to the strongly connected digraph G of Figure 1. After deleting from the dominator trees D and D^R respectively the bridges of G_s and G_s^R, we obtain the *bridge decomposition* of D and D^R into forests $\mathcal{D}$ and $\mathcal{D}^R$. Throughout the paper, we denote by D_u (resp., D_u^R) the tree in $\mathcal{D}$ (resp., $\mathcal{D}^R$) containing vertex u, and by r_u (resp., r_u^R) the root of D_u (resp., D_u^R). The following lemma from [13] holds for a flow graph G_s of a strongly connected digraph G (and hence also for the flow graph G_s^R of G^R).

▶ **Lemma 1** ([13]). *Let G be a strongly connected digraph and let (u, v) be a strong bridge of G. Also, let D be the dominator tree of the flow graph G_s, for an arbitrary start vertex s. Suppose $u = d(v)$. Let w be any vertex that is not a descendant of v in D. Then there is path from w to v in G that does not contain any proper descendant of v in D. Moreover, all simple paths in G from w to any descendant of v in D must contain the edge $(d(v), v)$.*

Lemma 1 gives an initial partition of the vertices of G into subsets that contain the 2-edge-connected blocks of G. That is, for any two vertices u and v, we have $u \leftrightarrow_{2e} v$ only if u and v are in the same trees in the forests $\mathcal{D}$ and $\mathcal{D}^R$ (i.e., $r_u = r_v$ and $r_u^R = r_v^R$).

Loop nesting forests and bridge-dominated components. Let G be a digraph. A *loop nesting forest* represents a hierarchy of strongly connected subgraphs of G [35], defined with respect to a dfs tree T of G, as follows. For any vertex u, $loop(u)$ is the set of all descendants x of u in T such that there is a path from x to u in G containing only descendants of u in T. Any two vertices in $loop(u)$ reach each other. Therefore, $loop(u)$ induces a strongly connected subgraph of G; it is the unique maximal set of descendants of u in T (that includes u) that does so. The $loop(u)$ sets form a laminar family of subsets of V: for any two vertices u and v, $loop(u)$ and $loop(v)$ are either disjoint or nested. The *loop nesting forest* H of G, with respect to T, is the forest in which the parent of any vertex v, denoted by $h(v)$, is the nearest proper ancestor u of v in T such that $v \in loop(u)$ if there is such a vertex u, and null otherwise. Then $loop(u)$ is the set of all descendants of vertex u in H, which we will also denote as $H(u)$ (the subtree of H rooted at vertex u). A loop nesting forest can be computed in linear time [5, 35]. Since we deal with strongly connected digraphs, each vertex is contained in a loop, so H is a tree. Therefore, we will refer to H as the *loop nesting tree* of G. Let $e = (u, v)$ be a bridge of the flow graph G_s, and let $G[D(v)]$ denote the

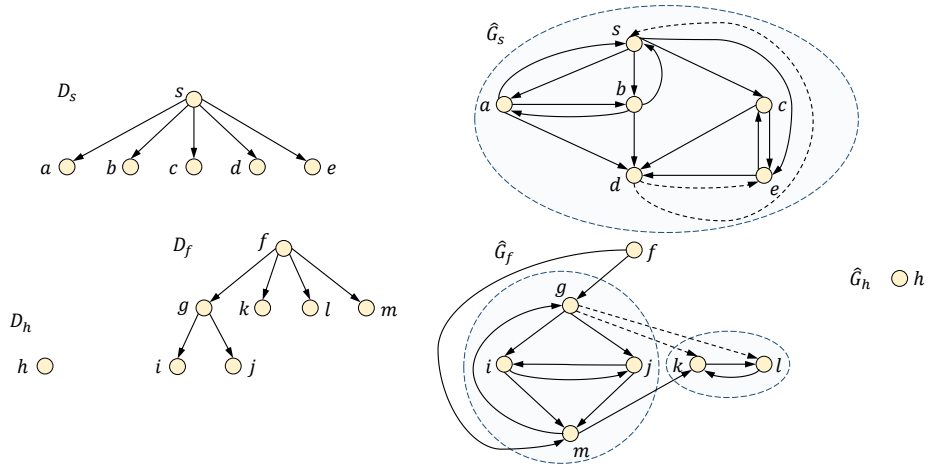


Figure 3 The bridge decomposition of the dominator tree D of Figure 2, the corresponding auxiliary graphs $\widehat{G}_r$ (the auxiliary edges are shown dashed) and their SCC's shown encircled.

subgraph induced by the vertices in $D(v)$. Let C be an SCC of $G[D(v)]$: we say that C is an *e-dominated component* of G. We also say that $C \subseteq V$ is a *bridge-dominated component* if it is an e-dominated component for some bridge e: it can be shown that bridge-dominated components form a laminar family. Let $e = (u, v)$ be a bridge of G_s, and let w be a vertex in $D(v)$ such that $h(w) \notin D(v)$. As shown in [16], $H(w)$ induces an SCC in $G[D(v)]$, and thus it is an e-dominated component.

Bridge decomposition and auxiliary graphs. Now we define a notion of *auxiliary graphs* that play a key role in our approach. Auxiliary graphs were defined in [13] to decompose the input digraph G into smaller digraphs (not necessarily subgraphs of G) that maintain the original 2-edge-connected blocks of G. Unfortunately, the auxiliary graphs of [13] are not suitable for our purposes, and we need a slightly different definition. For each root r of a tree in the bridge decomposition $\mathcal{D}$ we define the *auxiliary graph* $\widehat{G}_r = (V_r, E_r)$ *of* r as follows. The vertex set V_r of $\widehat{G}_r$ consists of all the vertices in D_r. The edge set E_r contains all the edges of G among the vertices of V_r, referred to as *ordinary* edges, and a set of *auxiliary* edges, which are obtained by contracting vertices in $V \setminus V_r$, as follows. Let v be a vertex in V_r that has a child w in $V \setminus V_r$. Note that (v, w) is a bridge and w is a root in the bridge decomposition $\mathcal{D}$ of D. For each such child w of v, we contract w and all its descendants in D into v. Figure 3 shows the bridge decomposition of the dominator tree D and the corresponding auxiliary graphs. Differently from [13], our auxiliary graphs do not preserve the 2-edge-connected blocks of G. Note that each vertex appears exactly in one auxiliary graph. Furthermore, each original edge corresponds to at most one auxiliary edge. Therefore, the total number of vertices in all auxiliary graphs is n, and the total number of edges is at most m. We use the term *auxiliary components* to refer to the SCC's of the auxiliary graphs.

▶ **Lemma 2.** *All the auxiliary graphs of a flow graph G_s can be computed in linear time.*

A new algorithm for 2-edge-connected blocks. We next sketch a new linear-time algorithm to compute the 2-edge-connected blocks that combines ideas from [13] and [16] and that will be useful for our incremental algorithm. We refer to this algorithm as the 2ECB labeling algorithm. Similarly to [16], our algorithm assigns a label to each vertex, so that two vertices

are 2-edge-connected if and only if they have the same label. The labels are defined by the bridge decomposition of the dominator trees and by the auxiliary components, as follows. Let $\widehat{G}_r$ be an auxiliary graph of G_s. We pick a *canonical vertex* for each SCC C of $\widehat{G}_r$, and denote by c_x the canonical vertex of the SCC that contains x. We define c_x^R for the SCC's of the auxiliary graphs of G_s^R analogously. We define the label of x as $label(x) = \langle r_x, c_x, r_x^R, c_x^R \rangle$.

▶ **Lemma 3.** *Let x and y be any vertices of G. Then, x and y are 2-edge-connected if and only if $label(x) = label(y)$.*

▶ **Theorem 4.** *The 2ECB labeling algorithm computes the 2-edge-connected blocks of a strongly connected digraph in linear time.*

Incremental dominators and incremental SCC's. We will use two other building blocks for our new algorithm, namely incremental algorithms for maintaining dominator trees and SCC's. As shown in [15], the dominator tree of a flow graph with n vertices can be maintained in $O(m \min\{n, k\} + kn)$ time during a sequence of k edge insertions, where m is the total number of edges after all insertions. For maintaining the SCC's of a digraph incrementally, Bender et al. [4] presented an algorithm that can handle the insertion of m edges in a digraph with n vertices in $O(m \min\{m^{1/2}, n^{2/3}\})$ time. Since we aim at an $O(mn)$ bound, we maintain the SCC's with a simpler data structure based on topological sorting [29], augmented so as to handle cycle contractions, as suggested by [17]. We refer to this data structure as IncSCC, and we will use it both for maintaing the SCC's of the input graph, and the auxiliary components (i.e., the SCC's of the auxiliary graphs). We maintain the SCC's and a topological order for them. Each SCC is represented by a canonical vertex, and the partition of the vertices into SCC's is maintained through a set union data structure [34, 36]. The data structure supports $unite(p, q)$, which, given canonical vertices p and q, merges the SCC's containing p and q into one new SCC and makes p the canonical vertex of the new SCC. It also supports $find(v)$, which returns the canonical vertex of the SCC containing v. Here we use the abbreviation $f(v)$ to stand for $find(v)$. The topological order is represented by a simple numbering scheme, where each canonical vertex is numbered with an integer in the range $[1, n]$, so that if (u, v) is an edge of G, then either $f(u) = f(v)$ (u and v are in the same SCC) or $f(u)$ is numbered less than $f(v)$ (when u and v are in different SCC's). With each canonical vertex p we store a list $out(p)$ of edges leaving vertices that are in the same SCC as p, i.e., edges (u, v) with $f(u) = p$. Note that $out(p)$ may contain multiple vertices in the same SCC (i.e., vertices u and v with $f(u) = f(v)$), due to the SCC contractions (and shortcut edges, in case of the auxiliary components) during edge insertions. Also, $out(p)$ may contain loops, that is, vertices v with $f(v) = p$. Each out list is stored as a doubly linked circular list, so that we can merge two lists and delete a vertex from a list in $O(1)$. When the incremental SCC data structure detects that a new SCC is formed, it locates the SCC's that are merged and chooses a canonical vertex for the new SCC. The IncSCC data structure can handle m edge insertions in a total of $O(mn)$ time.

3 Incremental 2-edge-connectivity in strongly connected digraphs

To maintain the 2-edge-connected blocks of a strongly connected digraph during edge insertions, we design an incremental version of the labeling algorithm of Section 2. In order to respond to the insertion of an edge, we have to update the vertex labels, so we need to update both the bridge decomposition of D and D^R, and the strongly connected components of the resulting auxiliary graphs. Note, in particular, that the second task involves moving

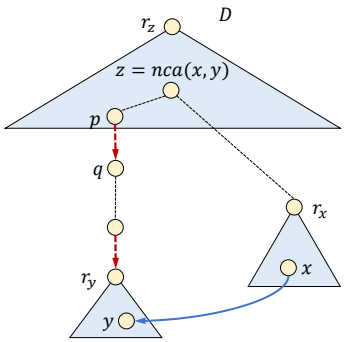


Figure 4 The bridge decomposition of D before the insertion of a new edge (x, y).

and merging vertices from one auxiliary graph to another. If labels are maintained explicitly, one can answer in $O(1)$ time queries on whether two vertices are 2-edge-connected, and report in $O(n)$ time all the 2-edge-connected blocks. Let (x, y) be the edge to be inserted. We say that vertex v is *affected* by the update if $d(v)$ (its parent in D) changes. Let $Dom(v)$ denote the set of all vertices that dominate v: note that $Dom(v)$ may change even if v is not affected. Similarly, an auxiliary component (resp., auxiliary graph) is affected if it contains an affected vertex.

We let $nca(x, y)$ denote the nearest common ancestor of x and y in the dominator tree D. We also denote by $D[u, v]$ the path from u to v in D. If $nca(x, y)$ and y are in different subtrees in the bridge decomposition of D before the insertion of the edge (x, y), we let (p, q) be the first bridge encountered on the path $D[nca(x, y), y]$ (Figure 4). We denote by $depth(v)$ the depth of vertex v in D. Most of the proofs in this section will be given in the full paper.

Affected vertices and canceled bridges. There are affected vertices after the insertion of (x, y) if and only if $nca(x, y)$ is not a descendant of $d(y)$ [32]. A characterization of the affected vertices is provided by the following lemma, which is a refinement of a result in [3].

▶ **Lemma 5.** ([15]) *A vertex v is affected after the insertion of edge (x, y) if and only if $depth(nca(x, y)) < depth(d(v))$ and there is a path π in G from y to v such that $depth(d(v)) < depth(w)$ for all $w \in \pi$. If v is affected, then it becomes a child of $nca(x, y)$ in D.*

The algorithm in [15] applies Lemma 5 to identify affected vertices by starting a search from y (if y is not affected, then no other vertex is). We assume that the outgoing and incoming edges of each vertex are maintained as linked lists, so that a new edge can be inserted in $O(1)$, and that the dominator tree D is represented by the parent function d. We also maintain the depth of vertices in D. We say that a vertex v is *scanned*, if the edges leaving v are examined during the search for affected vertices, and that it is *visited* if there is a scanned vertex u such that (u, v) is an edge in G. Every scanned vertex is either affected or a descendant of an affected vertex in D. By Lemma 5, a visited vertex v is scanned if $depth(nca(x, y)) < depth(d(v))$. Let (u, v) be a bridge of G_s. We say that (u, v) is *canceled* by the insertion of edge (x, y) if it is no longer a bridge after the insertion. We say that (u, v) is *locally canceled* if (u, v) is a canceled bridge and v is not affected. We need to treat the case of locally canceled bridges separately because in such an event the bridge decomposition of D changes, even if D remains the same. Note that if (u, v) is locally canceled, then $u = nca(x, y)$. In the next lemmata, we consider the effect of the insertion of edge (x, y) on

the bridges of G_s, and relate the affected and scanned vertices with the auxiliary components. Recall that (p, q) is the first bridge encountered on the path $D[nca(x, y), y]$ (Figure 4), $D(v)$ denotes the descendants of v in D, and $G[C]$ is the subgraph induced by the vertices in C.

▶ **Lemma 6.** *Suppose that bridge (p, q) is not locally canceled after the insertion of (x, y). Let $z = nca(x, y)$ and let v be an affected vertex such that $r_v \neq r_z$. All vertices reachable from v in $G[D(q)]$ are either affected or scanned.*

▶ **Lemma 7.** *Let $e = (u, v)$ be a bridge of G_s that is canceled by the insertion of edge (x, y). Then (i) y is a descendant of v in D, and (ii) y is in the same e-dominated component as v.*

▶ **Corollary 8.** *A bridge $e = (u, v)$ of G_s is canceled by the insertion of edge (x, y) if and only if $depth(nca(x, y)) \leq depth(u)$ and there is a path π in G from y to v such that $depth(u) < depth(w)$ for all $w \in \pi$.*

By Corollary 8, we can use the incremental algorithm of [15] to detect canceled bridges, without affecting the $O(mn)$ bound. Indeed, suppose $e = (u, v)$ is a canceled bridge. By Lemma 7, y is a descendant of v in D and in the same e-dominated component as v. Hence, v will be visited by the search from y. If a bridge (u, v) is locally canceled, there can be vertices in D_v that are not scanned, and that after the insertion will be located in D_u, without having their depth changed. This is a difficult case for our analysis: fortunately, the following lemma shows that this case can happen only $O(n)$ times overall.

▶ **Lemma 9.** *Suppose (u, v) is a bridge of G_s that is locally canceled by the insertion of edge (x, y). Then (u, v) is no longer a strong bridge in G after the insertion.*

Note that a canceled bridge that is not locally canceled may still appear as a bridge in G_s^R after the insertion of edge (x, y). The next lemmata allow us to identify the necessary changes in the auxiliary components of the affected subgraphs and $\widehat{G}_{r_z}$. All lemmata assume that bridge (p, q) is not locally canceled after the insertion of (x, y) and that $z = nca(x, y)$.

▶ **Lemma 10.** *Let C be an affected auxiliary component of an auxiliary graph $\widehat{G}_r$ with $r \neq r_z$. Then C consists of a set of affected siblings in D and possibly some of their affected or scanned descendants in D.*

An auxiliary component is *scanned* if it contains a scanned vertex. As with vertices, affected auxiliary components are also scanned (the converse is not necessarily true).

▶ **Lemma 11.** *Let C be a scanned auxiliary component of an auxiliary graph $\widehat{G}_r$ with $r \neq r_z$. Then all vertices in C are scanned.*

We say that a vertex v is *moved* if it is located in an auxiliary graph $\widehat{G}_r$ with $r \neq r_z$ before the insertion of (x, y), and in $\widehat{G}_{r_z}$ after the insertion. Lemmata 10 and 11 imply that if an auxiliary component C contains a moved vertex, then all vertices in the component are also moved. We call such an auxiliary component *moved*. Now we describe how to find the moved auxiliary components that need to be merged. Let H be the subgraph of G induced by the scanned vertices in $D(q)$. We refer to H as the *scanned subgraph*.

▶ **Lemma 12.** *Let ζ and ξ be two distinct roots in the bridge decomposition of D, such that $\zeta, \xi \neq r_z$, and D_ζ and D_ξ are contained in $D(q)$. Let C_ζ and C_ξ be scanned components in $\widehat{G}_\zeta$ and $\widehat{G}_\xi$, respectively. Then C_ζ and C_ξ are strongly connected in $G[D(q)]$ if and only if they are strongly connected in H.*

Now we introduce a dummy root r^* in H, together with an edge (v, r^*) for each scanned vertex v that has a leaving edge (v, w) such that $w \in D_z$ and w is in the auxiliary component of p in $\widehat{G}_{r_z}$. We denote this graph by H^*.

▶ **Lemma 13.** *A scanned vertex $v \notin D_z$ is strongly connected in $G[D(r_z)]$ to a vertex $w \in D_z$ if and only if r^* is reachable from v in H^*. In this case, v and p are also strongly connected in $G[D(r_z)]$.*

The Algorithm. We describe next our incremental algorithm for maintaining the 2-edge-connected blocks of a strongly connected digraph G. We refer to this algorithm as SCInc2ECB(G). We initialize the algorithm and the associated data structures by executing the labeling algorithm of Section 2. Algorithm Initialize(G, s), shown below, computes the dominator tree D, the set of bridges Br of flow graph G_s, the bridge decomposition $\mathcal{D}$ of D, and the corresponding auxiliary graphs $\widehat{G}_r$. Finally, for each auxiliary graph $\widehat{G}_r$, it finds its auxiliary components, computes the labels r_w and c_w for each vertex $w \in V_r$, and initializes an IncSCC data structure. The execution of Initialize(G^R, s) performs analogous steps in the reverse flow graph G_s^R.

Algorithm 1: Initialize(G, s)

```
1  Set s to be the designated start vertex of G.
2  Compute the dominator tree D and the set of bridges Br of the corresponding flow
     graph G_s.
3  Compute the bridge decomposition 𝒟 of D.
4  foreach root r in 𝒟 do
5  |  Compute the auxiliary graph Ĝ_r of r.
6  |  Compute the strongly connected components in Ĝ_r.
7  |  foreach strongly connected component C in Ĝ_r do
8  |  |  Choose a vertex v ∈ C as the canonical vertex of the auxiliary component C.
9  |  |  foreach vertex w ∈ C do
10 |  |  |  Set r_w = r and c_w = v.
11 |  |  end
12 |  end
13 |  Initialize a IncSCC data structure for Ĝ_r.
14 end
```

Algorithm 2: SCInsertEdge(G, e)

```
1  Let s be the designated start vertex of G, and let e = (x, y).
2  Compute the nearest common ancestor z and z^R of x and y in D and D^R respectively.
3  Update the dominator trees D and D^R, and return the lists S and S^R of the vertices
     that were scanned in D and D^R respectively.
4  if a bridge is locally canceled in G_s or in G_s^R then
5  |  Execute Initialize(G, s) and Initialize(G^R, s).
6  else
7  |  Execute UpdateAC(𝒟, z, x, y, S) and UpdateAC(𝒟^R, z^R, y, x, S^R).
8  end
```

Algorithm 3: UpdateAC$(\mathcal{D}, z, x, y, L)$

Let r_z be root of the tree D_z in $\mathcal{D}$ that contains z.
Let $c_{x'}$ be the canonical vertex of the nearest ancestor x' of x in D such that $x' \in D_z$.
Let (p, q) be the first bridge on the path $D[z, y]$, and let c_p be the canonical vertex of p.
Form the scanned graph H^* that contains the scanned vertices $S \setminus D_z$ and the edges among them.
Compute the strongly connected components $\mathcal{C}$ of $H^* \setminus r^*$ and order them topologically.
Compute the components $\mathcal{C}^*$ of $\mathcal{C}$ that reach r^* in H^*.
foreach *strongly connected component C in $\mathcal{C}^*$ that is moved* **do**
    Merge C with the component of c_p.
end
forall *strongly connected components in $\mathcal{C} \setminus \mathcal{C}^*$ that are moved* **do**
    Insert the components in the topological order of $\widehat{G}_{r_z}$ just after the component of c_p.
end
foreach *vertex $w \in S$* **do**
    if *w is moved to $\widehat{G}_{r_z}$* **then** set $r_w = r_z$.
end
Update the lists of out edges in the IncSCC data structures of $\widehat{G}_{r_z}$ and of the affected auxiliary graphs.
Insert edge $(c_{x'}, y)$ in the list of outgoing edges of $c_{x'}$ and update the IncSCC data structure of $\widehat{G}_{r_z}$.

When a new edge $e = (x, y)$ is inserted, algorithm SCInc2ECB executes procedure SCInsertEdge(G, e), which updates dominator trees D and D^R, together with the corresponding bridge decompositions. It also finds the set of scanned vertices in G_s and G_s^R. If a bridge of D or D^R is locally cancelled, then we restart the algorithm by executing Initialize. Otherwise, we need to update the auxiliary components in G_s and G_s^R. These updates are handled by procedure UpdateAC. Before describing UpdateAC, we provide some details on the implementation of the IncSCC data structures, which maintain the auxiliary components of each auxiliary graph $\widehat{G}_r$ using the "one-way search" structure of [17, Sections 2 and 6]. Since we need to insert and delete canonical vertices, we augment this data structure as follows. We maintain the canonical vertices of each auxiliary component in a linked list L_r, arranged according to the given topological order of $\widehat{G}_r$. For each vertex v in L_r, we also maintain a rank in L_r which is an integer in $[1, n]$ such that for any two canonical vertices u and v in L_r, $rank(u) < rank(v)$ if and only if u precedes v in L_r. The ranks of all vertices can be stored in a single array of size n. Also, with each canonical vertex w, we store a pointer to the location of w in L. We represent L_r with a doubly linked list so that we can insert and delete a canonical vertex in constant time. When we remove vertices from a list L_r we do not need to update the ranks of the remaining vertices in L_r. The insertion of an edge (x, y) may remove vertices from various lists L_r, but may insert vertices only in L_{r_z}. After these insertions, we recompute the ranks of all vertices in L_{r_z} just by traversing the list and assigning rank i to the i-th vertex in the list. We maintain links between an original edge e, stored in the adjacency lists of G, and at most one copy of e in a *out* list of IncSCC. This enables us to keep for each shortcut edge $e' = (v', w)$ a one-to-one correspondence with the original edge $e = (v, w)$ that created e'. We do that because if an ancestor of v is moved

to the auxiliary graph $\widehat{G}_{r_z}$ that contains v' ($v' = p$ in Figure 5), then e may correspond to a different shortcut edge or it may even become an ordinary edge of $\widehat{G}_{r_z}$. Using this mapping we can update the *out* lists of IncSCC. To initialize the IncSCC structure of an auxiliary graph, we compute a topological order of the auxiliary components in $\widehat{G}_r$, and create the list of outgoing edges $out(v)$ for each canonical vertex v.

If inserting edge (x, y) does not locally cancel a bridge in G_s and G_s^R, then we update the auxiliary components of G_s using procedure UpdateAC($\mathcal{D}, z, x, y, S$), where $\mathcal{D}$ is the updated bridge decomposition of D, $z = nca(x, y)$, and S is a list of the vertices scanned during the update of D. We do the same to update the auxiliary components of G_s^R. Procedure UpdateAC first computes the auxiliary components that are moved to $\widehat{G}_{r_z}$, possibly merging some of them, and then inserts the edge (x, y) as an original or a shortcut edge of $\widehat{G}_{r_z}$, depending on whether $x \in D_{r_z}$ or not. Note that the insertion of (x, y) may cause the creation of a new auxiliary component in $\widehat{G}_{r_z}$. Now we specify some further details in the implementation of UpdateAC. The vertices that are moved to $\widehat{G}_{r_z}$ are the scanned vertices in S that are not descendants of a strong bridge. Hence, we can mark the vertices that are moved to $\widehat{G}_{r_z}$ during the search for affected vertices. The next task is to update the *out* lists of the canonical vertices in $\widehat{G}_{r_z}$ and the affected auxiliary graphs. We process the list of scanned vertices S as follows. Let v be such a vertex. If v is not marked, i.e., is not moved to $\widehat{G}_{r_z}$, then we process the edges leaving v; otherwise, we process both the edges leaving v and the edges entering v. Suppose v is marked. Let (v, w) be an edge leaving v in G. If w is also in $\widehat{G}_{r_z}$ after the insertion, then we add the edge (v, w) in $out(f(v))$. Moreover, if w is not in S, then it was already located in $\widehat{G}_{r_z}$ before the insertion, so we delete the shortcut edge stored in $out(f(p))$. If w is not in $\widehat{G}_{r_z}$ after the insertion, then (v, w) is a bridge in D and we do nothing. Now consider an edge (w, v) entering v in G. If w is scanned, then we will process (w, v) while processing the edges leaving w. Otherwise, w remains a descendant of p, so we insert the edge (w, v) in $out(f(p))$. Now we consider the unmarked scanned vertices v. Let (v, w) an edge leaving v in G. If $w \in D_z$, we insert the edge (v, w) into $out(f(v'))$, where v' is the nearest marked ancestor of v in D. Otherwise, if $w \notin D(r_z)$, the edge (v'', w), where v'' is the nearest ancestor of v in D_w, already exists since v was a descendant of v'' before the insertion of (x, y). Next, we consider the updates in the L_r lists and the vertex ranks. While we process S, if we encounter a moved canonical vertex $v \in S$ that was located in an auxiliary graph $\widehat{G}_r$ with $r_z \neq r$, then we delete v from L_r. Note that we do not need to update the ranks of the remaining vertices in lists L_r with $r \neq r_z$. To update L_{r_z}, we insert the moved canonical vertices of the SCC's in $\mathcal{C} \setminus \mathcal{C}^*$, in a topological order of $H = H^* \setminus r^*$, just after $f(p)$. Then we traverse L_{r_z} and update the ranks of the canonical vertices. The final step is to actually insert edge (x, y) in the IncSCC data structure of $\widehat{G}_{r_z}$. We do that by adding (x, y) in $out(f(x'))$, where x' is the nearest ancestor of x in D_z. If $rank(f(x')) > rank(f(y))$, then we execute the forward-search procedure of IncSCC.

The proof of correctness of Algorithm SCInc2ECB will be given in the full paper.

Running time of SCInc2ECB. We analyze the running time of Algorithm SCInc2ECB. Recall that G is a strongly connected digraph with n vertices that undergoes a sequence of edge insertions. We let m be the total number of edges in G after all insertions ($m \geq n$). First, we bound the time spent by Initialize. This procedure is called twice in the beginning of the SCInc2ECB, and twice after each time a bridge in G_s or in G_s^R is locally canceled. Then, Lemma 9 implies that such an event can happen at most $2(n-1)$ times. Hence, there are at most $4n$ calls to Initialize, and since each execution takes $O(m)$ time, the total time spent on Initialize is $O(mn)$. Similarly, the dominator trees of G_s and G_s^R can be updated

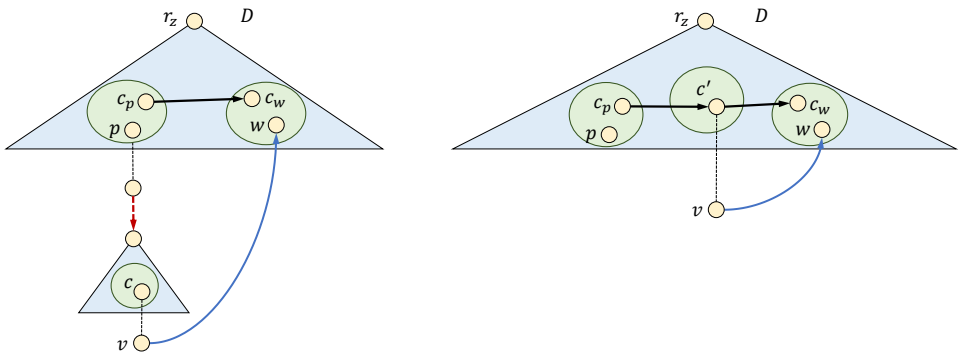


Figure 5 Before the insertion of (x, y), edge (v, w) corresponds to the shortcut edge (p, w) of $\widehat{G}_{r_z}$, and is stored in $out(c_p)$. An auxiliary component with canonical vertex c is affected by the insertion of (x, y) and is merged into a component with canonical vertex c' ($c' = c$ if the component is moved without merging with another component). Now c' becomes the canonical vertex of the nearest ancestor of v in D_z, and edge (v, w) is stored as a shortcut edge in $out(c')$.

in total $O(mn)$ time [15]. We next bound the total time required to update the auxiliary components. Consider an execution of UpdateAC. Let ν and μ, respectively, be the number of scanned vertices, after the insertion of edge (x, y), and their adjacent edges. The time to compute the affected subgraph H^*, compute the SCC's of $H^* \setminus r^*$, and the vertices that reach r^* is $O(\nu + \mu)$. In the same time, we can update the auxiliary components of $\widehat{G}_{r_z}$ and of the affected auxiliary graphs, their corresponding topological orders, and the *out* lists of the corresponding IncSCC data structures. Since each scanned vertex w is a descendant of an affected vertex, the depth of w decreases by at least one. Hence, the total time spent by UpdateAC for all insertions, excluding the execution of line 17, is $O(mn)$. It remains to bound the time required by the IncSCC data structures to handle the edge insertions in line 17 of UpdateAC. To do this, we extend the analysis from [17]. Note that we cannot immediately apply the analysis in [17], since here we have the complication that vertices and edges can be inserted to and removed from the IncSCC structures. We say that a vertex v and an edge e are *related* if there is a path that contains both v and e (in any order). Then, there are $O(mn)$ pairs of vertices and edges that can be related in all IncSCC structures for every auxiliary graph. We argue that each time the IncSCC structure traverses an edge (after the insertion in line 17 of UpdateAC), the cost of this action can be charged to a newly-related vertex-edge pair. Consider a vertex v and an edge $e = (u, w)$. Call the pair $\langle v, e\rangle$ *active* if v and e are in the same auxiliary graph $\widehat{G}_r$, and *inactive* otherwise. Note that since we identify shortcut edges with their corresponding original edge, e may actually appear in $\widehat{G}_r$ as an edge (u', w), where u' is the nearest ancestor of u in D_r. This fact, however, does not affect our analysis.

▶ **Lemma 14.** *The total number of edge traversals made during the forward searches in all IncSCC data structures is $O(mn)$.*

Proof. To prove the bound, it suffices to show that in all IncSCC data structures the total number of unrelated $\langle v, e\rangle$ pairs that are ever created is $O(mn)$. Consider an active pair $\langle v, e\rangle$ that becomes related in $\widehat{G}_r$. Then there is some path π in $G[D(r)]$ that contains both v and e. Suppose that the pair $\langle v, e\rangle$ later becomes active but unrelated in an auxiliary graph $\widehat{G}_{r'}$, where r' may be vertex r. Then π does not exist in $G[D(r')]$, which implies that some vertices of π are not descendants of r'. Then, by Lemma 1, π must contain the

bridge $(d(r'), r')$. Since π exists in $G[D(r)]$, the bridge $(d(r'), r')$ was a descendant of r before some insertion, and then became an ancestor of v. But this is impossible, since after an edge insertion, the new parent $d'(v)$ of v is on the path $D[s, d(v)]$. Hence, once a $\langle v, e\rangle$ pair becomes related, it can never become unrelated. The bound follows. ◀

▶ **Lemma 15.** *The total time to update all the IncSCC data structures is $O(mn)$.*

Proof. Updating the lists of out edges in the IncSCC data structures, and inserting or deleting canonical vertices can be charged to the cost of updating the dominator tree, and is thus $O(mn)$. By Lemma 14, all edge insertions that do not trigger merges of auxiliary components can be handled in $O(mn)$ time. The number of edge insertions that trigger merges of auxiliary components is at most $n - 1$, and each such insertion can be handled in $O(m + n)$ time, excluding unite operations. Taking into account also the total time for all unite operations yields the lemma. ◀

▶ **Theorem 16.** *The total running time of Algorithm SCInc2ECB for a sequence of edge insertions in a strongly connected digraph with n vertices is $O(mn)$, where m is the total number of edges in G after all insertions.*

Extension to general digraphs. Our approach can be extended to general (not strongly connected) digraphs, as shown in the following theorem. Details will be given in the full paper.

▶ **Theorem 17.** *We can maintain the 2-edge-connected blocks of a digraph with n vertices through a sequence of edge insertions in $O(mn)$ time, where m is the total number of edges in G after all insertions.*

References

1. A. Abboud and V. Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th IEEE Symp. on Foundations of Computer Science, FOCS*, pages 434–443, 2014. doi:10.1109/FOCS.2014.53.
2. S. Alstrup, D. Harel, P. W. Lauridsen, and M. Thorup. Dominators in linear time. *SIAM Journal on Computing*, 28(6):2117–32, 1999.
3. S. Alstrup and P. W. Lauridsen. A simple dynamic algorithm for maintaining a dominator tree. Technical Report 96-3, Department of Computer Science, University of Copenhagen, 1996.
4. M. A. Bender, J. T. Fineman, S. Gilbert, and R. E. Tarjan. A new approach to incremental cycle detection and related problems. *ACM Transactions on Algorithms*, 12(2):14:1–14:22, December 2015. doi:10.1145/2756553.
5. A. L. Buchsbaum, L. Georgiadis, H. Kaplan, A. Rogers, R. E. Tarjan, and J. R. Westbrook. Linear-time algorithms for dominators and other path-evaluation problems. *SIAM Journal on Computing*, 38(4):1533–1573, 2008.
6. C. Demetrescu and G. F. Italiano. A new approach to dynamic all pairs shortest paths. *Journal of ACM*, 51(6):968–992, 2004. doi:10.1145/1039488.1039492.
7. C. Demetrescu and G. F. Italiano. Mantaining dynamic matrices for fully dynamic transitive closure. *Algorithmica*, 51(4):387–427, 2008. doi:10.1007/s00453-007-9051-4.
8. D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig. Sparsification – A technique for speeding up dynamic graph algorithms. *Journal of ACM*, 44(5):669–696, September 1997. doi:10.1145/265910.265914.

9 W. Fraczak, L. Georgiadis, A. Miller, and R. E. Tarjan. Finding dominators via disjoint set union. *Journal of Discrete Algorithms*, 23:2–20, 2013. doi:10.1016/j.jda.2013.10.003.

10 P. G. Franciosa, G. Gambosi, and U. Nanni. The incremental maintenance of a depth-first-search tree in directed acyclic graphs. *Information Processing Letters*, 61(2):113–120, 1997. doi:10.1016/S0020-0190(96)00202-5.

11 G. N. Frederickson. Data structures for on-line updating of minimum spanning trees. *SIAM Journal on Computing*, 14:781–798, 1985.

12 H. N. Gabow. The minset-poset approach to representations of graph connectivity. *ACM Transactions on Algorithms*, 12(2):24:1–24:73, February 2016. doi:10.1145/2764909.

13 L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. 2-edge connectivity in directed graphs. In *Proc. 26th ACM-SIAM Symp. on Discrete Algorithms*, pages 1988–2005, 2015.

14 L. Georgiadis, G. F. Italiano, L. Laura, and N. Parotsidis. 2-vertex connectivity in directed graphs. In *Proc. 42nd Int'l. Coll. on Automata, Languages, and Programming*, pages 605–616, 2015.

15 L. Georgiadis, G. F. Italiano, L. Laura, and F. Santaroni. An experimental study of dynamic dominators. In *Proc. 20th European Symp. on Algorithms*, pages 491–502, 2012.

16 L. Georgiadis, G. F. Italiano, and N. Parotsidis. A New Framework for Strong Connectivity and 2-Connectivity in Directed Graphs. *ArXiv e-prints*, abs/1511.02913, November 2015. arXiv:1511.02913.

17 B. Haeupler, T. Kavitha, R. Mathew, S. Sen, and R. E. Tarjan. Incremental cycle detection, topological ordering, and strong component maintenance. *ACM Transactions on Algorithms*, 8(1):3:1–3:33, January 2012. doi:10.1145/2071379.2071382.

18 M. Henzinger, S. Krinninger, and V. Loitzenbauer. Finding 2-edge and 2-vertex strongly connected components in quadratic time. In *Proc. 42nd International Colloquium on Automata, Languages, and Programming (ICALP 2015)*, 2015.

19 M. R. Henzinger and V. King. Fully dynamic biconnectivity and transitive closure. In *Proc. 36th IEEE Symp. on Foundations of Computer Science*, pages 664–672, 1995. doi:10.1109/SFCS.1995.492668.

20 M. R. Henzinger and V. King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of ACM*, 46(4):502–536, 1999.

21 M. R. Henzinger and V. King. Maintaining minimum spanning forests in dynamic graphs. *SIAM Journal on Computing*, 31(2):364–374, February 2002. doi:10.1137/S0097539797327209.

22 J. Holm, K. de Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of ACM*, 48(4):723–760, July 2001. doi:10.1145/502090.502095.

23 G. F. Italiano. Amortized efficiency of a path retrieval data structure. *Theoretical Computer Science*, 48(3):273–281, 1986. doi:10.1016/0304-3975(86)90098-8.

24 G. F. Italiano, L. Laura, and F. Santaroni. Finding strong bridges and strong articulation points in linear time. *Theoretical Computer Science*, 447:74–84, 2012. doi:10.1016/j.tcs.2011.11.011.

25 R. Jaberi. Computing the 2-blocks of directed graphs. *RAIRO-Theor. Inf. Appl.*, 49(2):93–119, 2015. doi:10.1051/ita/2015001.

26 R. Jaberi. On computing the 2-vertex-connected components of directed graphs. *Discrete Applied Mathematics*, 204:164–172, 2016. doi:10.1016/j.dam.2015.10.001.

27 V. King. Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. In *Proc. 40th IEEE Symp. on Foundations of Computer Science, FOCS'99*, pages 81–91, 1999. doi:10.1109/SFFCS.1999.814580.

28 S. Makino. An algorithm for finding all the k-components of a digraph. *International Journal of Computer Mathematics*, 24(3–4):213–221, 1988.

29 A. Marchetti-Spaccamela, U. Nanni, and H. Rohnert. Maintaining a topological order under edge insertions. *Information Processing Letters*, 59(1):53–58, 1996. doi:10.1016/0020-0190(96)00075-0.

30 H. Nagamochi and T. Watanabe. Computing k-edge-connected components of a multigraph. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E76–A.4:513–517, 1993.

31 M. Pătraşcu and M. Thorup. Planning for fast connectivity updates. In *Proc. 48th IEEE Symp. on Foundations of Computer Science*, FOCS'07, pages 263–271, 2007. doi:10.1109/FOCS.2007.54.

32 G. Ramalingam and T. Reps. An incremental algorithm for maintaining the dominator tree of a reducible flowgraph. In *Proc. of the 21st ACM SIGPLAN-SIGACT Symp. on Principles of programming languages*, pages 287–296, 1994.

33 R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

34 R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of ACM*, 22(2):215–225, 1975.

35 R. E. Tarjan. Edge-disjoint spanning trees and depth-first search. *Acta Informatica*, 6(2):171–85, 1976.

36 R. E. Tarjan and J. van Leeuwen. Worst-case analysis of set union algorithms. *Journal of ACM*, 31(2):245–81, 1984.

37 M. Thorup. Near-optimal fully-dynamic graph connectivity. In *Proc. 32nd ACM Symp. on Theory of Computing*, STOC'00, pages 343–350, 2000. doi:10.1145/335305.335345.

38 M. Thorup. Fully-dynamic all-pairs shortest paths: Faster and allowing negative cycles. In *Algorithm Theory – SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory,*, pages 384–396, 2004. doi:10.1007/978-3-540-27810-8_33.

Unified Acceleration Method for Packing and Covering Problems via Diameter Reduction*

Di Wang[†1], **Satish Rao**[‡2], **and Michael W. Mahoney**[§3]

1 **Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, USA**
wangd@eecs.berkeley.edu

2 **Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, USA**
satishr@berkeley.edu

3 **International Computer Science Institute and Department of Statistics, UC Berkeley, Berkeley, USA**
mmahoney@stat.berkeley.edu

Abstract

In a series of recent breakthroughs, Allen-Zhu and Orecchia [2, 1] leveraged insights from the linear coupling method [15], which is a first-order optimization scheme, to provide improved algorithms for packing and covering linear programs. The result in [1] is particularly interesting, as the algorithm for packing LP achieves both width-independence and Nesterov-like acceleration, which was not known to be possible before. Somewhat surprisingly, however, while the dependence of the convergence rate on the error parameter ϵ for packing problems was improved to $O(1/\epsilon)$, which corresponds to what accelerated gradient methods are designed to achieve, the dependence for covering problems was only improved to $O(1/\epsilon^{1.5})$, and even that required a different more complicated algorithm, rather than from Nesterov-like acceleration. Given the primal-dual connection between packing and covering problems and since previous algorithms for these very related problems have led to the same ϵ dependence, this discrepancy is surprising, and it leaves open the question of the exact role that the linear coupling is playing in coordinating the complementary gradient and mirror descent step of the algorithm. In this paper, we clarify these issues, illustrating that the linear coupling method can lead to improved $O(1/\epsilon)$ dependence for both packing and covering problems in a unified manner, i.e., with the same algorithm and almost identical analysis. Our main technical result is a novel dimension lifting method that reduces the coordinate-wise diameters of the feasible region for covering LPs, which is the key structural property to enable the same Nesterov-like acceleration as in the case of packing LPs. The technique is of independent interest and that may be useful in applying the accelerated linear coupling method to other combinatorial problems.

1998 ACM Subject Classification G.1.6 Optimization

Keywords and phrases Convex optimization, Accelerated gradient descent, Linear program, Approximation algorithm, Packing and covering

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.50

* Full version available at http://arxiv.org/abs/1508.02439.

† DW was supported by ARO Grant W911NF-12-1-0541.

‡ SR was funded by NSF Grant CCF-1118083.

§ MM acknowledges the support of the NSF, AFOSR, and DARPA.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 50; pp. 50:1–50:13

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

1 Introduction

A fractional covering problem, in its generic form, can be written as the following linear program (LP): $\min_{x\geq 0}\{c^T x : Ax \geq b\}$, where $c \in \mathbb{R}^n_{\geq 0}, b \in \mathbb{R}^m_{\geq 0}$, and $A \in \mathbb{R}^{m\times n}_{\geq 0}$.

Without loss of generality, one can scale the coefficients, in which case one can write this LP in the standard form:

$$\min_{x\geq 0}\{\vec{1}^T x : Ax \geq \vec{1}\}, \text{ where } A \in \mathbb{R}^{m\times n}_{\geq 0} \tag{1}$$

The fractional packing problem, which is the dual of fractional covering, can be written in the standard form as:

$$\max_{y\geq 0}\{\vec{1}^T y : Ay \leq \vec{1}\}, \text{ where } A \in \mathbb{R}^{m'\times n'}_{\geq 0} \tag{2}$$

We denote by OPT the optimal value of a LP. In this case, we say that x is a $(1+\epsilon)$-*approximation* for the covering LP if $Ax \geq \vec{1}$ and $\vec{1}^T x \leq (1+\epsilon)\,\text{OPT}$, and we say that y is a $(1-\epsilon)$-*approximation* for the packing LP if $Ay \leq \vec{1}$ and $\vec{1}^T y \geq (1-\epsilon)\,\text{OPT}$.

Packing and covering problems are important classes of LPs with wide applications, including most resource allocation problems, and they have long drawn interest in theoretical computer science. Although one can use general LP solvers such as the interior point method to solve packing and covering with convergence rate of $\log(1/\epsilon)$, such algorithms usually have very high per-iteration cost, as methods such as the computation of the Hessian and matrix inversion are involved. In the setting of large-scale problems, low precision iterative solvers are often more popular choices. Such solvers usually run in time with a nearly-linear dependence on the problem size, and they have $\text{poly}(1/\epsilon)$ dependence on the approximation parameter. Most such work falls into one of two categories. The first category follows the approach of transforming LPs to convex optimization problems, then applying efficient first-order optimization algorithms. Examples of work in this category include [8, 3, 9, 12, 2, 1], and all except [2, 1] apply to more general classes of LPs. The second category is based on the Lagrangian relaxation framework, and some examples of work in this category include [11, 5, 7, 13, 14, 6, 4]. For a more detailed comparison of this prior work, see Table 1 in [1]. Also, based on whether the running time depends on the width ρ, a parameter which typically depends on the dimension and the largest entry of A, these algorithms can also be divided into width-dependent solvers and width-independent solvers. Width-dependent solvers are usually pseudo-polynomial, as the running time depends at least linearly on $\rho\,\text{OPT}$, which itself can be large, while width-independent solvers are independent or logarithmically dependent on the width.

In this paper, we describe a solver for covering LPs of the form (1). The solver is width-independent, and it is a first-order method with a linear rate of convergence. That is, if we let N be the number of non-zeros in A, then the running time of our algorithm is $O\left(N\frac{\log^2(N/\epsilon)\log(1/\epsilon)}{\epsilon}\right)$. To simplify the following discussion, we will follow the standard practice of using $\tilde{O}$ to hide poly-log factors, in which case the running time of our algorithm for the covering problem is $\tilde{O}\left(N/\epsilon\right)$. Among other things, our result is an improvement over the recent bound of $\tilde{O}(N/\epsilon^{1.5})$ provided by Allen-Zhu and Orecchia for the covering problem in [1], and our result corresponds to the linear rate of convergence that accelerated gradient methods are designed to achieve [9].

At least as interesting as the $\tilde{O}(1/\epsilon^{0.5})$ improvement for covering LPs, however, is the context of this problem and the main technical contribution that we developed and exploited to achieve our improvement.

- The context for our results has to do with the linear coupling method that was introduced recently by Allen-Zhu and Orecchia [15]. This is a first order method for solving convex optimization problems, and it provides a conceptually simple way to integrate a gradient descent step and mirror descent step in each iteration. In the setting of standard smooth convex optimization, the method achieves the same convergence rate as that of the accelerated gradient descent method of Nesterov [9], and indeed the former can be viewed as an insightful reinterpretation of the latter. The high-level view of the method as a coupling of gradient descent steps and mirror descent steps offers more flexibility to the framework, as the combination allows the two steps to complement each other in ways beyond simply Nesterov-like acceleration. Indeed, it has shown initial promise by providing improved algorithms for packing and covering LPs [2, 1]. The packing algorithm of Allen-Zhu and Orecchia in [1] is particularly surprising, as it exploits the linear coupling framework to achieve both width-independence and Nesterov-like acceleration, which is widely believed to be very difficult, and is the first success in a long line of works in this area.

  The particular motivation for our work is a striking discrepancy between bounds provided for packing and covering LPs in [1]. In particular, they provide a $(1-\epsilon)$-approximation solver for the packing problem in $\tilde{O}(N/\epsilon)$, but they are only able to obtain $\tilde{O}(N/\epsilon^{1.5})$ for the covering problem. In the case of covering, they are unable to use the linear coupling method to achieve Nesterov-like acceleration, and even to get width-independence the authors need to integrate some ad-hoc and complicated techniques. This discrepancy between results for packing and covering LPs is rare, due to the duality between them. Filling this gap is of particular interests, as not being able to do so would suggest some fundamental structural differences between the two problems.

- Our main technical contribution is a novel diameter reduction method for fractional covering LPs that helps resolve this discrepancy. Recall that the smoothness parameter, e.g., Lipschitz constant, and the diameter of the feasible region are the two most natural limiting factors for most gradient based optimization algorithms. Indeed, many applications of general first-order optimization techniques can be attributed to the existence of norms or proximal setups for the specific problems that gives both good smoothness and diameter properties. In the particular case of coordinate descent algorithms based on the linear coupling idea, we additionally need good coordinate-wise diameter properties to achieve accelerated convergence.

  This is easy to accomplish for packing problems, but it is not easy to do for covering problems, and this is the difference that leads to the $\tilde{O}(1/\epsilon^{0.5})$ discrepancy between packing and covering algorithms in previous work [1]. Our diameter reduction method for general covering problems is based on dimension lifting, which transforms the covering problem space to a higher dimensional space, and the feasible region in the lifted space has both good global diameter bounds with respect to the canonical norm for accelerated stochastic coordinate descent (as is needed generally [10, 1]) as well as good coordinate-wise diameter bounds (as is needed for linear coupling [1]). Thus, it is likely of interest more generally for combinatorial optimization problems.

Once the diameter reduction is achieved, covering LP shares all the essential properties necessary to achieve both width-independence and Nesterov-like acceleration as in the case of packing problems, and fits elegantly into the scheme and analysis from [1] that was developed for packing LPs. We obtain improved $\tilde{O}(N/\epsilon)$ results for covering LPs, and this provides a unified acceleration method (unified in the sense that it is with the same algorithm and almost identical analysis) for both packing and covering LPs.

We will start in Section 2 with a discussion of some selected technical ideas and challenges from previous work. Then, in Section 3 we will present our main technical contribution, a novel diameter reduction method for any covering LP of the form given in (1). Finally, in Section 4 we describe how to combine this with previous work to obtain a unified acceleration method for packing and covering problems.

2 High-level Description of Challenges

At a high level, we (as well as Allen-Zhu and Orecchia [2, 1]) use the same two-step approach of Nesterov [9]. The first step involves smoothing, which transforms the constrained problem into a *smooth* objective function with trivial or no constraints. By smooth, we mean that the gradient of the objective function has some property in the flavor of Lipschitz continuity. Once smoothing is accomplished, the second step uses one of several first order methods for convex optimization in order to obtain an approximate solution to the smoothed objective. Standard applications of this approach usually lead to width-dependent algorithms, where the width enters the performance analysis as the magnitude of the gradients.

The first width-independent result following the optimization approach in [2] achieves width-independence by truncating the gradient, thus effectively reducing the width to 1. The algorithm uses, in a white-box way, the coupling of mirror descent and gradient descent from [15], where the progress from gradient descent covers the loss incurred by the truncation of the gradient (see Eqn. (7) below for the precise formulation of this loss), thus achieving width-independence. However, the role of gradient descent in the coupling is limited to width-independence, but not acceleration.

To improve the sequential packing solver in [2] with convergence $\tilde{O}(1/\epsilon^3)$ to $\tilde{O}(1/\epsilon)$, the same authors in [1] apply a stochastic coordinate descent method based on the linear coupling idea. Barring the difference between Lipschitz and local Lipschitz continuity, the results in [1] can be viewed as a variant of accelerated coordinate descent method [10]. There are two places where the algorithm achieves an improvement over prior packing-covering results.

- One factor of improvement is due to the better coordinate-wise Lipschitz constant over the full dimensional Lipschitz constant. Intuitively, in the case of packing or covering, the gradient of variable x_i depends on the penalties of constraints involving x_i, which further depend on all the variables in those constraints. As a result, if we move all the variables simultaneously, we can only take a small step before changing the gradient of x_i drastically. Sometimes coordinate descent comes with a downside, since if we update one variable each iteration, computing n partial derivatives in n iterations can be much more expensive than computing all the n partial derivatives in the same iteration. However, it can be shown in the case of packing and covering LPs, there is no such computation overhead.
- The other factor of improvement comes from Nesterov-like acceleration. In addition to giving width-independence as in [2], the gradient descent also covers the regret term incurred by the mirror descent step (see Eqn. (7) below for the precise formulation of this regret), which is the key insight from the original linear coupling result [15] to reproduce Nesterov's accelerated convergence. It turns out that nice diameter properties are necessary for the latter to be possible. On a high level, the regret incurred by mirror descent is proportional to its step size, which has an upper-bound proportional to the coordinate-wise diameter of the feasible region. On the other hand, the progress made by the gradient descent step is also proportional to its step size, which is inversely proportional to the Lipschitz parameter. For both packing and covering problems, the

coordinate-wise Lipschitz parameter of x_i is proportional to $1/\|A_{:i}\|_\infty$, as $\|A_{:i}\|_\infty$ captures the impact of x_i on the values of the constraints, which determine the gradient of x_i. This works out particularly well for packing problems, since the packing constraints $Ax \le \vec{1}$ impose a natural coordinate-wise diameter of $x_i^* \le 1/\|A_{:i}\|_\infty$ on the feasible region, which aligns the gradient descent step size and mirror descent step size, making the coupling possible to accelerate. The same small coordinate-wise diameter is also crucial to get good global diameter for the proximal setup used in mirror descent, which is necessary for mirror descent to achieve good convergence.

The combination of gradient truncation, stochastic coordinate descent, and acceleration due to the nice diameter properties lead to the $\tilde{O}(N/\epsilon)$ solver for the packing LP [1].

Shifting to solvers for the covering LP, one obvious obstacle to reproducing the packing result is we no longer have the small diameters. Indeed, a naive coordinate-wise upper bound from the covering constraints only gives $x_i^* \le 1/\min_j\{A_{ji} : A_{ji} > 0\}$, which is far from sufficient to give acceleration as the packing solver in [1]. The authors instead go back to the setup in their earlier work [2], where linear coupling only gives width-independence. The authors use a negative-width technique as in [3] (Theorem 3.3 with $l = \sqrt{\epsilon}$), that leads to the (improved, but still worse than for packing) $\tilde{O}(1/\epsilon^{1.5})$ convergence rate.

To get an $\tilde{O}(1/\epsilon)$ solver for the covering LP, it seems crucial to relate the gradient descent step and mirror descent step the same way as in the packing solver in [1]. Thus, we will work directly to reduce the coordinate-wise diameter. Our main result (presented next in Section 3) is a general diameter reduction method to achieve the same diameter property as in the packing solver, and this enables us (in Section 4) to extend all the crucial ideas of the packing solver in [1], as outlined in this section, to get a covering solver with running time $\tilde{O}(N/\epsilon)$.

3 Diameter Reduction Method for General Covering Problems

Given any covering LP of the form in (1), characterized by a matrix A, we formulate an equivalent covering LP with good diameter properties. This will involve lifting the instance to higher dimensional space by adding variables and redundant constraints. On a high level, as we discussed in last section, the obstacle for covering problems lies in the discrepency between the large coordinate-wise diameter of the feasible region and the small gradient descent step size. Our answer is essentially for each variable to create multiple copies with different resolutions. Certain copies will be in charge of searching over larger regions, but for them we modify their coefficients in the lifted space to allow larger gradient descent steps. We use $i \in [n]$ to denote the indices of the variables (i.e., columns of A) and $j \in [m]$ to denote the indices of constraints (i.e., rows of A). For ease of comparison with [1], and since our unified approach for both packing and covering uses their packing solver and a similar analysis, we use the same notation whenever possible.

For any $i \in [n]$, let

$$r_i \stackrel{\text{def}}{=} \frac{\max_j\{A_{ji} : A_{ji} > 0\}}{\min_j\{A_{ji} : A_{ji} > 0\}},$$

be the ratio between the largest non-zero coefficient and the smallest non-zero coefficient of variable x_i in all constraints, and let $n_i \stackrel{\text{def}}{=} \lceil \log r_i \rceil$. We first duplicate each original variable n_i times to obtain $\bar{x}_{(i,l)}, i \in [n], l \in [n_i]$ as the new variables. In terms of the coefficient matrix, we now have a new matrix, call it $\bar{A} \in \mathbb{R}_{\ge 0}^{m \times (\sum_i n_i)}$, which contains n_i copies of the i-th column $A_{:i}$. We denote a column of $\bar{A}$ by the tuple (i, l) with $l \in [n_i]$. Obviously,

the covering LP given by $\bar{A}$ is equivalent to the original covering LP given by A. Adding additional copies of variables, however, will allow us to improve the diameter. To reduce the diameter of this new covering LP, we further decrease some of the coefficients in $\bar{A}$, and we put upper bounds on the variables. In particular, for j, i, l, we have

$$\bar{A}_{j,(i,l)} = \min\{A_{j,i}, 2^l \min_j\{A_{ji} : A_{ji} > 0\}\}, \tag{3}$$

and for variable $\bar{x}_{(i,l)}$, we add the constraint

$$\bar{x}_{(i,l)} \leq \frac{2}{2^l \min_j\{A_{ji} : A_{ji} > 0\}}. \tag{4}$$

The next lemma shows that the covering LP given by $\bar{A}$ and the covering LP given by A are equivalent.

▶ **Lemma 1.** *The covering LP of A and the covering LP of $\bar{A}$ have the same optimal value* OPT. *Furthermore, there exists an optimal solution of the covering LP of $\bar{A}$ inside the region specified by* (4).

Proof. Let $\overline{\text{OPT}}$ be the optimal of the LP given by the covering constraints of $\bar{A}$ and the coordinate-wise upper-bounds in (4). We need to show $\text{OPT} = \overline{\text{OPT}}$. Given any feasible solution $\bar{x}$, consider the solution x where $x_i = \sum_{l=1}^{n_i} \bar{x}_{(i,l)}$. It is obvious $\vec{1}^T x = \vec{1}^T \bar{x}$, and $Ax \geq \vec{1}$, as coefficients in $\bar{A}$ are no larger than coefficients in A. Thus $\text{OPT} \leq \overline{\text{OPT}}$.

For the other direction, consider any feasible x. For each i, we can assume without loss of generality that

$$x_i \leq \frac{1}{\min_j\{A_{ji} : A_{ji} > 0\}}.$$

Let l_i be the largest index such that

$$x_i \leq \frac{2}{2^{l_i} \min_j\{A_{ji} : A_{ji} > 0\}},$$

and then let

$$\bar{x}_{(i,l)} = \begin{cases} x_i & \text{if } l = l_i \\ 0 & \text{if } l \neq l_i \end{cases}.$$

By construction, $\bar{x}$ satisfies all the upper bounds described in (4). Furthermore, for constraint j, we must have $\bar{A}_{j:}\bar{x} \geq 1$. Since for any i, $\bar{A}_{j,(i,l_i)}$ differs from A_{ji} only when $A_{ji} > 2^{l_i} \min_j\{A_{ji} : A_{ji} > 0\}$, and we must have $l_i < n_i$ in this case by definition of n_i, which gives $\bar{x}_{(i,l_i)} = x_i \geq \frac{1}{2^{l_i} \min_j\{A_{ji} : A_{ji} > 0\}}$ by our choice of l_i being the largest possible. Then we know $\bar{A}_{j,(i,l_i)} = 2^{l_i} \min_j\{A_{ji} : A_{ji} > 0\}$, so the j-th constraint is satisfied. Thus $\text{OPT} \geq \overline{\text{OPT}}$, and we can conclude $\text{OPT} = \overline{\text{OPT}}$. ◀

Given the equivalence of the covering LP defined by $\bar{A}$ and that defined by A, we now point out that the seemingly-redundant constraints of (4) turn out to be crucial. The reason is that we can search over a feasible region with nice diameter properties necessary to tap the full power of the linear coupling method. In particular, we can rewrite the constraints (4) to be

$$\bar{x}_{(i,l)} \leq \frac{2}{\|\bar{A}_{:(i,l)}\|_\infty}.$$

For any i, this is the same upper bound on $\bar{x}_{(i,l)}$ for $l < n_i$ (consider the row $j^* = \text{argmax}_j\{A_{ji}, A_{ji} > 0\}$), and it is a relaxation on $\bar{x}_{(i,n_i)}$.

The price we pay for this diameter improvement is that the new LP defined by $\bar{A}$ is larger than that defined by A. Two comments on this are in order. First, by Observation 3, r_i is bounded by n^2m/ϵ^2, and so the diameter reduction step only increases the problem size by $O(\log(mn/\epsilon))$. Second, we have presented our diameter reduction as an explicit pre-processing step so we can use one unified optimization algorithm (Algorithm 1 below) for both packing and covering, but in practice the diameter reduction would not have to be carried out explicitly. It can equivalently be implemented implicitly within the algorithm (a trivially-modified version of Algorithm 1 below) by randomly choosing a scale after picking the coordinate i and then computing $\bar{A}_{j,(i,l)}$ in (3) by shifting bits on the fly.

Given this reduction, in the rest of the paper, when we refer to the covering LP, we will implicitly be referring to the diameter reduced version, and we have the additional guarantee that there exists an optimal solution x^* to (1) such that

$$0 \leq x_i^* \leq \frac{2}{\|A_{:i}\|_\infty} \quad \forall i \in [n]. \tag{5}$$

4 An Accelerated Solver for (Packing and) Covering LPs

In this section, we will show covering LPs fit neatly into the scheme and analysis developed for packing LPs in [1], thus establishing a unified acceleration method for packing and covering problems. To motivate this, recall that for packing problems of the form (2), bounds of the form (5) automatically follow from the packing constraints $Ax \leq \vec{1}$. For readers familiar with the packing LP solver in [1], it should be plausible that—once we have this diameter property—the same stochastic coordinate descent optimization scheme will lead to a $\tilde{O}(N/\epsilon)$ covering LP solver.

In Section 4.1, we'll present some preliminaries and describe how we perform smoothing on the original covering objective function; and then in Section 4.2, we'll present the main algorithm. This algorithm involves a mirror descent step, that will be described in Section 4.3, a gradient descent step, that will be described in Section 4.4, and a careful coupling between the two, that will be described in Section 4.5. Finally, in Section 4.6, we will describe how to ensure we start at a good starting point. Some of the following results are technically-tedious but conceptually-straightforward extensions of analogous results from [1], and some of the results are restated from [1]; we defer most of the proofs to the full version.

4.1 Preliminaries and Smoothing the Objective

To start, let's assume that $\min_{j\in[m]} \|A_{j:}\|_\infty = 1$. This assumption is without loss of generality: since we can simply scale A for this to hold without sacrificing approximation quality. With this assumption, the following lemma holds.

▶ **Lemma 2.** $\text{OPT} \in [1, m]$.

With OPT being at least 1, the error we introduce later in the smoothing step will be small enough that the smoothing function approximates the covering LP well enough with respect to ϵ around the optimum.

▶ **Observation 3.** *It can be shown that to obtain a $(1+O(\epsilon))$-approximation, we can eliminate entries smaller than $\frac{\epsilon}{mn}$ and entries larger than $\frac{n}{\epsilon}$ from matrix A.*

We will turn the covering LP objective into a smoothed objective function $f_\mu(x)$, as used in [4, 2, 1], and we are going to find a $(1+\epsilon)$-approximation of the covering LP by approximately minimizing $f_\mu(x)$ over the region

$$\Delta \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : 0 \le x_i \le \frac{3}{\|A_{:i}\|_\infty}\}.$$

The function $f_\mu(x)$ is

$$f_\mu(x) \stackrel{\text{def}}{=} \vec{1}^T x + \max_{y \ge 0}\{y^T(\vec{1} - Ax) + \mu H(y)\},$$

and it is a smoothed objective in the sense that it turns the covering constraints into soft penalties, with $H(y)$ being a regularization term. Here, we use the generalized entropy $H(y) = -\sum_j y_j \log y_j + y_j$, where μ is the smoothing parameter balancing the penalty and the regularization. It is straightforward to compute the optimal y, and write $f_\mu(x)$ explicitly, as stated in the following lemma.

▶ **Lemma 4.** *$f_\mu(x) = \vec{1}^T x + \mu \sum_{j=1}^m p_j(x)$, where $p_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}(1 - (Ax)_j))$.*

Optimizing $f_\mu(x)$ over Δ gives a good approximation to OPT, in the following sense. If we let x^* be an optimal solution satisfying (5), and $u^* \stackrel{\text{def}}{=} (1+\epsilon/2)x^* \in \Delta$, then we have the properties in the following lemma.

▶ **Lemma 5.** *Setting the smoothing parameter $\mu = \frac{\epsilon}{4\log(nm/\epsilon)}$, we have*

1. *$f_\mu(u^*) \le (1+\epsilon)\,\mathrm{OPT}$.*
2. *$f_\mu(x) \ge (1-\epsilon)\,\mathrm{OPT}$ for any $x \ge 0$.*
3. *For any $x \ge 0$ satisfying $f_\mu(x) \le 2\,\mathrm{OPT}$, we must have $Ax \ge (1-\epsilon)\vec{1}$.*
4. *If $x \ge 0$ satisfies $f_\mu(x) \le (1+O(\epsilon))\,\mathrm{OPT}$, then $\frac{1}{1-\epsilon}x$ is a $(1+O(\epsilon))$-approximation to the covering LP.*
5. *The gradient of $f_\mu(x)$ is*

$$\nabla f_\mu(x) = \vec{1} - A^T p(\vec{x}) \quad \text{where} \quad p_j(x) \stackrel{\text{def}}{=} \exp(\frac{1}{\mu}(1 - (Ax)_j),$$

and $\nabla_i f_\mu(x) = 1 - \sum_j A_{ji} p_j(x) \in [-\infty, 1]$.

Although $f_\mu(x)$ gives a good approximation to the covering LP, $f_\mu(x)$ doesn't have the necessary Lipschitz-smoothness property due to the fast changing nature of exponential functions. However, $f_\mu(x)$ is *locally Lipschitz continuous*, in a sense quantified by the following lemma, and so we have a good improvement with a gradient step within certain range.

▶ **Lemma 6.** *Let $L \stackrel{\text{def}}{=} \frac{4}{\mu}$, for any $x \in \Delta$, and $i \in [n]$*

1. *If $\nabla_i f_\mu(x) \in (-1, 1)$, then for all $|\gamma| \le \frac{1}{L\|A_{:i}\|_\infty}$, we have*

$$|\nabla_i f_\mu(x) - \nabla_i f_\mu(x + \gamma\,\mathbf{e}_i)| \le L\|A_{:i}\|_\infty |\gamma|.$$

2. *If $\nabla_i f_\mu(x) \le -1$, then for all $\gamma \le \frac{1}{L\|A_{:i}\|_\infty}$, we have*

$$\nabla_i f_\mu(x + \gamma\,\mathbf{e}_i) \le (1 - \frac{L\|A_{:i}\|_\infty}{2}|\gamma|)\nabla_i f_\mu(x).$$

We call $L\|A_{:i}\|_\infty$ the *coordinate-wise local Lipschitz constant*. The significance of Lemma 6 is that for covering LPs the coordinate-wise local Lipschitz constant is inversely proportional to the coordinate-wise diameter. (This fact has been established previously for the case of packing LPs [1].)

4.2 An Accelerated Coordinate Descent Algorithm

Algorithm 1 Accelerated stochastic coordinate descent for both packing and covering

Input: $A \in \mathbb{R}^{m\times n}_{\geq 0}, x^{\text{start}} \in \Delta, f_\mu, \epsilon$ **Output**: $y_T \in \Delta$

$\mu \leftarrow \frac{\epsilon}{4\log(nm/\epsilon)}, L \leftarrow \frac{4}{\mu}, \tau \leftarrow \frac{1}{8nL}$

$T \leftarrow \lceil 8nL\log(1/\epsilon)\rceil = \tilde{O}(\frac{n}{\epsilon})$

$x_0, y_0, z_0 \leftarrow x^{\text{start}}, \alpha_0 \leftarrow \frac{1}{nL}$

for $k = 1$ to T **do**

  $\alpha_k \leftarrow \frac{1}{1-\tau}\alpha_{k-1}$

  $x_k \leftarrow \tau z_{k-1} + (1-\tau)y_{k-1}$

  Select $i \in [n]$ uniformly at random.

  ▷ Gradient truncation:

  Let $(\xi_k^{(i)})_i \leftarrow \begin{cases} -\mathbf{e}_i & \nabla_i f_\mu(x_k) < -1 \\ \nabla_i f_\mu(x_k)\cdot \mathbf{e}_i & \nabla_i f_\mu(x_k) \in [-1,1] \\ \mathbf{e}_i & \nabla_i f_\mu(x_k) > 1 \end{cases}$

  ▷ Mirror descent step:

  $z_k \leftarrow z_k^{(i)} \stackrel{\text{def}}{=} \text{argmin}_{z\in\Delta}\{V_{z_{k-1}}(z) + \langle z, n\alpha_k \xi_k^{(i)}\rangle\}$.

  ▷ Gradient descent step:

  $y_k \leftarrow y_k^{(i)} \stackrel{\text{def}}{=} x_k + \frac{1}{n\alpha_k L}(z_k^{(i)} - z_{k-1})$

end for

return y_T.

We will now show that the accelerated coordinate descent used in packing LP solver in [1] also works as a covering LP solver, with appropriately-chosen starting points and smoothed objective functions. Consider Algorithm 1, which is our main accelerated stochastic coordinate descent for both packing and covering. Note for both packing and covering LPs, we give $\Delta = \{x \in \mathbb{R}^n : 0 \leq x_i \leq \frac{3}{\|A_{:i}\|_\infty}\}$ as the input feasible region. The correctness of this algorithm and its running time guarantees for the packing problem have already been nicely presented in [1], and so here we will focus on the covering problem.

Our main result is summarized in the following theorems.

▶ **Theorem 7.** *With x^{start} computable in time $\tilde{O}(N)$ to be specified later, Algorithm 1 outputs y_T satisfying $\mathbb{E}[f_\mu(y_T)] \leq (1+6\epsilon)\,\text{OPT}$, and the running time is $\tilde{O}(N/\epsilon)$.*

A standard application of Markov bound gives the following corollary.

▶ **Corollary 8.** *There is a algorithm that, with probability at least $9/10$, computes a $(1+O(\epsilon))$-approximation to the fractional covering problem and has $\tilde{O}(N/\epsilon)$ expected running time.*

Before proceeding with our proof of these theorems, we discuss briefly the optimization scheme from [1] we will use. First, the A-norm is used as the proximal setup for mirror descent, where

$$\|x\|_A = \sqrt{\sum_i \|A_{:i}\|_\infty x_i^2}, \tag{6}$$

The corresponding distance generating function is $w(x) = \frac{1}{2}\|x\|_A^2$, and the Bregman divergence is $V_x(y) = \frac{1}{2}\|x-y\|_A^2$.[1]

[1] In particular, w is a 1-strongly convex function with respect to $\|\cdot\|_A$, and $V_x(y) \stackrel{\text{def}}{=} w(y) - \langle \nabla w(x), y - x\rangle - w(x)$. See [15] for a detailed discussion of mirror descent as well as and several interpretations.

Next, observe that Algorithm 1 works as follows. Each iteration integrates a mirror descent step and a gradient descent step. The standard analysis of mirror descent gives a convergence of $\frac{1}{\epsilon^2}$, and it depends on the width of the problem. Here is how the coupling of gradient descent and mirror descent achieves both width-independence and linear-rate acceleration.

- To eliminate the width from the convergence rate, the gradient $\nabla_i f_\mu(x_k)$ is split into the small component, $\xi_k^{(i)} = \max\{-1, \nabla_i f_\mu(x_k)\}\,\mathbf{e}_i$, and the large component, $\eta_k^{(i)} = \nabla_i f_\mu(x_k)\,\mathbf{e}_i - \xi_k^{(i)}$. Only the small component $\xi^{(i)}$ is given to the mirror descent step, and thus the width is effectively 1. However, the truncation incurs loss from the large component, as the mirror descent only acts on the small component. The progress from the gradient descent step is used to cover that loss.
- In order to get to $1/\epsilon$ convergence, recall that the $1/\epsilon^2$ in the convergence of mirror descent is largely due to the regret term accumulated along all iterations of mirror descent. The progress from the gradient step also covers the regret from the mirror descent step (see Eqn. (7) below for the precise formulation of this loss and regret). This enables the coupling to get Nesterov-like acceleration using the same approach in [15].

Before we moving to formalize the above discussion, here are some lemmas about the algorithm. The first lemma says that the gradient step we take is always valid (i.e., in Δ), which is crucial in the sense that we need the step length to be at least $\frac{1}{n\alpha_k L}$ of the mirror descent step length for the coupling to work.

▶ **Lemma 9.** *We have $x_k, y_k, z_k \in \Delta$ for all $k = 0, 1, \ldots, T$.*

The second lemma is clearly crucial to achieve the nearly linear time $\tilde{O}(N/\epsilon)$ algorithm.

▶ **Lemma 10.** *Each iteration can be implemented in expected $O(N/n)$ time.*

4.3 Mirror Descent Step

We now analyze the mirror descent step of Algorithm 1:

$$z_k \leftarrow z_k^{(i)} \stackrel{\text{def}}{=} \operatorname*{argmin}_{z\in\Delta}\{V_{z_{k-1}}(z) + \langle z, n\alpha_k \xi_k^{(i)}\rangle\}.$$

▶ **Lemma 11.** $\langle n\alpha_k \xi_k^{(i)}, z_{k-1} - u^*\rangle \le n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle + V_{z_{k-1}}(u^*) - V_{z_k}(u^*)\,.$

Also, we note that the mirror descent step, defined above in a variational way, can be explicitly written as

1. $z_k^{(i)} \leftarrow z_{k-1}$
2. $z_k^{(i)} \leftarrow z_k^{(i)} - n\alpha_k \xi_k^{(i)}/\|A_{:i}\|_\infty$
3. If $z_{k,i}^{(i)} < 0, z_{k,i}^{(i)} \leftarrow 0$; if $z_{k,i}^{(i)} > 3/\|A_{:i}\|_\infty, z_{k,i}^{(i)} \leftarrow 3/\|A_{:i}\|_\infty$.

4.4 Gradient Descent Step

We now analyze the gradient descent step of Algorithm 1. In particular, from the explicit formulation of the mirror descent step, we have that $|z_{k,i}^{(i)} - z_{k-1,i}| \le \frac{n\alpha_k|\xi_k^{(i)}|}{\|A_{:i}\|_\infty}$, which gives

$$|y_{k,i}^{(i)} - x_{k,i}| = \frac{1}{n\alpha_k L}|z_{k,i}^{(i)} - z_{k-1,i}| \le \frac{|\xi_k^{(i)}|}{L\|A_{:i}\|_\infty}.$$

The gradient step we take is within the local region, and so Lemma 6 applies. We bound the progress from the gradient descent step in the following lemma.

▶ **Lemma 12.** $f_\mu(x_k) - f_\mu(y_k^{(i)}) \ge \frac{1}{2}\langle \nabla f_\mu(x_k), x_k - y_k^{(i)}\rangle\,.$

4.5 Coupling of Gradient and Mirror Descent

Here, we will analyze the coupling between the gradient descent and mirror descent steps. This and the next section will give a proof of Theorem 7.

As we take steps on random coordinates, we will write the full gradient as

$$\nabla f_\mu(x_k) = \mathbb{E}_i[n\nabla_i f_\mu(x_k)] = \mathbb{E}_i[n\eta_k^{(i)} + n\xi_k^{(i)}].$$

As discussed earlier, we have the small component $\xi_k^{(i)} \in (-1, 1)\,\mathbf{e}_i$ and the large component $\eta_k^{(i)} = \nabla_i f_\mu(x_k) - \xi_k^{(i)} \in (-\infty, 0]\,\mathbf{e}_i$. We put the gradient and mirror descent steps together, and we bound the gap to optimality at iteration k:

$$\begin{aligned}
\alpha_k(f_\mu(x_k) - f_\mu(u^*)) \leq& \langle \alpha_k \nabla f_\mu(x_k), x_k - u^*\rangle \\
=& \langle \alpha_k \nabla f_\mu(x_k), x_k - z_{k-1}\rangle + \langle \alpha_k \nabla f_\mu(x_k), z_{k-1} - u^*\rangle \\
=& \langle \alpha_k \nabla f_\mu(x_k), x_k - z_{k-1}\rangle + \mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^*\rangle] \\
&+ \mathbb{E}_i[\langle n\alpha_k \xi_k^{(i)}, z_{k-1} - u^*\rangle] \\
=& \frac{1-\tau}{\tau}\alpha_k \langle \nabla f_\mu(x_k), y_{k-1} - x_k\rangle + \mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^*\rangle] \\
&+ \mathbb{E}_i[\langle n\alpha_k \xi_k^{(i)}, z_{k-1} - u^*\rangle] \\
\leq& \frac{1-\tau}{\tau}\alpha_k (f_\mu(y_{k-1}) - f_\mu(x_k)) + \mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^*\rangle] \\
&+ \mathbb{E}_i[n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle + V_{z_{k-1}}(u^*) - V_{z_k^{(i)}}(u^*)].
\end{aligned}$$

The first line is due to convexity. The fourth line is due to $x_k = \tau z_{k-1} + (1-\tau)y_{k-1}$, so $\tau(x_k - z_{k-1}) = (1-\tau)(y_{k-1} - x_k)$. The last line is by Lemma 11.

We need to use the progress from the gradient step given in Lemma 12 to cover the loss from $\eta_k^{(i)}$, and the regret from the mirror descent step:

$$\underbrace{\mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^*\rangle]}_{\text{loss from } \eta_k^{(i)}} + \underbrace{\mathbb{E}_i[n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle]}_{\text{regret from mirror descent}}, \tag{7}$$

The following lemma crucially relies on the nice coordinate-wise diameters of the feasible region Δ.

▶ **Lemma 13.** *The (scaled) progress from the gradient step covers both the loss from gradient truncation and the regret incurred by the mirror descent step*

$$\mathbb{E}_i[\langle n\alpha_k \eta_k^{(i)}, z_{k-1} - u^*\rangle] + \mathbb{E}_i[n^2\alpha_k^2 L\langle \xi_k^{(i)}, x_k - y_k^{(i)}\rangle] \leq \mathbb{E}_i[8n\alpha_k L(f_\mu(x_k) - f_\mu(y_k^{(i)}))].$$

Now we can show this gives Nesterov-like acceleration. We have

$$\begin{aligned}
\alpha_k(f_\mu(x_k) - f_\mu(u^*)) \leq& \frac{1-\tau}{\tau}\alpha_k (f_\mu(y_{k-1}) - f_\mu(x_k)) + \mathbb{E}_i[8n\alpha_k L(f_\mu(x_k) - f_\mu(y_k^{(i)})] \\
&+ \mathbb{E}_i[V_{z_{k-1}}(u^*) - V_{z_k^{(i)}}(u^*)].
\end{aligned}$$

With our choice of $\tau = \frac{1}{8nL}$, $\alpha_k = \frac{1}{1-\tau}\alpha_{k-1}$, we get

$$-\alpha_k f_\mu(u^*) \leq 8nL\alpha_{k-1} f_\mu(y_{k-1}) - \mathbb{E}_i[8nL\alpha_k f_\mu(y_k^{(i)})] + \mathbb{E}_i[V_{z_{k-1}}(u^*) - V_{z_k^{(i)}}(u^*)].$$

Telescoping the above inequality [2] along $k = 1, \ldots, T$, we get

$$\mathbb{E}[8nL\alpha_T f_\mu(y_T)] \leq \sum_{k=1}^{T} \alpha_k f_\mu(u^*) + 8nL\alpha_0 f_\mu(y_0) + V_{z_0}(u^*),$$

and thus

$$\mathbb{E}[f_\mu(y_T)] \leq \frac{\sum_{k=1}^{T} \alpha_k}{8nL\alpha_T} f_\mu(u^*) + \frac{\alpha_0}{\alpha_T} f_\mu(y_0) + \frac{1}{8nL\alpha_T} V_{z_0}(u^*).$$

We have $\sum_{k=1}^{T} \alpha_k = \alpha_T \sum_{k=0}^{T-1} (1 - \frac{1}{8nL})^k = 8nL\alpha_T(1 - (1 - \frac{1}{8nL})^T) \leq 8nL\alpha_T$, and by our choice of $T = \lceil 8nL \log(1/\epsilon) \rceil$, we also have

$$\frac{\alpha_0}{\alpha_T} = (1 - \frac{1}{8nL})^T \leq \epsilon, \frac{1}{8nL\alpha_T} \leq \frac{\epsilon}{8nL\alpha_0} = \frac{\epsilon}{8},$$

and thus

$$\mathbb{E}_i[f_\mu(y_T)] \leq f_\mu(u^*) + \epsilon f_\mu(y_0) + \frac{\epsilon}{8} V_{z_0}(u^*). \tag{8}$$

4.6 Finding a Good Starting Point

From (8), we see a good starting point $y_0 = x^{\text{start}}$ for Algorithm 1 is a point that is not too far away from the optimal in terms of the function value (i.e small $f_\mu(y_0)$), and not too far away from u^* in A-norm (i.e. small $V_{z_0}(u^*)$). For packing problems, starting with the all-0's vector will work, but this will not work for covering problems. Instead, for covering problems, we will show now a good enough x^{start} can be obtained in $\tilde{O}(N)$.

To do so, recall that we can get a 2-approximation $x^{\#}$ to the original covering LP in time $\tilde{O}(N)$ using various nearly linear time covering solvers, e.g., those of [7, 4, 6, 14]. Without loss of generality, we can assume $x_i^{\#} \in [0, \frac{2}{\|A_{:i}\|_\infty}]$, since we can use the diameter reduction process as specified in Lemma 1 to get a equivalent solution satisfying the conditions. Then, we have the following lemma.

▶ **Lemma 14.** *Let* $x^{\text{start}} = (1 + \epsilon/2)x^{\#}$*, we have* $x^{\text{start}} \in \Delta$*,* $f_\mu(x^{\text{start}}) \leq 4\,\text{OPT}$*, and* $V_{x^{\text{start}}}(u^*) \leq 6\,\text{OPT}$

It is now clear from (8) that we have

$$\mathbb{E}_i[f_\mu(y_T)] \leq f_\mu(u^*) + \epsilon f_\mu(y_0) + \frac{\epsilon}{8} V_{z_0}(u^*) \leq (1+\epsilon)\,\text{OPT} + 4\epsilon\,\text{OPT} + \epsilon\,\text{OPT} = (1+6\epsilon)\,\text{OPT}\,.$$

Thus, we have the approximation guarantee in Theorem 7. The running time follows directly from Lemma 10 and $T = \tilde{O}(n/\epsilon)$.

---

[2] More accurately, the telescoping works on

$$-\alpha_k f_\mu(u^*) \leq 8nL\alpha_{k-1}\mathbb{E}_{I_{k-1}}[f_\mu(y_{k-1})] - \mathbb{E}_{I_k}[8nL\alpha_k f_\mu(y_k^{(i)})] + \mathbb{E}_{I_{k-1}}[V_{z_{k-1}}(u^*)] - \mathbb{E}_{I_k}[V_{z_k^{(i)}}(u^*)].$$

where I_k is all the random coordinate choices made through the first iteration till k-th iteration. The final expectation on $f_\mu(y_T)$ is over all the T random choices.

References

1. Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC'15, pages 229–236, 2015. Newer version available at `http://arxiv.org/abs/1411.1124`.
2. Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 1439–1456, 2015.
3. Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. `doi:10.4086/toc.2012.v008a006`.
4. Baruch Awerbuch and Rohit Khandekar. Stateless distributed gradient descent for positive linear programs. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 691–700, 2008. `doi:10.1145/1374376.1374476`.
5. Lisa Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 1001–1010, 2004. URL: `http://dl.acm.org/citation.cfm?id=982792.982942`.
6. Christos Koufogiannakis and Neal E. Young. A nearly linear-time PTAS for explicit fractional packing and covering linear programs. *Algorithmica*, 70(4):648–674, 2014. `doi:10.1007/s00453-013-9771-6`.
7. Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 448–457, 1993. `doi:10.1145/167088.167211`.
8. Arkadi Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004. `doi:10.1137/S1052623403425629`.
9. Yurii Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005. `doi:10.1007/s10107-004-0552-5`.
10. Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012. `doi:10.1137/100802001`.
11. Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 495–504, 1991. `doi:10.1109/SFCS.1991.185411`.
12. James Renegar. Efficient first-order methods for linear programming and semidefinite programming. *CoRR*, abs/1409.5832, 2014. URL: `http://arxiv.org/abs/1409.5832`.
13. Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 538–546, 2001. `doi:10.1109/SFCS.2001.959930`.
14. Neal E. Young. Nearly linear-time approximation schemes for mixed packing/covering and facility-location linear programs. *CoRR*, abs/1407.3015, 2014. URL: `http://arxiv.org/abs/1407.3015`.
15. Zeyuan Allen Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *CoRR*, abs/1407.1537, 2014. URL: `http://arxiv.org/abs/1407.1537`.

Random-Edge Is Slower Than Random-Facet on Abstract Cubes

Thomas Dueholm Hansen[*1] **and Uri Zwick**[†2]

1 **Department of Computer Science, Aarhus University, Aarhus, Denmark**
tdh@cs.au.dk

2 **Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
zwick@tau.ac.il

Abstract

Random-Edge and Random-Facet are two very natural randomized pivoting rules for the simplex algorithm. The behavior of Random-Facet is fairly well understood. It performs an expected *sub-exponential* number of pivoting steps on *any* linear program, or more generally, on any *Acyclic Unique Sink Orientation* (AUSO) of an arbitrary polytope, making it the fastest known pivoting rule for the simplex algorithm. The behavior of Random-Edge is much less understood. We show that in the AUSO setting, Random-Edge is *slower* than Random-Facet. To do that, we construct AUSOs of the n-dimensional hypercube on which Random-Edge performs an expected number of $2^{\Omega(\sqrt{n \log n})}$ steps. This improves on a $2^{\Omega(\sqrt[3]{n})}$ lower bound of Matoušek and Szabó. As Random-Facet performs an expected number of $2^{O(\sqrt{n})}$ steps on *any* n-dimensional AUSO, this established our result. Improving our $2^{\Omega(\sqrt{n \log n})}$ lower bound seems to require radically new techniques.

1998 ACM Subject Classification G.1.6 Optimization – Linear Programming, G.2.1 Combinatorics – Combinatorial Algorithms

Keywords and phrases Linear programming, the Simplex Algorithm, Pivoting rules, Acyclic Unique Sink Orientations

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.51

1 Introduction

Linear programming and the simplex algorithm. Linear programs (LPs) [3, 4, 28, 32] are among the most important mathematical optimization problems. The *simplex algorithm* (Dantzig [4]) is one of the most widely used methods for solving linear programs. It starts at a *vertex* of the polytope corresponding to the linear program. (We assume, for simplicity, that the LP is feasible, bounded, non-degenerate, that no two vertices have the same objective value, and that an initial vertex of the polytope is available.) If the current vertex is not optimal, then at least one of the *edges* incident to it leads to a neighboring vertex with a larger objective value. A *pivoting rule* determines which one of these vertices to move to. The simplex algorithm, with any pivoting rule, is guaranteed to find an optimal solution of an LP in a finite number of steps.

Unfortunately, with essentially all known *deterministic* pivoting rules, the simplex method requires an *exponential* number of steps on some LPs (see Klee and Minty [24] and [1, 2, 7,

* Thomas Dueholm Hansen was supported by the Carlsberg Foundation, grant no. CF14-0617.
† Uri Zwick was supported by BSF grant no. 2012338 and by the The Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 51; pp. 51:1–51:14

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

15, 18]). While there are polynomial time algorithms for solving LP problems, most notably the *ellipsoid algorithm* (Khachian [23]) and *interior point methods* (Karmarkar [22]), these algorithms are not *strongly* polynomial, i.e., their running time, in the unit-cost model, depends on the number of *bits* in the representation of the coefficients of the LP, and not just on the combinatorial size of the problem, i.e., the number of variables and constraints. The question whether there exists a strongly polynomial time algorithm for solving linear programs is of great theoretical importance.

Randomized pivoting rules. Kalai [20] and Matoušek, Sharir and Welzl [29] devised a *randomized* pivoting rule, Random-Facet, that performs a *subexponential* number of pivoting steps, in expectation, on *any* LP. In particular, for an LP with n variables and $O(n)$ constraints, the expected number of pivoting steps performed by Random-Facet is at most $2^{O(\sqrt{n \log n})}$. A slightly improved version that performs an expected number of at most $2^{O(\sqrt{n})}$ pivoting steps on such LPs was recently obtained by the authors [17]. This improved version is currently the fastest known pivoting rule for the simplex algorithm.

Perhaps the most natural randomized pivoting rule is Random-Edge. When there are several improving edges from the current vertex, simply choose one of them uniformly at random. The best upper bound for Random-Edge on general LPs is exponential (Gärtner and Kaibel [13]). Whether this can be improved to a subexponential upper bound, similar to the one available for Random-Facet, is an intriguing open problem.

Friedmann et al. [8, 9, 11], building on results of Friedmann [6] and Fearnley [5], obtained a $2^{\tilde{\Omega}(n^{1/3})}$ lower bound on the expected number of pivoting steps performed by Random-Facet on LPs that correspond to *shortest paths* problems. [1] Friedmann et al. [9] also constructed LPs on which Random-Edge performs an expected number of $2^{\tilde{\Omega}(n^{1/4})}$ pivoting steps. We believe that this lower bound is not tight. However, as the $2^{\tilde{\Omega}(n^{1/4})}$ lower bound is already quite complicated, improving it seems to be a hard task. As a step in this direction we obtain a $2^{\Omega(\sqrt{n \log n})}$ lower bound for Random-Edge in the more general setting of *Acyclic Unique Sink Orientations* (AUSOs). In particular, this shows that in the AUSO setting, Random-Edge is slower than Random-Facet.

Acyclic Unique Sink Orientations (AUSOs). Each bounded LP has a polytope associated with it. The geometric vertices and edges of this polytope define a combinatorial graph. The objective function of the LP defines an *orientation* of the edges of this graph; An edge connecting vertices u and v is directed from u to v if and only if v has a better objective value. (We assume that no two vertices have the same objective value.) This orientation is clearly *acyclic*. It follows easily from the properties of LPs that every subgraph corresponding to a geometric *face* of the polytope has a *unique sink*, i.e., a unique vertex with no outgoing edges. The unique sink of the whole graph is then the optimal vertex of the LP.

This naturally motivates the definition of *Acyclic Unique Sink Orientations* (AUSOs). Let $G(P)$ be that graph that corresponds to a polytope P. An orientation of $G(P)$ is said to be an AUSO if and only if it is acyclic and any subgraph that corresponds to a face of P has a unique sink. The term AUSO was introduced by Szabó and Welzl [34]. The same notion was considered before, however, under several different names. Williamson Hoke [36] refers to them as *Completely Unimodal Numberings*, while Kalai [19, 20, 21] refers to them as *Abstract*

[1] In [9] we obtained a $2^{\tilde{\Omega}(\sqrt{n})}$ lower bound for a one-permutation variant of Random-Facet and erroneously claimed that the expected number of steps performed by this variant is equal to the expected number of steps performed by Random-Facet. Unfortunately, as we point out in [10], this is *not* the case.

Objective Functions. AUSOs provide an appealing abstraction of LPs. The subexponential upper bound on the behavior of Random-Facet only relies on the AUSOs properties and is thus valid also in the AUSO setting.

Of special interest are AUSOs of the n-dimensional *hypercube* whose properties we review in Section 2. In the sequel we restrict our attention to such AUSOs. Gärtner [12] proved a $2^{O(n^{1/2})}$ upper bound on the complexity of Random-Facet on AUSOs, and Matoušek [27] obtained an essentially tight $2^{\Omega(n^{1/2})}$ lower bound. (The improved version of Random-Facet of [17], which is not specialized to cubes, essentially matches Gärtner's bound. It is an interesting open problem whether a similar lower bound can be obtained for this algorithm.) Matoušek and Szabó [30, 31] obtained a lower bound of $2^{\Omega(n^{1/3})}$ on the complexity of Random-Edge on AUSOs. We improve their lower bound to $2^{\Omega(\sqrt{n \log n})}$, thus showing that Random-Edge is slower than Random-Facet on AUSOs. Hansen et al. [16] obtained a 1.8^n upper bound for Random-Edge on AUSOs. There is a large gap between the available upper and lower bounds. However, the $2^{\Omega(\sqrt{n \log n})}$ lower bound that we obtain is the best lower bound that can be obtained using current techniques. Improving it would require the use of *non-layered* AUSOs which are not yet known to exist.

Organization of paper. In the next section we review some basic properties of AUSOs of the Boolean hypercube. In Section 3 we describe a randomized product construction of AUSOs that plays a central role in the lower bound of Matoušek and Szabó [30, 31] as well as in our improved lower bound of $2^{\Omega(\sqrt{n})}$ which is described in Section 4. Section 5 contains an analysis of a random walk with *reshuffles* on a simple *path AUSO*. This analysis is needed to complete the proof of the lower bound presented in Section 4. In Section 6 we make the final push and improve the lower bound from $2^{\Omega(\sqrt{n})}$ to $2^{\Omega(\sqrt{n \log n})}$. In Section 7 we note that the AUSOs used to prove all known lower bounds for Random-Edge are *layered* and explain why it is unlikely that layered AUSOs could be used to obtain further improved lower bounds. A similar observation was made independently by Gärtner and Thomas [14]. We end in Section 8 with some concluding remarks and open problems.

2 Acyclic Unique Sink Orientations of the Hypercube

In this section we review the definition of AUSOs and describe some of their basic properties. We also introduce a simple *path* AUSO which plays a central role in our lower bound.

The n-cube is the undirected graph $C_n = (V_n, E_n)$, where $V_n = \{0,1\}^n$ and $E_n = \{\{\mathbf{x}, \mathbf{y}\} \mid \mathbf{x}, \mathbf{y} \in V_n, d_H(\mathbf{x}, \mathbf{y}) = 1\}$, where $d_H(\mathbf{x}, \mathbf{y})$ is the *Hamming distance* between $\mathbf{x}$ and $\mathbf{y}$. Every string $\mathbf{s} \in \{0,1,*\}^n$ defines a *subcube* of C_n induced by the vertex set $V_s = \{\mathbf{x} \in \{0,1\}^n \mid \mathbf{x}_i = \mathbf{s}_i \text{ or } \mathbf{s}_i = *\}$. The subcube defined by $\mathbf{s}$ is clearly isomorphic to a cube whose dimension is the number of $*$'s in $\mathbf{s}$.

An *orientation* of the n-cube is a directed graph obtained by orienting each edge $\{\mathbf{x}, \mathbf{y}\} \in E_n$ either from $\mathbf{x}$ to $\mathbf{y}$ or from $\mathbf{y}$ to $\mathbf{x}$. An orientation can be specified using a mapping $A : V_n \to \{0,1\}^n$ that satisfies the condition $A(\mathbf{x})_i \neq A(\mathbf{x} \oplus i)_i$, for every $\mathbf{x} \in V_n$. [2] The i-th edge of $\mathbf{x}$, i.e., $\{\mathbf{x}, \mathbf{x} \oplus i\}$ is directed *away* from $\mathbf{x}$ if and only if $A(\mathbf{x})_i = 1$. (Some authors refer to A as the *out-map* of the orientation. We consider it to be the orientation itself.) An orientation of a cube clearly induces orientations on all its subcubes. An orientation is

[2] Here, $A(\mathbf{x})_i$ denotes the i-th bit of $A(\mathbf{x}) \in \{0,1\}^n$, and $\mathbf{x} \oplus i$ is a shorthand for $\mathbf{x} \oplus \mathbf{e}_i$, where $\mathbf{e}_i = 0^{i-1}10^{n-i}$ is the i-th unit vector.

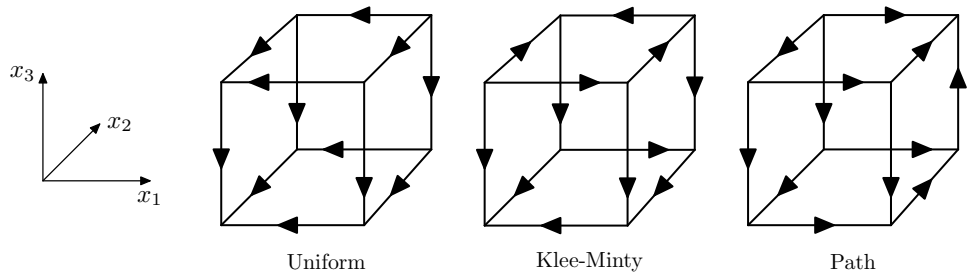


Figure 1 The uniform, Klee-Minty and the path AUSOs in dimension 3.

acyclic if there are no directed cycles. A *sink* of an orientation A of C_n is a vertex $\mathbf{x} \in V_n$ with no outgoing edges, i.e., $A(\mathbf{x}) = 0^n$. An acyclic orientation clearly has at least one sink, but it may have several.

▶ **Definition 1** (AUSOs)**.** An orientation A of the n-cube is an n-AUSO if and only if it is *acyclic* and if every subcube has a *unique sink* under A.

Perhaps the simplest AUSO is the *uniform* AUSO defined as follows $U(\mathbf{x}) = \mathbf{x}$. The sink of this AUSO is clearly 0^n. A more interesting and famous AUSO is the Klee-Minty cube [24] defined as follows $KM(\mathbf{x})_i = x_i \oplus x_{i+1} \oplus \cdots x_n$, for $i = 1, 2, \ldots, n$, where $\mathbf{x} = x_1 x_2 \ldots x_n$. The sink is again 0^n. (It is an instructive exercise to verify that the Klee-Minty cube is indeed an AUSO.) The Klee-Minty cube plays a prominent role in the lower bound of Matoušek and Szabó [30, 31]. One of the major steps in obtaining our improved lower bound is replacing the Klee-Minty cubes used in the lower bound of Matoušek and Szabó [30, 31] by the following much simpler AUSO.

▶ **Definition 2** (Path AUSO)**.** The *path* n-AUSO is defined as follows

$$P(\mathbf{x})_i = x_i \oplus (x_1 \wedge x_2 \wedge \cdots \wedge x_{i-1}) \ .$$

In other words, if $\mathbf{x} = x_1 x_2 \ldots x_n \in \{0,1\}^n$, then the i-th edge of $\mathbf{x}$ is directed away from $\mathbf{x}$ if and only if $x_1 x_2 \ldots x_i = 1^{i-1}0$ or $x_i = 1$ but $x_1 x_2 \ldots x_{i-1} \neq 1^{i-1}$. (Informally, a bit wants to become, or remain, a 1 if and only if all the bits preceding it are 1.) The sink of the path AUSO is clearly at 1^n. The uniform, Klee-Minty and the path AUSO for $n = 3$ are shown in Figure 1. The path AUSO is also used in a building block in [9, 11]. A realization of the path AUSO as a shortest paths problem in a weighted directed graph is given in Figure 2. Each n-bit string $\mathbf{x} = x_1 x_2 \ldots x_n$ corresponds to a *tree* of directed paths directed towards the target vertex 0. If $x_i = 0$, then $(i, 0)$ is included in the tree, otherwise $(i, i-1)$ is included in the tree. The i-th edge of $\mathbf{x}$ in the n-cube is directed away from $\mathbf{x}$ if and only if switching the outgoing edge of i in the tree reduces the distance to the target of at least one vertex. With this interpretation, it is clear that the path AUSO is an AUSO. (It is of course also easy to prove it directly.)

▶ **Lemma 3.** *The path AUSO is indeed an AUSO.*

Schurr and Szabó [33] describe two useful methods of constructing new AUSOs from existing ones. The first method takes an n-AUSO and *expands* each one of its 2^n vertices into an m-AUSO, resulting in an $(n+m)$-AUSO. (Each one of the 2^n vertices may be expanded into a different m-AUSO.) Schurr and Szabó [33] call it the *blowup* construction. We view it as a *product* of AUSOs.

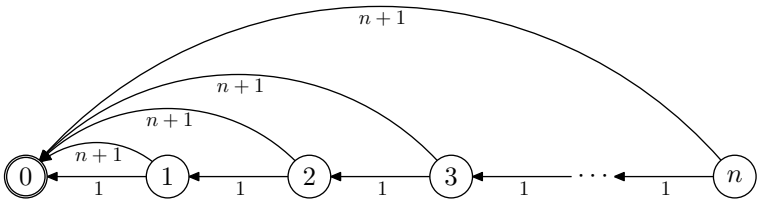


Figure 2 A realization of the path AUSO as a shortest paths problem in a weighted directed graph. Numbers attached to edges are edge weights.

▶ **Definition 4** (Product of AUSOs [33]). Let A be an n-AUSO, and let $m > 1$. For every $\mathbf{x} \in \{0,1\}^n$, let $B_{\mathbf{x}}$ be an m-AUSO. Define an $(n+m)$-AUSO $C = A \times \langle B_{\mathbf{x}} \rangle$ as follows:

$$C(\mathbf{x}, \mathbf{y}) \;=\; (A(\mathbf{x}), B_{\mathbf{x}}(\mathbf{y})) \;, \quad \text{for } \mathbf{x} \in \{0,1\}^n, \mathbf{y} \in \{0,1\}^m \;.$$

The proof that the product AUSO is indeed an AUSO is straightforward. (For the details, see [33].) Before describing the second method we need another definition.

▶ **Definition 5** (Hypersink [33]). A subcube of an AUSO is a *hypersink* if and only if all edges between vertices of the subcube and vertices outside the subcube are directed towards the subcube.

▶ **Lemma 6** (Hypersink replacement [33]). *Let A be an AUSO, let B be a subcube of A which is a hypersink, and let B' be an AUSO of the same dimension as B. Then, the AUSO A' obtained by modifying the orientation of the edges within B so that the resulting orientation is isomorphic to B' is again an AUSO.*

The proof of the lemma is again straightforward and can be found in [33]. Finally, the following well-known lemma will be used in Section 6.

▶ **Lemma 7** (Unique outmaps [34]). *Let A be an n-AUSO. Then, for every $\mathbf{x}_1 \neq \mathbf{x}_2 \in \{0,1\}^n$ we have $A(\mathbf{x}_1) \neq A(\mathbf{x}_2)$. In particular, the number of vertices of A of out-degree k is $\binom{n}{k}$.*

3 Random-Edge on Randomized Products

Matoušek and Szabó [30, 31] introduced the following simple and elegant *randomized product* construction. Let A be an n-AUSO and let B be an m-AUSO. Assume, for simplicity, that the sinks of A and B are at 1^n and 1^m, respectively. For every $\mathbf{x} \in \{0,1\}^n$, let $B_{\mathbf{x}}$ be a version of B in which the m coordinates are *randomly permuted*, each permutation being equally likely. (Note that the sink of $B_{\mathbf{x}}$ is still at 1^m.) Consider now the product $A \times \langle B_{\mathbf{x}} \rangle$. It is easy to check that the copy of A corresponding to $\mathbf{y} = 1^m$ is a hypersink of $A \times \langle B_{\mathbf{x}} \rangle$. We replace this copy of A by a *random translation* A' of A. A random translation of an n-AUSO A is obtained by choosing a random vertex $\mathbf{x}_0 \in \{0,1\}^n$ and making it the sink of an AUSO A' isomorphic to A. More precisely, we let $A'(\mathbf{x}) = A(\mathbf{x} \oplus \mathbf{x}_0)$. A formal definition of randomized products follows.

▶ **Definition 8** (Randomized product [30, 31]). Let A be an n-AUSO and let B be an m-AUSO whose sinks are at 1^n and 1^m, respectively. For every $\mathbf{x} \in \{0,1\}^n$, let $B_{\mathbf{x}}$ be a randomly permuted version of B, and let A' be a random translation of A. Then, the *randomized product* $C = A \times_R B$ is a distribution over AUSOs defined as follows:

$$C(\mathbf{x}, \mathbf{y}) \;=\; \begin{cases} (A(\mathbf{x}), B_{\mathbf{x}}(\mathbf{y})) & \text{, if } \mathbf{y} \neq 1^m \;, \\ (A'(\mathbf{x}), B_{\mathbf{x}}(\mathbf{y})) & \text{, otherwise .} \end{cases}$$

As $C = A \times_R B$ is obtained by performing a product of AUSOs followed by a hypersink replacement, it is clearly an AUSO, or rather a probability distribution over AUSOs.

Consider now the behavior of Random-Edge on $C = A \times_R B$, starting from some vertex $(\mathbf{x}, \mathbf{y})$, where $\mathbf{x} \in \{0,1\}^n$, $\mathbf{y} \in \{0,1\}^m$. Assume at first that $\mathbf{x} \neq 1^n$ and $\mathbf{y} \neq 1^m$. Each step of Random-Edge either changes a coordinate of $\mathbf{x}$ or of $\mathbf{y}$. If we only consider the steps in which $\mathbf{x}$ changes, and look only at $\mathbf{x}$, then as long as $\mathbf{y} \neq 1^m$, we get a standard Random-Edge walk on A.

The random walk on $\mathbf{y}$ induced by the Random-Edge walk on $C = A \times_R B$ is more interesting. When $\mathbf{x}$ is fixed, $\mathbf{y}$ performs a standard Random-Edge walk on $B_{\mathbf{x}}$. But, when a step from $\mathbf{x}$ to $\mathbf{x}'$ in A is made, the walk on $\mathbf{y}$ finds itself in a new AUSO $B_{\mathbf{x}'}$. Due to the acyclicity of A, $B_{\mathbf{x}'}$ was never visited before, and is thus completely random. The resulting random walk on $\mathbf{y}$ is what Matoušek and Szabó [30, 31] call a *random walk with reshuffles* on B. Such a random walk is likely to be much longer than a standard Random-Edge walk on B as the random reshuffles tend to destroy progress made since the last reshuffle.

When the random walk with reshuffles on B reaches its sink $\mathbf{y} = 1^m$, the hypersink A' is entered, and $\mathbf{y}$ remains fixed. Note that $\mathbf{y} = 1^m$ may be reached either before $\mathbf{x}$ reaches 1^n, or after it does. We prefer the second option, as then the Random-Edge walk on A ran to completion. Our construction will ensure that this happens with very high probability. As A' is a random translation of A, the ensuing random walk on A', even if it starts at $\mathbf{x} = 1^n$, is equivalent to a Random-Edge walk starting at a *random* vertex of A. Thus, under appropriate conditions, a Random-Edge walk on $C = A \times_R B$ is expected to be at least *twice* as long as a Random-Edge walk on A itself. This is the crux of the matter.

We next consider the probability of a reshuffle in a random walk with reshuffles on B. This probability must be high to make the random walk long. If the current vertex in $C = A \times_R B$ is $(\mathbf{x}, \mathbf{y})$ and there are s and t outgoing edges from $\mathbf{x}$ in A and $\mathbf{y}$ in B, respectively, then a reshuffle occurs with probability $s/(s+t)$, as Random-Edge on $C = A \times_R B$ chooses each outgoing edge with equal probability and each step in A causes a reshuffle of B. We let Random-Reshuffle$_k$ denote a random walk with reshuffles on B induced by a Random-Edge walk on $C = A \times_R B$, conditioned on all visited vertices of A having outdegree at least k. The probability of a reshuffle when $\mathbf{y}$ has outdegree t in B is then *at least* $k/(k+t)$.

To ensure that reshuffles occur with sufficiently high probability, we need a slight generalization of the randomized product construction.

▶ **Definition 9** (k-fold randomized product [30, 31])**.** Let A be an n-AUSO and let B be an m-AUSO whose sinks are at 1^n and 1^m, respectively, and let $k \geq 1$. For every $\mathbf{x} \in \{0,1\}^n$ and $1 \leq i \leq k$, let $B_{\mathbf{x},i}$ be a randomly permuted version of B, and let A' be a random translation of A. Then, the *k-fold randomized product* $C = A \times_R^k B$ is a distribution over AUSOs defined as follows:

$$C(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_k) \;=\; \begin{cases} (A(\mathbf{x}), B_{\mathbf{x},1}(\mathbf{y}_1), \ldots, B_{\mathbf{x},k}(\mathbf{y}_k)) & , \text{ if } \mathbf{y}_1 \neq 1^m \wedge \cdots \wedge \mathbf{y}_k \neq 1^m \;, \\ (A'(\mathbf{x}), B_{\mathbf{x},1}(\mathbf{y}_1), \ldots, B_{\mathbf{x},k}(\mathbf{y}_k)) & , \text{ otherwise} \;. \end{cases}$$

It is not difficult to check that for every random choice $C = A \times_R^k B$ is indeed an AUSO. Matoušek and Szabó [30, 31] need to apply the k-fold randomized product with $k = n^{1/3}$. We only need $k = 2$. As a result, the lower bound is improved from $2^{\Omega(\sqrt[3]{n})}$ to $2^{\Omega(\sqrt{n})}$. Using $k = 1$ is actually enough, but the probabilistic analysis is slightly more complicated. The details will appear in the full version of the paper.

▶ **Lemma 10** ([30, 31])**.** *Let A be an n-AUSO, let B be an m-AUSO and let $k > 1$. Suppose that the probability that the* Random-Edge *walk on A, starting at a random vertex, reaches*

a vertex with outdegree less than k in less than T steps is at most p. Suppose that the probability that a Random-Reshuffle$_k$ *walk on B, starting at a random vertex, reaches the sink in less than T steps is at most q. Then, the probability that the* Random-Edge *walk on $C = A \times_R^k B$, starting at a random vertex, reaches a vertex of outdegree less than k in less than $2T$ steps is at most $2p + kq$.*

Proof. Partition the Random-Edge walk on $C = A \times_R^k B$, starting at a random vertex, into two stages. The first stage ends when one of $B_1, \ldots, B_k$ reaches its sink, i.e., when a state of the form $(\mathbf{x}, \mathbf{y}_1, \ldots, \mathbf{y}_k)$, where $\mathbf{y}_i = 1^m$, for some $1 \leq i \leq k$, is reached. When the first stage ends, the orientation on the first n-coordinates changes to A', a random translation of A, and the second stage begins. The second stage lasts until the sink of $C = A \times_R^k B$ is reached.

The probability that the first stage lasts less than T steps is at most $p + kq$. Indeed, during the first T steps on C, at most T steps on A are performed. Thus, the probability that the Random-Edge walk on A reaches a vertex of outdegree less than k is at most p. As long as the walk on A visits vertices of outdegree at least k, the induced random walks on $B_1, \ldots, B_k$ are Random-Reshuffle$_k$ walks. Thus, the probability that a specific B_i reaches its sink during the first T steps on C is at most q, and the claim follows. During the first phase, there is at least one improving switch in every B_i, thus the outgedree of each visited vertex is at least k.

In the second stage, the orientation on the first n-coordinates changes to A'. As A' is a random translation of A, this is equivalent to starting a Random-Edge walk on A at a random starting vertex. The probability that this random walk reaches a vertex of outdegree less than k in less than T is steps is at most p.

Thus, the probability that the Random-Edge walk on $C = A \times_R^k B$, starting at a random vertex, reaches a vertex of outdegree less than k in less than $2T$ steps is at most $(p + kq) + p = 2p + kq$. ◀

4 The Lower Bound

A distribution of AUSOs on which Random-Edge makes $2^{\Omega(\sqrt{n})}$ steps, with high probability, is obtained by iterating the randomized product construction of the previous section. We follow the footsteps of Matoušek and Szabó [30, 31] making one crucial change; We use the *path* AUSO, defined in Section 2, instead of the Klee-Minty cube, as the main building block. Using the path AUSO allows us to simplify the proof and improve the lower bound. The following lemma, whose proof is given in the next section, is one of the main technical contributions of this paper.

▶ **Lemma 11.** *Let P_m be the path m-AUSO. There are constants $\alpha, \beta > 0$ such that the probability that* Random-Reshuffle$_2$ *on P_m, starting from a random vertex, performs less than $2^{\alpha m}$ steps before reaching the sink is at most $2^{-\beta m}$.*

Let m be an integer, and let $\ell = \gamma m$, where $\gamma < \min\{\alpha, \beta\}$, where α, β are the constants in Lemma 11. Let A_0 be an arbitrary m-AUSO. For concreteness, we let $A_0 = P_m$, the *path* AUSO of Section 2. We construct a sequence of $A_0, A_1, \ldots, A_\ell$ of AUSOs, where $A_i = A_{i-1} \times_R^2 P_m$ is an $(2i+1)m$-AUSO, for $0 \leq i \leq \ell$. Note that A_ℓ is of dimension $(2\ell+1)m = O(m^2)$. The following lemma, which easily implies a $2^{\Omega(\sqrt{n})}$ lower bound, claims that with high probability, Random-Edge performs at least $2^{\Omega(m)}$ steps when started at a random vertex of A_ℓ. We give an improved version of the lemma in Section 6.

▶ **Lemma 12.** *The probability that* Random-Edge *performs less than 2^ℓ steps when started at a random vertex of A_ℓ, where $\ell < \alpha m$, is at most $4 \cdot 2^{\ell - \beta m}$.*

Proof. Let p_i be the probability that RANDOM-EDGE, started at a random vertex of A_i, performs less than 2^i steps before reaching a vertex of outdegree less than 2. Note that $p_0 = (m+1)2^{-m}$ is just the probability that the random starting vertex of A_0 has outdegree at most 1, which by Lemma 7 is exactly $(m+1)2^{-m}$.

Let q_i be the probability that the RANDOM-RESHUFFLE$_2$ walk on P_m, in which the probability of reshuffle from a vertex of outdegree j is at least $2/(j+2)$, reaches the sink in less than 2^i steps, when started at a random vertex. Clearly, for every $1 \le i \le \ell$ we have $q_i \le q_\ell$. As $\ell \le \alpha m$, Lemma 11 implies that $q_\ell \le 2^{-\beta m}$.

By Lemma 10, we have $p_i \le 2(p_{i-1} + q_{i-1})$. It follows easily by induction that $p_i \le 2^i p_0 + 2(2^i - 1)q_i$, for $1 \le i \le \ell$. Thus, $p_\ell \le 2^\ell (m+1)2^{-m} + 2^{\ell+1}2^{-\beta m} \le 4 \cdot 2^{\ell - \beta m}$. ◀

As a corollary, we get:

▶ **Theorem 13.** *There exist n-AUSOs and appropriate starting points from which* RANDOM-EDGE *performs $2^{\Omega(n^{1/2})}$ steps with probability at least $1 - 2^{-\delta\sqrt{n}}$, for some $\delta > 0$.*

5 Random Walk with Reshuffles on the Path AUSO

In this section we provide a proof of Lemma 11, completing the proof of our first lower bound.

Let $\mathbf{y} \in \{0,1\}^m$ be a state of RANDOM-RESHUFFLE$_2$ on the path AUSO P_m. Let k be the *weight*, i.e., the number of 1's in $\mathbf{y}$ and let i be the number of leading 1's in $\mathbf{y}$. For example, if $m = 7$ and $\mathbf{y} = 1100101$, then $k = 4$ and $i = 2$. Also let $j = k - i$, the number of non-leading 1's. We say that $\mathbf{y}$ is of *type* (i, j). The outdegree of a state of type (i, j) is clearly $j + 1$.

From a state $\mathbf{y}$ of type (i, j) of RANDOM-RESHUFFLE$_2$ on P_m, there is a reshuffle with a probability of at *least* $2/(j+3)$, as all states of type (i, j) are of outdegree $j + 1$. Several reshuffles may occur in a row, but they have exactly the same effect as a single reshuffle.

If a state of weight k is reshuffled, the obtained state is of type $(k-j, j)$, where $0 \le j \le k$, with probability $a_{k,j} = \binom{m-(k-j+1)}{j} / \binom{m}{k}$. (Among the $\binom{m}{k}$ binary strings of length m and weight k, there are exactly $\binom{m-(k-j+1)}{j}$ strings that start with $1^{k-j}0$.) Clearly $\sum_{j=0}^{k} a_{k,j} = 1$.

When a reshuffle occurs, we initially consider only the type of the state obtained. We *delay* the decision as to which state exactly we are in. (All states belonging to the type are equally likely.) This simplifies the analysis, as we only need to consider $O(m^2)$ types ($\binom{m+1}{2} + 1$ to be exact) rather than 2^m possible states.

Suppose that we are now in a state of type (i, j), where $k = i + j$. One or more reshuffles are performed with a probability of at least $2/(j+3)$, and then a standard move is performed. Let (i', j'), where $k' = i' + j'$, be the type of the new state obtained. Note that $k' = k - 1$ or $k' = k + 1$, as a reshuffle does not change the weight and a single move either increases or decreases the weight by 1. We show that the probability of a weight increase, i.e., $k' = k + 1$ is bounded by some constant $c < \frac{1}{2}$. As the weight of a random starting vertex is close to $m/2$ and as the weight of the sink is m, it would follow that an exponential number of steps are needed, with high probability, to reach the sink.

Let $\bar{p}_k$ be the probability of a weight increase from a state of weight k given that a reshuffle occurs. We have $\bar{p}_k = \sum_{j=0}^{k} \frac{a_{k,j}}{j+1}$, as the reshuffle generates a state of type $(k-j, j)$ with probability $a_{k,j}$ and the probability of a weight increase from such a state is $1/(j+1)$.

Let $p_{k,j}(s)$ be the probability of a weight increase from a state of type $(k-j, j)$, when the corresponding state in A has s outgoing edges. The probability of a reshuffle from a state of type $(k-j, j)$ is $\frac{s}{j+s+1}$, and the probability of no reshuffle is $\frac{j+1}{j+s+1}$. If there is no reshuffle,

the probability of a weight increase is $\frac{1}{j+1}$. If there is a reshuffle, then the probability of a weight increase is $\bar{p}_k$. Hence,

$$p_{k,j}(s) \;=\; \frac{j+1}{j+s+1}\cdot\frac{1}{j+1} + \frac{s}{j+s+1}\cdot\bar{p}_k \;=\; \frac{1}{j+s+1} + \frac{s}{j+s+1}\cdot\bar{p}_k\;.$$

We next upper bound $\bar{p}_k$, the probability of a weight increase following a reshuffle from a state of weight k.

▶ **Lemma 14.** $\bar{p}_k \;<\; \dfrac{m-k}{(k+1)(m-k-1)}$.

Proof. A simple manipulation of binomial coefficients yields:

$$\frac{a_{k,j}}{j+1} \;=\; \frac{\binom{m-k+j-1}{j}}{\binom{m}{k}}\frac{1}{j+1} \;=\; \frac{\binom{m-k+j-1}{j+1}}{\binom{m}{k}}\frac{1}{m-k-1} \;=\; \frac{1}{m-k-1}\frac{\binom{m}{k+1}}{\binom{m}{k}}a_{k+1,j+1}\;.$$

Hence,

$$\bar{p}_k = \sum_{j=0}^{k}\frac{a_{k,j}}{j+1} = \frac{1}{m-k-1}\frac{\binom{m}{k+1}}{\binom{m}{k}}\sum_{j=0}^{k}a_{k+1,j+1} < \frac{1}{m-k-1}\frac{\binom{m}{k+1}}{\binom{m}{k}} = \frac{m-k}{(k+1)(m-k-1)}.$$

◀

▶ **Lemma 15.** *Let (i,j) be a type of* Random-Reshuffle$_2$ *walk on the path AUSO P_m of weight $k=i+j$ satisfying $8 \le k \le m-9$. Then, the probability that the type obtained after one step of a* Random-Reshuffle$_2$ *walk, i.e., a reshuffle with a probability of at least $2/(j+3)$ followed by a standard move, is of weight $k+1$ is at most $\frac{5}{12}$.*

Proof. We need to show that $p_{k,j}(s) \le \frac{5}{12}$, for every $8 \le k \le m-9$, $0 \le j \le k$, and $s \ge 2$. Recall that $p_{k,j}(s) = \frac{j+1}{j+s+1}\cdot\frac{1}{j+1} + \frac{s}{j+s+1}\cdot\bar{p}_k$. For $8 \le k \le m-9$ we have $\bar{p}_k \le \frac{1}{8}$, as $\frac{1}{k+1} \le \frac{1}{9}$ while $\frac{m-k}{m-k-1} \le \frac{9}{8}$. If $j \ge 2$, then $p_{k,j}(s) \le \frac{1}{3}$, for every $s \ge 1$, as $\frac{1}{j+1} \le \frac{1}{3}$ and $\bar{p}_k \le \frac{1}{3}$. If $j=1$, then $p_{k,1}(s) \le \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{8} = \frac{5}{16}$, again for every $s \ge 1$. Finally, if $j=0$, then $p_{k,0}(s) \le \frac{1}{3} + \frac{2}{3}\cdot\frac{1}{8} = \frac{5}{12}$, for every $s \ge 2$. ◀

Lemma 15 does not hold for Random-Reshuffle$_1$ as for $j=0$ we relied on the assumption $s \ge 2$. This is why we need to use 2-fold randomized products. Alternatively, we can look at *two* steps of Random-Reshuffle$_1$ on P_m. This yields a negative drift from all types, including $(i,0)$, but the proof is slightly more complicated. The details will appear in the full version of the paper.

▶ **Definition 16** (Biased random walk)**.** Let $0 \le c < \frac{1}{2}$. A random process on states $\{0,1,\ldots,n\}$ is *c-bounded* if and only if whenever the process is in state i, where $i > 0$, then it moves to state $i+1$ with probability at most c, and to state $i-1$ with the complementary probability of at least $1-c$. From state 0, the process moves to state 1 with probability 1.

In the above definition, the transition probabilities may depend on the history of the random process, as well as on external factors. The following claim is well known. For completeness, we include a proof that follows the presentation in Levin, Peres and Wilmer [25] (Section 17.3.1).

▶ **Lemma 17.** *Let $0 \le c < \frac{1}{2}$. The probability that a c-bounded random process on $\{0,1,\ldots,n\}$ that starts at state 0 reaches state n in less than K steps is at most $K\left(\frac{c}{1-c}\right)^{n-1}$. In particular, for any $\alpha < \frac{1-c}{c}$, the probability that the random walk reaches n in less than α^{n-1} steps is at most $\left(\frac{\alpha c}{1-c}\right)^{n-1}$.*

Proof. Let us assume that the probability of moving from i to $i+1$ is exactly c. It is not difficult to show that this implies the more general claim.

Let X_t be a c-bounded random walk on the integers starting at 1, i.e., $X_0 = 1$. Let $r = (1-c)/c$. It is easy to check that r^{X_t} is a *martingale*. Consider the *stopping time* τ defined as the smallest t for which $X_t = 0$ or $X_t = n$. By the *Optional Stopping Theorem* we get that $\mathbb{E}[r^{X_\tau}] = \mathbb{E}[r^{X_0}] = r$. Let a be the probability that the random walk reaches n before it reaches 0. Then $\mathbb{E}[r^{X_\tau}] = ar^n + (1-a)$. Thus $ar^n + (1-a) = r$ and $a = \frac{r-1}{r^n-1} < (1/r)^{n-1}$.

Returning to the c-bounded walk on $\{0, 1, \ldots, n\}$ starting at 0, we note that the expected number of times that this random walk returns to 0 before visiting n is a geometric random variable with parameter a. Thus the probability that the random walk reaches n in less than K steps, and in particular less than K returns to 0, is at most $1-(1-a)^K < Ka < K\left(\frac{c}{1-c}\right)^{n-1}$. ◀

Putting everything together, we get a proof of Lemma 11.

Proof of Lemma 11. A Random-Reshuffle$_2$ walk on the path AUSO P_m induces a random walk on *types* (i, j). The random walk on types induces a random walk on *weights*. Lemma 15 claims that when $8 \le k \le m-9$, the random walk on the weights is $\frac{5}{12}$-bounded. The weight of a random starting point of Random-Reshuffle$_2$ is binomially distributed. By Chernoff bound, the probability that the starting state of the bounded walk on the weights starts at a weight larger than $(\frac{1}{2}+\delta)m$, for some $\delta > 0$, is exponentially small. By Lemma 17 the probability that the random walk moves from the random starting point to $m-9$ in at most an exponential number of steps is exponentially small. The existence of appropriate constants α, β follows easily. ◀

6 An Improved Lower Bound

The improved lower bound of $2^{\Omega(\sqrt{n \log n})}$ is obtained using essentially the same construction but with a strengthened analysis. We again let $A_0 = P_m$ and $A_i = A_{i-1} \times^2_R P_m$, for $i = 1, ..., \ell$, but this time we choose $\ell = \gamma m \log m$, for some $\gamma > 0$. We can still show that with high probability Random-Edge performs at least 2^ℓ steps on A_ℓ. Thus A_ℓ is an n-AUSO, where $n = 2\gamma m^2 \log m$, on which Random-Edge performs with high probability at least 2^ℓ steps, where $\ell = \gamma m \log m = \Omega(\sqrt{n \log n})$. This establishes the following theorem, which is the main result of this paper.

▶ **Theorem 18.** *There exist n-AUSOs and appropriate starting points from which* Random-Edge *performs $2^{\Omega(\sqrt{n \log n})}$ steps with probability at least $1 - 2^{-\delta\sqrt{n}}$, for some $\delta > 0$.*

To prove this result, we need a strengthened version Lemma 11. Lemma 11 states that there are constants $\alpha, \beta > 0$ such that the probability that Random-Reshuffle$_2$ on P_m, starting from a random vertex, performs less than $2^{\alpha m}$ steps before reaching the sink is at most $2^{-\beta m}$. The proof of Lemma 11 relied on the fact that from a state of type (i, j), the probability of a reshuffle is at least $2/(j+3)$. If the reshuffle probabilities are always $2/(j+3)$, then Lemma 11 is essentially tight. However, the reshuffle probability is $2/(j+3)$ only if the vertex $\mathbf{x} \in A$ has only two outgoing edges. By Lemma 7 this can happen at most $\binom{n}{2}$ times. More generally, let $s = \sqrt{n}$. The number of vertices in A of outdegree less than s is at most $\sum_{i=0}^{s} \binom{n}{i} \le n^s$.

▶ **Definition 19** (Random-Reshuffle$_k^{(N,s)}$)**.** A Random-Reshuffle$_k^{(N,s)}$ walk on an m-AUSO B is a random walk with reshuffles on B in which the reshuffle probability from a

vertex $\mathbf{y} \in \{0,1\}^m$ of outdegree t is at least $s/(s+t)$, except for at most N times in which the reshuffle probability is only guaranteed to be at least $k/(k+t)$.

By the above discussion, the random process on B induced by a RANDOM-EDGE walk on $A \times_R^2 B$ is a RANDOM-RESHUFFLE$_k^{(N,s)}$ process, with $N \leq n^s$. The following lemma is a strengthening of Lemma 11. The proof will appear in the full version of the paper.

▶ **Lemma 20.** *Let P_m be the path m-AUSO. There are constants $\alpha, \beta > 0$ such that the probability that* RANDOM-RESHUFFLE$_2^{(N,s)}$ *on P_m, where $s = m^{1/2}$, $N \leq m^{3s}$, starting from a random vertex, performs less than $2^{\alpha m \log m}$ steps before reaching the sink is at most $2^{-\beta m}$.*

We note that Lemma 20 is essentially best possible, which is one of the reasons we believe that improving our $2^{\Omega(\sqrt{n \log n})}$ lower bound will require new techniques. (See also Section 7.)

▶ **Lemma 21.** *Let B be any m-AUSO. Then, with high probability* RANDOM-RESHUFFLE$_k$ *on B makes at most $O((km)^m) = 2^{O(m \log(km))}$ steps before reaching its sink.*

Proof. Each vertex of B is of distance at most m from the sink (see, e.g., [26, 16]). Thus, if there are m consecutive steps with no reshuffles and the right edge in B is followed, then the sink is reached. This happens with a probability of at least $((k+1)m)^{-m}$. ◀

The proof of Lemma 20 relies on the following strengthening of Lemma 17. The proof will appear in the full version of the paper.

▶ **Lemma 22.** *Let $0 \leq c_1 < c_2 < \frac{1}{2}$. Consider a random walk on $\{0, 1, \ldots, n\}$ in which in each step a controller is allowed to flip a coin with success probability at most c_1, or a coin with success probability at most c_2. The controller is allowed to make the second choice at most N times. According to the outcome of the coin, the process moves from i to either $i-1$ or $i+1$. (From 0 the move is always to 1.) Then, the probability that the walk reaches n in less than K steps is at most $K\left(\frac{c_1}{1-c_1}\right)^{n/2-1} + N\left(\frac{c_2}{1-c_2}\right)^{n/2-1}$.*

Relying on these strengthened lemmas, Theorem 18 follows using essentially the same arguments used to prove Theorem 13.

7 Decomposable and Layered AUSOs

The AUSOs used above to obtain the $2^{\Omega(\sqrt{n \log n})}$ lower bound, as well as the AUSOs used by Matoušek and Szabó [30, 31] and by Friedmann et al. [9] are *decomposable*.

▶ **Definition 23** ((k,ℓ)-decomposable AUSO)**.** A $(k\ell)$-AUSO is said to be (k,ℓ)-decomposable if its coordinates can be partitioned into k blocks $B_1, \ldots, B_k$ each of size ℓ such that if $\mathbf{x}$ is the sink of A and $\mathbf{y}$ agrees with $\mathbf{x}$ in $B_i, B_{i+1}, \ldots, B_k$, for some $1 \leq i \leq k$, then all the edges in these coordinates are directed towards $\mathbf{y}$.

Decomposable AUSOs belong to a much wider class of *layered AUSOs*. A layered AUSO is an AUSO whose vertices can be partitioned into a relatively small number of layers such that from each non-sink vertex there is a relatively short directed path to a vertex of a lower layer, and such that no directed path may lead from a vertex to a vertex of a higher layer. More formally:

▶ **Definition 24** ((k,ℓ)-Layered AUSOs)**.** An n-AUSO A is said to be (k,ℓ)-*layered* if its vertices can be partitioned into disjoint *layers* $L_0, L_1, \ldots, L_k$ such that: *(i)* L_0 only contains the sink. *(ii)* If $\mathbf{x} \in L_i$, where $i > 0$, then there is a directed path of length at most ℓ in A from $\mathbf{x}$ to some vertex $\mathbf{y} \in L_j$, for $j < i$. *(iii)* If $\mathbf{x} \in L_i$, then there is no directed path in A from $\mathbf{x}$ to any vertex $\mathbf{y} \in L_j$, for $j > i$.

▶ **Lemma 25.** *If A is a (k,ℓ)-decomposable AUSO then it is also (k,ℓ)-layered.*

Proof. Let $B_1, B_2, \ldots, B_k$ be the partition of the coordinates of A and let $\mathbf{x}$ be the sink of A. For $i = 0, 1, \ldots k$, let L_i be the set of vertices that agree with $\mathbf{x}$ in all the coordinates of $B_{i+1}, B_{i+2}, \ldots, B_k$ and differ from $\mathbf{x}$ in some coordinate of B_i. It is easy to check that all conditions are met by using the fact that every vertex in a subcube of dimension ℓ has a path of length at most ℓ to the sink of that subcube (see, e.g., [26, 16]). ◀

We next show that Random-Edge is fairly quick on layered AUSOs.

▶ **Lemma 26.** *The expected number of steps that* Random-Edge *performs on a (k,ℓ)-layered n-AUSO, from any starting vertex, is at most $k\ell n^\ell$.*

Proof. We show that the expected number of steps per layer is at most ℓn^ℓ, which proves the lemma. From every vertex in the current layer there is at least one path of length ℓ to a lower layer. Random-Edge follows this path with a probability of at least $1/n^\ell$. Therefore the expected number of trials before successfully following such a path is at most n^ℓ, and since each trial uses at most ℓ steps, the total expected number of steps is at most ℓn^ℓ. ◀

We note that the n-AUSOs used to obtain our $2^{\Omega(\sqrt{n \log n})}$ lower bound are (k,ℓ)-layered for $k = O(\sqrt{n \log n})$ and $\ell = O(\sqrt{n/\log n})$. Lemma 26 therefore shows that the expected number of steps performed by Random-Edge on our AUSOs is at most $2^{O(\sqrt{n \log n})}$, which means that our analysis of these AUSOs is tight up to a constant factor in the exponent. Moreover, to improve the lower bound it is necessary to construct n-AUSOs that are not $(2^{O(\sqrt{n \log n})}, O(\sqrt{n/\log n}))$-layered, or, more generally, not (k,ℓ)-layered for any choice of k and ℓ such that $k\ell n^\ell = 2^{O(\sqrt{n \log n})}$. No such AUSOs are currently known to exist. Similarly, to improve on the 1.8^n upper bound of Hansen et al. [16] it is enough to prove that every n-AUSO is $(2^{cn}, dn/\log n)$-layered, for $c + d < 0.847 < \log_2 1.8$.

The lower bound of Friedmann et al. [9] uses AUSOs that simulate a binary counter. A t-bit counter is simulated by $(t, n/t)$-decomposable n-AUSO. The lower bound obtained using such AUSOs is roughly 2^t. As $(t, n/t)$-decomposable n-AUSO are also $(t, n/t)$-layered, we get using Lemma 26 that the best lower bound that can be obtained using such AUSOs is $\min\{2^t, n^{n/t+1}\}$, which attains a maximum value of about $2^{\Theta(\sqrt{n \log n})}$ when $t = \sqrt{n \log n}$.

A notion of the *niceness* of AUSOs and USOs, which is related to the notions of decomposable and layered AUSOs was defined by Welzl [35]. Gärtner and Thomas [14] have independently observed that the AUSOs used to obtain our lower bound are $\sqrt{n \log n}$-nice and therefore cannot be used to obtain further improved lower bounds.

8 Concluding remarks and open problems

We proved a $2^{\Omega(\sqrt{n \log n})}$ lower bound on the number of steps performed by Random-Edge on some AUSOs. The two obvious open problems are proving a $2^{o(n)}$ upper bound and an improved lower bound of $2^{\omega(\sqrt{n \log n})}$. We believe that radically new techniques would be needed to prove a lower bound of $2^{\omega(\sqrt{n \log n})}$. Another interesting open problem is improving the $2^{\Omega(n^{1/4})}$ lower bound of [9] for AUSOs that correspond to actual linear programs.

Acknowledgments. We would like to thank Mikkel Thorup for encouraging us to improve our lower bound from $2^{\Omega(\sqrt{n})}$ to $2^{\Omega(\sqrt{n \log n})}$, to Yuval Peres for pointing us to [25] for a proof of Lemma 17, and to Bernd Gärtner and Antonis Thomas for bringing the concept of niceness of AUSOs to our attention.

References

1. N. Amenta and G.M. Ziegler. Deformed products and maximal shadows of polytopes. In *Advances in Discrete and Computational Geometry*, pages 57–90, Providence, 1996. Amer. Math. Soc. Contemporary Mathematics 223.
2. D. Avis and V. Chvátal. Notes on Bland's pivoting rule. In *Polyhedral Combinatorics*, volume 8 of *Mathematical Programming Studies*, pages 24–34. Springer, 1978.
3. V. Chvátal. *Linear programming*. W. H. Freeman and Company, 1983.
4. G.B. Dantzig. *Linear programming and extensions*. Princeton University Press, 1963.
5. J. Fearnley. Exponential lower bounds for policy iteration. In *Proc. of 37th ICALP*, pages 551–562, 2010. doi:10.1007/978-3-642-14162-1_46.
6. O. Friedmann. An exponential lower bound for the latest deterministic strategy iteration algorithms. *Logical Methods in Computer Science*, 7(3), 2011. doi:10.2168/LMCS-7(3:23)2011.
7. O. Friedmann. A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games. In *Proc. of 15th IPCO*, pages 192–206, 2011. doi:10.1007/978-3-642-20807-2_16.
8. O. Friedmann, T.D. Hansen, and U. Zwick. A subexponential lower bound for the Random Facet algorithm for parity games. In *Proc. of 22nd SODA*, pages 202–216, 2011.
9. O. Friedmann, T.D. Hansen, and U. Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proc. of 43th STOC*, pages 283–292, 2011. doi:10.1145/1993636.1993675.
10. O. Friedmann, T.D. Hansen, and U. Zwick. Errata for: A subexponential lower bound for the random facet algorithm for parity games. *CoRR*, abs/1410.7871, 2014. URL: http://arxiv.org/abs/1410.7871.
11. O. Friedmann, T.D. Hansen, and U. Zwick. Random-facet and random-bland require subexponential time even for shortest paths. *CoRR*, abs/1410.7530, 2014. URL: http://arxiv.org/abs/1410.7530.
12. B. Gärtner. The Random-Facet simplex algorithm on combinatorial cubes. *Random Structures and Algorithms*, 20(3):353–381, 2002. doi:10.1002/rsa.10034.
13. B. Gärtner and V. Kaibel. Two new bounds for the Random-Edge simplex-algorithm. *SIAM J. Discrete Math.*, 21(1):178–190, 2007. doi:10.1137/05062370X.
14. B. Gärtner and A. Thomas. The niceness of unique sink orientations. Unpublished manuscript, 2016. URL: https://sites.google.com/site/antonisthomas/research/niceness.pdf.
15. D. Goldfarb and W.Y. Sit. Worst case behavior of the steepest edge simplex method. *Discrete Applied Mathematics*, 1(4):277–285, 1979.
16. T.D. Hansen, M. Paterson, and U. Zwick. Improved upper bounds for Random-Edge and Random-Jump on abstract cubes. In *Proc. of 25th SODA*, pages 874–881, 2014.
17. T.D. Hansen and U. Zwick. An improved version of the Random-Facet pivoting rule for the simplex algorithm. In *Proc. of 47th STOC*, pages 209–218, 2015.
18. R. G. Jeroslow. The simplex algorithm with the pivot rule of maximizing criterion improvement. *Discrete Mathematics*, 4(4):367–377, 1973.
19. G. Kalai. A simple way to tell a simple polytope from its graph. *Journal of combinatorial theory, Series A*, 49(2):381–383, 1988.
20. G. Kalai. A subexponential randomized simplex algorithm (extended abstract). In *Proc. of 24th STOC*, pages 475–482, 1992.
21. G. Kalai. Linear programming, the simplex algorithm and simple polytopes. *Mathematical Programming*, 79:217–233, 1997.
22. N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

23 L.G. Khachiyan. A polynomial time algorithm in linear programming. *Soviet Math. Dokl.*, 20:191–194, 1979.

24 V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175. Academic Press, New York, 1972.

25 D.A. Levin, Y. Peres, and E.L. Wilmer. *Markov chains and mixing times*. American Mathematical Soc., 2009.

26 Y. Mansour and S.P. Singh. On the complexity of policy iteration. In *Proc. of the 15th UAI*, pages 401–408, 1999.

27 J. Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures and Algorithms*, 5(4):591–608, 1994.

28 J. Matoušek and B. Gärtner. *Understanding and using linear programming*. Springer, 2007.

29 J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498–516, 1996.

30 J. Matoušek and T. Szabó. Random Edge can be exponential on abstract cubes. In *Proc. of 45th FOCS*, pages 92–100, 2004.

31 J. Matoušek and T. Szabó. RANDOM EDGE can be exponential on abstract cubes. *Advances in Mathematics*, 1(204):262–277, 2006.

32 A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.

33 I. Schurr and T. Szabó. Finding the sink takes some time: An almost quadratic lower bound for finding the sink of unique sink oriented cubes. *Discrete & Computational Geometry*, 31(4):627–642, 2004. doi:10.1007/s00454-003-0813-8.

34 T. Szabó and E. Welzl. Unique sink orientations of cubes. In *Proc. of 42th FOCS*, pages 547–555, 2001.

35 E. Welzl. Towards the peak – problem section. Unpublished manuscript, 2001. URL: http://www.ti.inf.ethz.ch/ew/workshops/01-lc/problems/node7.html.

36 K. Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Applied Mathematics*, 20(1):69–81, 1988. doi:10.1016/0166-218X(88)90042-X.

Approximating the Solution to Mixed Packing and Covering LPs in Parallel $\tilde{O}(\epsilon^{-3})$ Time

Michael W. Mahoney[1], Satish Rao[2], Di Wang[3], and Peng Zhang[4]

1 **International Computer Science Institute and Department of Statistics, UC Berkeley, Berkeley, USA**
mmahoney@stat.berkeley.edu
2 **Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, USA**
satishr@berkeley.edu
3 **Department of Electrical Engineering and Computer Sciences, UC Berkeley, Berkeley, USA**
wangd@eecs.berkeley.edu
4 **Department of Computer Science, Georgia Tech, Atlanta, USA**
pzhang60@gatech.edu

Abstract

We study the problem of approximately solving positive linear programs (LPs). This class of LPs models a wide range of fundamental problems in combinatorial optimization and operations research, such as many resource allocation problems, solving non-negative linear systems, computing tomography, single/multi commodity flows on graphs, etc. For the special cases of pure packing or pure covering LPs, recent result by Allen-Zhu and Orecchia [2] gives $\tilde{O}(\frac{1}{\epsilon^3})$-time parallel algorithm, which breaks the longstanding $\tilde{O}(\frac{1}{\epsilon^4})$ running time bound by the seminal work of Luby and Nisan [10].

We present new parallel algorithm with running time $\tilde{O}(\frac{1}{\epsilon^3})$ for the more general mixed packing and covering LPs, which improves upon the $\tilde{O}(\frac{1}{\epsilon^4})$-time algorithm of Young [18, 19]. Our work leverages the ideas from both the optimization oriented approach [2, 17], as well as the more combinatorial approach with phases [18, 19]. In addition, our algorithm, when directly applied to pure packing or pure covering LPs, gives a improved running time of $\tilde{O}(\frac{1}{\epsilon^2})$.

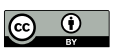
1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity

Keywords and phrases Mixed packing and covering, Linear program, Approximation algorithm, Parallel algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.52

1 Introduction

Mixed packing and covering linear programs (LPs) are LPs formulated with non-negative coefficients, non-negative constraints and non-negative variables. They model a wide range of fundamental problems in combinatorial optimization and operations research, thus have long drawn interest in theoretical computer science [10, 18, 2, 1]. Notable special cases include pure packing LPs and pure covering LPs, which apply to most resource allocation problems, and can be formulated respectively as $\max_{\boldsymbol{x}\geq 0}\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$ and $\min_{\boldsymbol{x}\geq 0}\{\boldsymbol{c}^T x : \boldsymbol{A}x \geq \boldsymbol{b}\}$ where $\boldsymbol{c}, \boldsymbol{A}, \boldsymbol{b} \geq 0$. More general than the pure packing and covering LPs, the mixed packing and covering LPs further capture problems requiring both packing and covering constraints, including solving non-negative linear systems, computing tomography, and single/multi commodity flows on graphs.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 52; pp. 52:1–52:14


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

Formally, the mixed packing and covering LP is the optimization problem

$$\min\{\lambda : \boldsymbol{P}\boldsymbol{x} \leq \lambda \boldsymbol{p}, \boldsymbol{C}\boldsymbol{x} \geq \boldsymbol{c}, \boldsymbol{x} \geq 0\} \tag{1}$$

where $\boldsymbol{P}, \boldsymbol{C}, \boldsymbol{p}, \boldsymbol{c}$ all have non-negative entries. A $(1+\epsilon)$-approximation is a feasible solution $\lambda, \boldsymbol{x}$ achieving $\lambda \leq (1+\epsilon)\lambda_{\text{OPT}}$.

Although one can use general LP solvers such as interior point method to solve packing and covering with convergence rate of $\log(\frac{1}{\epsilon})$ [8, 9, 5], such algorithms usually have very high per-iteration cost, as methods such as the computation of the Hessian and matrix inversion are involved. With the abundance of large-scale datasets, as well as the growing reliance on multiprocessors and cloud computing, low precision iterative solvers that can be highly parallelized are often more popular choices. Such parallel solvers compute approximate solutions usually in time with a poly-log dependence on the problem size, and nearly-linear total work, but they have $\text{poly}(\frac{1}{\epsilon})$ dependence on the approximation parameter ϵ.

Based on whether the running time depends on the width ρ, a parameter which typically depends on the dimension and the largest entry of $\boldsymbol{A}$, these algorithms can be divided into width-dependent solvers and width-independent solvers. Width-dependent solvers are usually pseudo-polynomial, as the running time depends at least linearly on $\rho\,\text{OPT}$, which itself can be large, while width-independent solvers are more efficient in the sense that they provide truly polynomial-time approximation solvers.

In this paper we focus on width-independent algorithms that produce $1+\epsilon$ approximations in $\text{poly}(\log n, \frac{1}{\epsilon})$ time and nearly-linear work. Time and work are standard notions from parallel algorithms that correspond to the longest chain of dependent operations and the total operations performed. In particular, time has a natural correspondence with iteration count, and these two measures have been used to study the performance of packing/covering LPs before [19]. Since the focus of this line of work is not on optimizing the log factors, we will follow the standard practice of using $\tilde{O}$ to hide poly-log factors (which are at most $\log^3 n$) in our discussion[1].

1.1 Previous work

Most of the works on mixed packing and covering LPs, as well as the works on the special cases of pure packing and pure covering LPs, take one of the two approaches. The first approach is based on turning the constrained LPs into convex and smooth objective functions with trivial or no constraints. Approximately solving the LP is then reduced to approximately optimizing the smoothed function (see [13]), and general-purpose optimization schemes are usually applied directly. This approach traditionally gives algorithms that work in more general settings, but are width-dependent in the case of packing and covering (e.g., [12, 3, 13, 16]). Recent breakthroughs in [2, 1] leverage the insight from optimization ([21]) and the special structure of packing and covering problems, and get the first width-independent algorithms using first order optimization methods. In particular, [2] gives a parallel algorithm that takes $\tilde{O}(\frac{1}{\epsilon^3})$ time and $\tilde{O}(\frac{N}{\epsilon^3})$ work. Here N is the size of the input, i.e., the total number of non-zeros in the constraint matrices $\boldsymbol{P}$ and $\boldsymbol{C}$. The result can be improved to run in $\tilde{O}(\frac{1}{\epsilon^2})$ time and $\tilde{O}(\frac{N}{\epsilon^2})$ work with simple modifications [17]. As for sequential algorithms, the remarkable result in [1] combines width-independence with Nesterov-like acceleration ([13, 14]), and gets a randomized algorithm with running time $\tilde{O}(\frac{N}{\epsilon})$. However, both results

[1] $\tilde{O}(f(n))$ is often used to denote $O(f(n)\log^{O(1)}(f(n)))$, we modify it to include an additional factor of $\log^{O(1)} n$.

in [2, 1] are limited to pure packing and pure covering problems, and prior to our work no methods are able to obtain a time of $\tilde{O}(\frac{1}{\epsilon^3})$ or better for mixed covering and packing LPs [19].

The other approach is based on the Lagrangian-relaxation framework, where, similar to the optimization approach, certain hard constraints are replaced by a soft scalar-value penalty function, and the partial solution is iteratively updated to satisfy the remaining constraints while minimizing the increase of the penalty function. The analysis of the Lagrangian-relaxation algorithms have more of a combinatorial flavor compared to the optimization schemes. Examples include the seminal work of Luby and Nisan [10], which gives the first width-independent algorithm for packing and covering, as well as subsequent works which improve the Luby and Nisan result in various ways [6, 18, 19, 7, 4, 11]. Among them, only [18, 19, 11] work with mixed packing and covering LPs, while others only work in the pure packing and pure covering setting. For algorithms working on mixed packing and covering, the results in [18, 19] have the best theoretical guarantee on parallel running time, with $\tilde{O}(\frac{1}{\epsilon^4})$ running time and $\tilde{O}(\frac{N}{\epsilon^2})$ total work. The result in [11] has worse bounds, but is stateless, which is a computational model on distributed algorithms with more restrictions on the processors (see [4]).

1.2 Our results

In this paper, we present a parallel algorithm that, given a mixed packing and covering LP with m variables and n total constraints, in $\tilde{O}(\frac{1}{\epsilon^3})$ iterations computes a $(1+\epsilon)$-approximate solution, or correctly reports the original mixed packing and covering LP is infeasible. The algorithm is deterministic and width-independent.

The bottleneck of each iteration is a matrix-vector multiplication, and can be implemented in $O(\log N)$ depth, in which case the running time of our algorithm is $\tilde{O}(\frac{1}{\epsilon^3})$. The total work of the algorithm we present in the paper is $\tilde{O}(\frac{N}{\epsilon^3})$. In particular, our result improves upon the current fastest parallel algorithm of mixed packing and covering LPs in [18, 19], where the running time is $\tilde{O}(\frac{1}{\epsilon^4})$. The work of the parallel algorithm in [18, 19] is $\tilde{O}(\frac{N}{\epsilon^2})$. We note that using a simple lazy update modification on the algorithm, which is the same technique used in [18, 19], we can reduce the work of our algorithm to $\tilde{O}(\frac{N}{\epsilon^2})$. Same as in [18, 19], this comes at the cost of requiring a centralized step in the parallel algorithm. Since the iteration count is the more interesting side of this line of work, we will not incorporate the lazy update in our algorithm for simplicity.

Furthermore, in the case of pure packing problem or pure covering problem, our algorithm allows a similar but simplified analysis, and will converge in $\tilde{O}(\frac{1}{\epsilon^2})$ iterations. This matches the running time achieved by the line of work [2, 17], but has the advantage of being deterministic and without centralized steps. We note that the technique we use in the analysis of the potential function in this work can be applied in a straightforward way to improve the result of [2] to have $\tilde{O}(\frac{1}{\epsilon^2})$ running time.

2 Technical overview

To compute $(1+\epsilon)$-approximation of a mixed packing and covering LP in the optimization form (1), via standard reduction and scaling (e.g., see [18]), it suffices to solve a $(1+O(\epsilon))$-feasibility problem as specified in (3) and (4).

At a high level, our work follows the Lagrangian-relaxation approach as in [18, 19]. In particular, we replace the hard packing and covering constraints with a scalar-valued potential function, which is continuous and smooth. The potential function measures how far away

the current solution is from satisfying all the captured constraints. As in [18], we use the soft-max $\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x})$ and soft-min $\operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$ in our potential function

$$\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}) = \ln(\sum_j \exp(\boldsymbol{P}\boldsymbol{x})_j), \text{ and } \operatorname{lmin}(\boldsymbol{C}\boldsymbol{x}) = -\ln(\sum_j \exp(-\boldsymbol{C}\boldsymbol{x})_j).$$

In particular, these functions give smooth approximation to $\max_j(\boldsymbol{P}\boldsymbol{x})_j$ and $\min_j(\boldsymbol{C}\boldsymbol{x})_j$:

$$\begin{aligned} \max_j(\boldsymbol{P}\boldsymbol{x})_j \leq \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}) \leq \max_j(\boldsymbol{P}\boldsymbol{x})_j + \ln n \\ \min_j(\boldsymbol{C}\boldsymbol{x})_j \geq \operatorname{lmin}(\boldsymbol{C}\boldsymbol{x}) \geq \min_j(\boldsymbol{C}\boldsymbol{x})_j - \ln n. \end{aligned} \tag{2}$$

The potential function we use will be

$$f(\boldsymbol{x}) = \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}) - \operatorname{lmin}(\boldsymbol{C}^{(t)}\boldsymbol{x}),$$

which is approximately the difference between $\max_j(\boldsymbol{P}\boldsymbol{x})_j$ and $\min_j(\boldsymbol{C}\boldsymbol{x})_j$.

The overall framework is to start with a $\boldsymbol{x}$ of very small values, and iteratively increase $\boldsymbol{x}$ while keeping $f(\boldsymbol{x})$ small. Roughly, when $\boldsymbol{x}$ is large enough so that

$$\max\{\max_j(\boldsymbol{P}\boldsymbol{x})_j, \min_j(\boldsymbol{C}\boldsymbol{x})_j\} \geq \frac{1}{\epsilon} f(\boldsymbol{x}),$$

we know that $\max_j(\boldsymbol{P}\boldsymbol{x})_j \leq (1 + O(\epsilon)) \min_j(\boldsymbol{C}\boldsymbol{x})_j$.

Same as in [18, 19], each iteration a subset of the variables are picked based on the gradients of the potential function, and are updated within a local smooth region so we can bound the change of the potential function. In particular, for each variable $\boldsymbol{x}_i$ we compute the packing gradient $\boldsymbol{a}_i = \nabla_i \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x})$, and the covering gradient $\boldsymbol{b}_i = \nabla_i \operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$. We want to update variables without increasing $f(\boldsymbol{x})$, and since $\nabla_i f(\boldsymbol{x}) = \boldsymbol{a}_i - \boldsymbol{b}_i$, a variable will be included in the update subset only when its covering gradient $\boldsymbol{b}_i$ is larger than its packing gradient $\boldsymbol{a}_i$.

However, unlike the parallel algorithm in [18, 19] that multiplicatively updates all variables in the subset with a uniform step size, we further incorporate the gradients $\boldsymbol{a}$ and $\boldsymbol{b}$ into step sizes of individual variables' updates. This discriminative multiplicative step size allows more aggressive updates on average, and is directly motivated by the line of works using gradient based optimization methods [2, 1, 17]. However, we move away from the optimization oriented view of these update steps in favor of more localized and adhoc analyses, which was developed to analyze direct adaptations of Young's algorithm for purely-packing SDPs [15], leading to bounds similar to the optimization based approaches [20].

Similar to Young's algorithm [18], the overall progress of the algorithm is captured by how large the constraints become. A sufficient termination condition for the algorithm is when a variable is increased by more than a certain amount, so we know $\max_j(\boldsymbol{P}\boldsymbol{x})_j$ and $\min_j(\boldsymbol{C}\boldsymbol{x})_j$ are large enough. To bound the number of iterations before any variable gets too large, we combine the notion of phases from [18] with the more refined analysis of gradient updates from more recent works [17]. This is owing to the clearer combinatorial structure of our interpretation of discriminative multiplicative steps.

On a high level, the analysis considers *phases*, where each phase is a consecutive sequence of iterations. The definition of a phase captures a local window, where the algorithm only makes limited global progress. The limited global progress ensures that inside a single phase the landscape doesn't change too much, which translate to certain monotonicity-like property on the gradients within the interactions captured by a single phase. In [18], each phase is defined to cover very little overall progress, which gives a very strong monotonicity-like

property. In particular, any variable being increased at the last iteration of a phase must have been increased in every iteration of the phase, which, coupled with a lower bound on each increase, gives a bound of the number of iterations in a phase.

In our analysis, we significantly expand the phases to capture larger global progress, leading to a smaller number of phases. The larger phases lead to a weaker monotonicity-like property on the gradients within a phase, and we develop a new approach to bound the number of iterations in our phase. We divide the iterations into two groups: some initial warm-up *bad* iterations followed by subsequent *good* iterations containing more interesting segments of the path to convergence (see Definition 7 for formal definitions). The bad iterations are ones where packing gradients are much smaller than covering gradients. These are the easy iterations to deal with, since a small ratio of the packing gradient over the covering gradient is a clear signal to increase a variable by a lot. An analysis identical to Young's algorithm [18] shows that they must occur near the very start of a phase, and there cannot be too many of them. In the subsequent good iterations, the packing gradients for all variables are relatively large comparing to their covering counterparts. These iterations capture the more difficult part of the path to convergence, since we only get weak signals as to which variables to update, and we can only increase them by small steps. We can show that as long as the problem is feasible, there cannot be too many good iterations in a phase. Intuitively, if the primal LP is feasible, the dual solution certifies that there must be some key variable(s) we increase during the phase to achieve the fixed global progress. Particularly, we know that there is at least one variable that on average has smaller packing gradients than covering gradients. Moreover, since in the good iterations, the packing gradients are all at least on the same scale as the covering gradients, an argument in the spirit of Markov's inequality then implies that the corresponding variable was increased by a large amount, which in turn leads to a bound on the number of iterations of a phase.

2.1 Remarks

We note that the phases in our result are virtual: we only need them in the analysis, but not in the actual execution of the algorithm. In particular, this modification to Young's algorithm [18] removes the dependency of updates on the phase of the overall algorithm as well as the current gradient. We believe this direct removal of phases also apply to other variants of Young's algorithm [19].

Furthermore, we believe that there is a more natural variant of the analysis that does not rely on phases, and treats all the iterations in a completely symmetric manner. Such an analysis is likely crucial for extending our results to the SDP setting, where the gradients exhibit much weaker monotonicity behaviors [20]. We are optimistic that it will lead to an $\tilde{O}(\frac{1}{\epsilon^2})$ bound for the mixed packing-covering case, which we believe is the more likely asymptotic behaviors of phase-less, gradient update variants of Young's algorithm [18, 19].

3 Parallel Algorithm for Mixed Packing and Covering LPs

3.1 Preliminaries

To compute $(1+\epsilon)$-approximation of a mixed packing and covering LP in the optimization form (1), via standard reduction and scaling (e.g., see [18]), it suffices to solve the following $(1+O(\epsilon))$-feasibility problem, that is, either find $\boldsymbol{x} \geq 0$ such that

$$0 < \max_j (\boldsymbol{P}\boldsymbol{x})_j \leq (1+\epsilon) \min_j (\boldsymbol{C}\boldsymbol{x})_j. \tag{3}$$

or conclude the following LP is infeasible

$$\begin{aligned} \boldsymbol{C}\boldsymbol{x} &\geq \mathbf{1} \\ \boldsymbol{P}\boldsymbol{x} &\leq (1-10\epsilon)\mathbf{1} \\ \boldsymbol{x} &\geq 0. \end{aligned} \tag{4}$$

We present our parallel $\tilde{O}(1/\epsilon^3)$ routine in Algorithm 1 for solving the $(1+O(\epsilon))$-feasibility problem above, that is, either find $\boldsymbol{x} \geq 0$ satisfying (3), or certify the infeasibility of (4). The input contains a packing constraint matrix $\boldsymbol{P} \in \mathbb{R}_{\geq 0}^{n_P \times m}$, a covering constraint matrix $\boldsymbol{C} \in \mathbb{R}_{\geq 0}^{n_C \times m}$, and an error parameter $\epsilon > 0$. That is, there are m variables, n_P packing constraints, and n_C covering constraints. We also use $n = n_P + n_C$ to denote the total number of constraints.

To certify that (4) is infeasible, we rely on the dual LP of (4).

▶ **Lemma 1.** *By duality,* (4) *is infeasible if there exists* $\boldsymbol{y}, \boldsymbol{z} \geq 0$ *s.t.*

$$(1-10\epsilon)\frac{\boldsymbol{C}^T\boldsymbol{z}}{\mathbf{1}^T\boldsymbol{z}} < \frac{\boldsymbol{P}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{y}}. \tag{5}$$

Proof. Eqn. (5) is a direct reformulation of the dual LP of (4). Since we only need the sufficient condition, the result is by weak duality. If there exists any $\boldsymbol{x} \geq 0$ satisfying (4), we have

$$\begin{aligned} (1-10\epsilon)\frac{\boldsymbol{x}^T\boldsymbol{C}^T\boldsymbol{z}}{\mathbf{1}^T\boldsymbol{z}} &\geq (1-10\epsilon)\frac{\mathbf{1}^T\boldsymbol{z}}{\mathbf{1}^T\boldsymbol{z}} = 1-10\epsilon, \\ \frac{\boldsymbol{x}^T\boldsymbol{P}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{y}} &\leq (1-10\epsilon)\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{y}} = 1-10\epsilon. \end{aligned}$$

Together they give $1 < 1$, contradiction. ◀

3.2 Algorithm

We start with small $\boldsymbol{x}_i^{(0)} = \frac{1}{m\|\boldsymbol{P}_{:,i}\|_\infty}, \forall\, i \in [m]$, and keep increasing $\boldsymbol{x}$ properly, until it reaches the termination condition in line 4, that is, $\max\{\max_j(\boldsymbol{P}\boldsymbol{x})_j, \min_j(\boldsymbol{C}\boldsymbol{x})_j\} \geq K = \frac{10\ln n}{\epsilon}$. The reason of the chosen K value is stated in Lemma 5.

In each iteration of the while-loop, we first delete all covering constraints which have already reached K. Since $\boldsymbol{x}$ never decreases, we know that once a row is deleted, we no longer need to look at it. Note the covering matrix cannot be empty, since we enter the iteration with $\min_j(\boldsymbol{C}\boldsymbol{x})_j < K$. We compute the vectors $\boldsymbol{y}, \boldsymbol{z}$, which are exponentials of the values of the packing and covering constraints respectively. We then compute $\boldsymbol{a}$ and $\boldsymbol{b}$, which can be considered as gradients of $\mathrm{lmax}(\boldsymbol{P}\boldsymbol{x})$ and $\mathrm{lmin}(\boldsymbol{C}\boldsymbol{x})$ respectively, and use them to guide our update on $\boldsymbol{x}$. In particular, we update $\boldsymbol{x}_i$ if $\boldsymbol{a}_i \leq (1-\epsilon/50)\boldsymbol{b}_i$ (i.e., $i \in B$). Furthermore, we update $\boldsymbol{x}_i$ multiplicatively by a factor depends on the ratio of $\frac{\boldsymbol{a}_i}{\boldsymbol{b}_i}$, as specified in Eqn. (6) and line 12. Note that the smallest update in our algorithm is by a factor of $(1+\Omega(\frac{\epsilon^2}{\ln n}))$, which is the same as the fixed update step size in [19], and in general our updates take larger steps.

Note that in our analysis, we equivalently view $\boldsymbol{z}$ as the full n_C-dimensional vector, where the coordinates corresponding to deleted constraints are filled by 0's. In particular, the matrix-vector product of the original $\boldsymbol{C}$ with the n_C-dimensional $\boldsymbol{z}$ will be the same as the product of the reduced covering matrix $\boldsymbol{C}^{(t)}$ and reduced $\boldsymbol{z}$.

Algorithm 1 Parallel algorithm for mixed packing and covering LPs

Input: $\boldsymbol{P}, \boldsymbol{C}, \epsilon$

Output: "infeasible" or $\boldsymbol{x} \geq 0$ s.t. $\max_j(\boldsymbol{P}\boldsymbol{x})_j \leq (1+\epsilon)\min_j(\boldsymbol{C}\boldsymbol{x})_j$

1: Let $K = \frac{10\ln n}{\epsilon}$, $\alpha = \frac{1}{K}$, where n is the number of constraints.
2: Initialize $\boldsymbol{x}_i^{(0)} = \frac{1}{m\|\boldsymbol{P}_{:,i}\|_\infty}, \forall\ i \in [m]$, where m is the number of variables.
3: Let $t = 0$.
4: **while** $\max_j(\boldsymbol{P}\boldsymbol{x})_j < K$ and $\min_j(\boldsymbol{C}\boldsymbol{x})_j < K$ **do**
5: Let $\boldsymbol{C}^{(t)}$ be $\boldsymbol{C}$ with rows j such that $\left(\boldsymbol{C}\boldsymbol{x}^{(t)}\right)_j \geq K$ deleted.
6: Let $\boldsymbol{y}^{(t)} = \exp\left(\boldsymbol{P}\boldsymbol{x}^{(t)}\right), \boldsymbol{z}^{(t)} = \exp\left(-\boldsymbol{C}^{(t)}\boldsymbol{x}^{(t)}\right)$.
7:
8: $\boldsymbol{a}^{(t)} = \frac{\boldsymbol{P}^T\boldsymbol{y}^{(t)}}{\boldsymbol{1}^T\boldsymbol{y}^{(t)}}, \boldsymbol{b}^{(t)} = \frac{(\boldsymbol{C}^{(t)})^T\boldsymbol{z}^{(t)}}{\boldsymbol{1}^T\boldsymbol{z}^{(t)}}$.
9: Define $B^{(t)} = \{i : \boldsymbol{a}_i^{(t)} \leq (1-\frac{\epsilon}{50})\boldsymbol{b}_i^{(t)}\}$.
10: If $B^{(t)} = \emptyset$, then return "infeasible".
11: Let

$$\Delta_i^{(t)} = \begin{cases} \frac{1}{2}(1-\frac{\boldsymbol{a}_i^{(t)}}{\boldsymbol{b}_i^{(t)}}) \in [\epsilon/100, \frac{1}{2}] & \text{if } i \in B^{(t)} \\ 0 & \text{if } i \notin B^{(t)} \end{cases} \tag{6}$$

12: $\boldsymbol{x}_i^{(t+1)} \leftarrow \boldsymbol{x}_i^{(t)}(1+\alpha\Delta_i^{(t)})$.
13: $t \leftarrow t+1$.
14: **end while**
15: **return** $\boldsymbol{x} = \frac{\boldsymbol{x}^{(t)}}{K}$.

3.3 Proof of Correctness

In this section we will show Algorithm 1 will terminate, and output the correct answer.

Lemma 2 shows that empty B certifies the infeasibility of the input instance (4), which proves the correctness if we end up in the case of line 10.

▶ **Lemma 2.** *If the problem instance* (4) *is feasible, then*

$$\forall\ \boldsymbol{x} \geq 0, B = \{i : \boldsymbol{a}_i \leq (1-\frac{\epsilon}{50})\boldsymbol{b}_i\} \neq \emptyset.$$

Proof. Assume by contradiction, $\exists\ \boldsymbol{x} \geq 0, \forall\ i \in [m], \boldsymbol{a}_i^{(t)} > (1-\frac{\epsilon}{50})\boldsymbol{b}_i^{(t)}$. By definition of $\boldsymbol{a}, \boldsymbol{b}$, it is equivalent to $\exists\ \boldsymbol{y}, \boldsymbol{z} \geq 0$ such that

$$(1-\frac{\epsilon}{50})\frac{\boldsymbol{C}^T\boldsymbol{z}}{\boldsymbol{1}^T\boldsymbol{z}} < \frac{\boldsymbol{P}^T\boldsymbol{y}}{\boldsymbol{1}^T\boldsymbol{y}}.$$

Then the result follows directly from Lemma 1. ◀

If Algorithm 1 does not terminate with line 10, it must increase at least one variable by at least a factor of $(1+\frac{\epsilon^2}{10\ln n})$ each iteration, so the algorithm must reach the termination condition of the while loop at some point, and we need to show the output $\boldsymbol{x}$ satisfies (3). Recall that we use the potential function,

$$f(\boldsymbol{x}^{(t)}) = \text{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t)}) - \text{lmin}(\boldsymbol{C}^{(t)}\boldsymbol{x}^{(t)}) = \ln(\boldsymbol{1}^T\boldsymbol{y}^{(t)}) + \ln(\boldsymbol{1}^T\boldsymbol{z}^{(t)}).$$

We first quantify the changes of lmax and lmin when we update the variables. This type of smoothness analysis is standard in analyzing algorithms that make updates using gradient information. Similar results are derived in other works on packing and covering (see [2, 18]). The particular analysis we develop can deal with larger gradient steps. In particular, the approach of our analysis allows updates that may move the gradients of some variables out of their respective coordinate-wise smooth regions, as long as we can still bound the combined impact on the potential function from updates of all variables. This approach can extend straightforwardly to show larger updates also work in [2], and improve their pure packing algorithm to run in $\tilde{O}(\frac{1}{\epsilon^2})$ iterations. Since the proof is straightforward but technically tedious, we omit it.

▶ **Lemma 3.** *At each iteration t,*

$$\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t+1)}) \le \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}^{(t)}) + \alpha\langle \boldsymbol{a}^{(t)}, (1+\Delta^{(t)}) \circ \Delta^{(t)} \circ \boldsymbol{x}^{(t)}\rangle$$

and

$$\operatorname{lmin}(\boldsymbol{C}^{(t+1)}\boldsymbol{x}^{(t+1)}) \ge \operatorname{lmin}(\boldsymbol{C}^{(t)}\boldsymbol{x}^{(t)}) + \alpha\langle \boldsymbol{b}^{(t)}, (1-\Delta^{(t)}) \circ \Delta^{(t)} \circ \boldsymbol{x}^{(t)}\rangle,$$

where $\Delta \circ \boldsymbol{x}$ is the entry-wise product vector, i.e., $(\Delta \circ \boldsymbol{x})_i = \Delta_i \boldsymbol{x}_i$.

With the above bounds on the changes of the two components $\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x})$ and $\operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$, we can show how our updates move the potential function $f(\boldsymbol{x})$.

▶ **Lemma 4.** *Given $\max_j(\boldsymbol{P}\boldsymbol{x}^{(t)})_j < \frac{10\ln n}{\epsilon}$ and $\min_j(\boldsymbol{C}\boldsymbol{x}^{(t)})_j < \frac{10\ln n}{\epsilon}$, we always have $f(\boldsymbol{x}^{(t)}) \le 2\ln n$ during the execution of Algorithm 1.*

Proof. Initially, $\boldsymbol{x}_i^{(0)} = \frac{1}{m\|\boldsymbol{P}_{:,i}\|_\infty}$, we have $\boldsymbol{P}\boldsymbol{x}^{(0)} \le \boldsymbol{1}$ and $\boldsymbol{C}\boldsymbol{x} \ge 0$, thus $f(\boldsymbol{x}^{(0)}) \le 2\ln n$. To show $f(\boldsymbol{x}) \le 2\ln n$ for all iterations t before terminate, it suffices to show that $f(\boldsymbol{x})$ is non-increasing during the process. From Lemma 3,

$$\begin{aligned} f(\boldsymbol{x}^{(t+1)}) - f(\boldsymbol{x}^{(t)}) &\le \alpha\langle \boldsymbol{a}, (1+\Delta)\circ\Delta\circ\boldsymbol{x}^{(t)}\rangle - \alpha\langle \boldsymbol{b}, (1-\Delta)\circ\Delta\circ\boldsymbol{x}^{(t)}\rangle \\ &= \sum_i \alpha\Delta_i\boldsymbol{x}_i(\boldsymbol{a}_i(1+\Delta_i) - \boldsymbol{b}_i(1-\Delta_i)). \end{aligned}$$

For each $i \in [m]$, by our update rule (6), either Δ_i, or $\Delta_i = \frac{1}{2}(1 - \frac{\boldsymbol{a}_i}{\boldsymbol{b}_i})$, in which case

$$\boldsymbol{a}_i(1+\Delta_i) - \boldsymbol{b}_i(1-\Delta_i) = \frac{3\boldsymbol{a}_i\boldsymbol{b}_i - \boldsymbol{a}_i^2}{2\boldsymbol{b}_i} - \frac{\boldsymbol{a}_i + \boldsymbol{b}_i}{2} = \frac{2\boldsymbol{a}_i\boldsymbol{b}_i - \boldsymbol{a}_i^2 - \boldsymbol{b}_i^2}{2\boldsymbol{b}_i} \le 0,$$

so all the summands are non-positive, thus $f(\boldsymbol{x})$ is non-increasing. ◀

The above lemma guarantees that the difference between $\operatorname{lmax}(\boldsymbol{P}\boldsymbol{x})$ and $\operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$ is bounded by $2\ln n$, which by Eqn. (2) suggests $\max_j(\boldsymbol{P}\boldsymbol{x})_j \le \min_j(\boldsymbol{C}\boldsymbol{x})_j + O(\ln n)$. Then when the two terms are large at termination, we are approximately feasible as the difference is a factor of ϵ smaller.

▶ **Lemma 5.** *If Algorithm 1 terminates with line 15, then it returns an $\boldsymbol{x} \ge 0$ with $0 < \max_j(\boldsymbol{P}\boldsymbol{x})_j \le (1+\epsilon)\min_j(\boldsymbol{C}\boldsymbol{x})_j$.*

Proof. Suppose the algorithm terminates at iteration T, that is, $\max_j(\boldsymbol{P}\boldsymbol{x}^{(T)})_j \ge \frac{10\ln n}{\epsilon}$ or $\min_j(\boldsymbol{C}\boldsymbol{x}^{(T)})_j \ge \frac{10\ln n}{\epsilon}$. Consider iteration $T-1$, the covering matrix is not empty (otherwise, the algorithm terminates before iteration T). Since $\boldsymbol{x}^{(T)} = \boldsymbol{x}^{(T-1)} \circ (1 + \alpha\Delta^{(T-1)}) \le (1 + \frac{5\epsilon}{\ln n})\boldsymbol{x}^{(T-1)}$, we have $\max_j(\boldsymbol{P}\boldsymbol{x}^{(T-1)})_j \ge \frac{5\ln n}{\epsilon}$ or $\min_j(\boldsymbol{C}\boldsymbol{x}^{(T-1)})_j \ge \frac{5\ln n}{\epsilon}$.

By Lemma 4,

$$\begin{aligned}\max_j(\boldsymbol{P}\boldsymbol{x}^{(T-1)})_j &\leq \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}^{(T-1)}) \leq \operatorname{lmin}(\boldsymbol{C}^{(T-1)}\boldsymbol{x}^{(T-1)}) + 2\ln n \\ &\leq \min_j(\boldsymbol{C}\boldsymbol{x}^{(T-1)})_j + 2\ln n.\end{aligned}$$

Since $2\ln n \leq \epsilon \cdot \frac{5\ln n}{\epsilon}$, we have

$$\max_j(\boldsymbol{P}\boldsymbol{x}^{(T-1)})_j \leq (1+\epsilon)\min_j(\boldsymbol{C}\boldsymbol{x}^{(T-1)})_j.$$

This also gives $\max_j(\boldsymbol{P}\boldsymbol{x}^{(T)})_j \leq (1+\epsilon)\min_j(\boldsymbol{C}\boldsymbol{x}^{(T)})_j$, since $\boldsymbol{x}^T$ is within in multiplicative factor $1+\frac{\epsilon}{10\ln n}$ of $\boldsymbol{x}^{T-1}$. Since we start with $\boldsymbol{x} > 0$, and only increase $\boldsymbol{x}$, we also have $\max_j(\boldsymbol{P}\boldsymbol{x})_j > 0$. So the $\boldsymbol{x}$ we return at the end satisfies (3). ◀

3.4 Analysis of Convergence

So far we have proved that Algorithm 1 will terminate, and will either output $\boldsymbol{x}$ satisfying (3) at the end, or terminate earlier and correctly certify (4) is infeasible. In this section we show that if (4) is feasible, Algorithm 1 must finish with the first case in $\tilde{O}(\frac{1}{\epsilon^3})$ iterations, so if the algorithm takes more than $\frac{1000\ln n\ln(\frac{m}{\epsilon})}{\epsilon^3}$ iterations to complete, we can terminate it, and correctly output that (4) is infeasible.

We adapt the concept *phase* from Young's algorithm. Note phase is only used in our analysis, and our algorithm does not contain phase. Formally, phase s contains the iterations t such that

$$\frac{n_P}{n_C}\cdot 2^s \leq \frac{\mathbf{1}^T\boldsymbol{y}^{(t)}}{\mathbf{1}^T\boldsymbol{z}^{(t)}} < \frac{n_P}{n_C}\cdot 2^{s+1}$$

where n_P is the number of packing constraints and n_C is the number of covering constraints.

Since we only increase $\boldsymbol{x}$, $\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{z}}$ is monotonically increasing, so each phase covers a consecutive sequence of iterations. Furthermore, as $\ln(\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{z}}) = \operatorname{lmax}(\boldsymbol{P}\boldsymbol{x}) + \operatorname{lmin}(\boldsymbol{C}\boldsymbol{x})$ measures global progress towards termination, each phase captures a fixed amount of progress. From our definition of phases, and the termination condition, we have the following lemma.

▶ **Lemma 6.** *The total number of phases in Algorithm 1 is* $O(\frac{\log n}{\epsilon})$.

Proof. Since $\boldsymbol{x}$ is monotonically increasing, $\boldsymbol{y} = \exp(\boldsymbol{P}\boldsymbol{x})$ and $\boldsymbol{z} = \exp(-\boldsymbol{C}\boldsymbol{x})$ are monotonically increasing and decreasing respectively, which implies that the quantity $\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{z}}$ is monotonically increasing. Initially $\boldsymbol{P}\boldsymbol{x}^{(0)} \geq 0$, $\boldsymbol{C}\boldsymbol{x}^{(0)} \geq 0$, we know $\frac{\mathbf{1}^T\boldsymbol{y}^{(0)}}{\mathbf{1}^T\boldsymbol{z}^{(0)}} \geq \frac{n_P}{n_C}$. By the termination condition in Algorithm 1, the ratio never goes beyond $n_P\exp(\frac{10\log n}{\epsilon})$. Therefore, the total number of phases is $O(\frac{\log n}{\epsilon})$. ◀

We now bound the number of iterations in a single phase. The iterations of a phase are divided into two groups, the bad iterations and the good iterations, formally defined as follows.

▶ **Definition 7.** If in an iteration t, we have for all i

$$\frac{\boldsymbol{a}_i^{(t)}}{\boldsymbol{b}_i^{(t)}} > \frac{1}{3}, \tag{7}$$

then we call it a *good* iteration. Otherwise we call it a *bad* iteration.

Note that it is also possible for a phase to contain only bad iterations or only good iterations. We bound the total number of iterations in the two groups separately.

As discussed earlier, the bad iterations capture the initial warm-up iterations of a phase, where in any bad iteration, we can identify some variable $\boldsymbol{x}_i$ with a strong signal (i.e., $\frac{\boldsymbol{a}_i}{\boldsymbol{b}_i} \leq \frac{1}{3}$), so we can increase the variable by a lot. This restricts the warm-up sequence from getting too long, and we formalize the intuition in the following lemma.

▶ **Lemma 8.** *In a single phase, the number of bad iterations is at most* $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$.

Proof. We will prove the result by showing that there cannot be any bad iteration after the initial $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ iterations of a phase. By contradiction, if for any variable i, after $\Omega(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$ iterations of a phase, we have at iteration t such that for some i,

$$\frac{\boldsymbol{a}_i^{(t)}}{\boldsymbol{b}_i^{(t)}} = \frac{(\boldsymbol{P}^T\boldsymbol{y}^{(t)})_i}{\boldsymbol{1}^T\boldsymbol{y}^{(t)}} \frac{\boldsymbol{1}^T\boldsymbol{z}^{(t)}}{(\boldsymbol{C}^T\boldsymbol{z}^{(t)})_i} \leq \frac{1}{3},$$

then this ratio is at most $\frac{2}{3}$ in all previous iterations of this phase, since $\frac{(\boldsymbol{P}^T\boldsymbol{y})_i}{(\boldsymbol{C}^T\boldsymbol{z})_i}$ is monotonically increasing, and $2^s \leq \frac{\boldsymbol{1}^T\boldsymbol{y}}{\boldsymbol{1}^T\boldsymbol{z}} < 2^{s+1}$ in this phase. Equivalently, this is saying $\boldsymbol{a}_i \leq \frac{2}{3}\boldsymbol{b}_i$, so $i \in B$ in all previous $\Omega(\ln n \ln(\frac{m}{\epsilon})/\epsilon)$ iterations of the phase, and $\Delta_i \geq \frac{1}{6}$ in all those iterations.

Each iteration the multiplicative update on $\boldsymbol{x}_i$ is $(1+\alpha\Delta_i)$, which is $(1+\Theta(\frac{\epsilon}{10\ln n}))$ since $\Delta_i \geq \frac{1}{6}$. As $\boldsymbol{x}_i$ starts with $\frac{1}{m\|P_{:,i}\|_\infty}$, after $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ updates, we have $\boldsymbol{x}_i \gg \frac{10\ln n}{\epsilon\|P_{:,i}\|_\infty}$, which gives $\max_j(\boldsymbol{P}\boldsymbol{x})_j \gg \frac{10\ln n}{\epsilon}$, so the algorithm must have terminated. ◀

The above lemma guarantees that all iterations after the first $100 \ln n \ln(\frac{m}{\epsilon})/\epsilon$ must be good iterations, so we proceed to bound the number of these good iterations in a single phase. Without loss of generality, we index these good iterations in a phase as $1, \ldots, T$ by shifting t.

We first identify one variable that must be updated extensively in these iterations.

▶ **Lemma 9.** *Suppose the instance* (4) *is feasible, then There exists* $i \in [m]$ *such that*

$$\sum_{t=1}^{T} \boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)} \geq 10\epsilon \sum_{t=1}^{T} \boldsymbol{b}_i^{(t)}. \tag{8}$$

Proof. Define $\overline{\boldsymbol{y}}$ and $\overline{\boldsymbol{z}}$ to be the sum of the normalized gradients of iterations $1, \ldots, T$, that is,

$$\overline{\boldsymbol{y}} = \sum_{t=1}^{T} \frac{\boldsymbol{y}^{(t)}}{\boldsymbol{1}^T\boldsymbol{y}^{(t)}}, \quad \overline{\boldsymbol{z}} = \sum_{t=1}^{T} \frac{\boldsymbol{z}^{(t)}}{\boldsymbol{1}^T\boldsymbol{z}^{(t)}}.$$

Note $\boldsymbol{1}^T\overline{\boldsymbol{y}} = \boldsymbol{1}^T\overline{\boldsymbol{z}} = T$. Recall $\boldsymbol{a}_i^{(t)}$ and $\boldsymbol{b}_i^{(t)}$ are respectively $\frac{(\boldsymbol{P}^T\boldsymbol{y}^{(t)})_i}{\boldsymbol{1}^T\boldsymbol{y}^{(t)}}$ and $\frac{(\boldsymbol{C}^T\boldsymbol{z}^{(t)})_i}{\boldsymbol{1}^T\boldsymbol{z}^{(t)}}$, then

$$\sum_{t=1}^{T} \boldsymbol{a}_i^{(t)} = \frac{T(\boldsymbol{P}^T\overline{\boldsymbol{y}})_i}{\boldsymbol{1}^T\overline{\boldsymbol{y}}}, \quad \sum_{t=1}^{T} \boldsymbol{b}_i^{(t)} = \frac{T(\boldsymbol{C}^T\overline{\boldsymbol{z}})_i}{\boldsymbol{1}^T\overline{\boldsymbol{z}}}.$$

Assume by contradiction, $\forall\, i \in [m]$, $\sum_{t=1}^{T} \boldsymbol{a}_i^{(t)} > (1-10\epsilon)\sum_{t=1}^{T} \boldsymbol{b}_i^{(t)}$, that is,

$$\frac{\boldsymbol{P}^T\overline{\boldsymbol{y}}}{\boldsymbol{1}^T\overline{\boldsymbol{y}}} > (1-10\epsilon)\frac{\boldsymbol{C}^T\overline{\boldsymbol{z}}}{\boldsymbol{1}^T\overline{\boldsymbol{z}}}.$$

By Lemma 1, $\overline{\boldsymbol{y}}, \overline{\boldsymbol{z}}$ certify infeasibility of the instance (4), which contradicts the assumption. ◀

The above claim gives us a variable that on average has smaller packing gradients than covering gradients in this iteration. Together with the property we have on the good iterations (7), we can bound the number of good iterations.

▶ **Lemma 10.** *In a single phase, the number of good iterations is at most* $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon^2)$.

Proof. Let $\boldsymbol{x}_i$ be a variable satisfying Eqn. (8). We want to turn Eqn. (8) into some lower bound on the total multiplicative update on $\boldsymbol{x}_i$ through these iterations. Intuitively, a bad case is that in some iteration t, $\boldsymbol{a}_i^{(t)}, \boldsymbol{b}_i^{(t)}$ are much larger than the values in other iterations, since they can dominate the terms from other iterations in Eqn. (8), but not much to the total update of $\boldsymbol{x}_i$, since their ratio is what matters to the update. However, since we are inside one single phase, and only looking at good iterations, we can show the bad scenario will not show up.

Formally, let $l = \boldsymbol{a}_i^{(1)}$ and $u = \boldsymbol{b}_i^{(1)}$. Since $(\boldsymbol{P}^T\boldsymbol{y})_i$ monotonically increases, and $\mathbf{1}^T\boldsymbol{y}$ will increase but not by more than a factor of 2 in a phase, we have

$$\boldsymbol{a}_i^{(t)} = \frac{(\boldsymbol{P}^T\boldsymbol{y}^{(t)})_i}{\mathbf{1}^T\boldsymbol{y}^{(t)}} \geq l/2 \qquad \forall t = 1, \ldots, T. \tag{9}$$

Similarly, we have

$$\boldsymbol{b}_i^{(t)} = \frac{(\boldsymbol{C}^T\boldsymbol{z}^{(t)})_i}{\mathbf{1}^T\boldsymbol{z}^{(t)}} \leq 2u \qquad \forall t = 1, \ldots, T. \tag{10}$$

Furthermore, since we are looking at the good iterations, we have

$$l \geq \frac{1}{3}u. \tag{11}$$

The inequalities above allow us to turn the difference-based guarantee from Eqn. (8) into lower bounds on ratios we need.

By the update (6), we have

$$\Delta_i^{(t)} \geq \frac{(1-\frac{\epsilon}{50})\boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)}}{2\boldsymbol{b}_i^{(t)}}.$$

So we can lower bound the total update on $\boldsymbol{x}_i$ as follows

$$\begin{aligned}
\boldsymbol{x}_i^{(T)} \geq& \boldsymbol{x}_i^{(1)} \exp\left(\frac{\alpha\sum_t \Delta_i^{(t)}}{2}\right) \\
=& \boldsymbol{x}_i^{(1)} \exp\left(\frac{\alpha}{4}\sum_t \frac{(1-\frac{\epsilon}{50})\boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)}}{\boldsymbol{b}_i^{(t)}}\right) \\
\geq& \boldsymbol{x}_i^{(1)} \exp\left(\frac{\alpha}{4}\sum_t \left(\frac{\boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)}}{2u} - \frac{\epsilon}{50}\right)\right)
\end{aligned}$$

where we used (10) in the last line.

From Eqn. (8), we have

$$\begin{aligned}
\sum_t \boldsymbol{b}_i^{(t)} - \boldsymbol{a}_i^{(t)} \geq& 10\epsilon \sum_i \boldsymbol{b}_i^{(t)} \\
\geq& \frac{10\epsilon}{1-10\epsilon}\sum_t \boldsymbol{a}_i^{(t)} \\
\geq& \frac{\epsilon u T}{1-10\epsilon} \geq \epsilon u T.
\end{aligned}$$

The first two lines both follow from Eqn. (8), the next line follows from $\boldsymbol{a}_i^{(t)} \geq l/2 \geq u/6$.

Thus

$$\boldsymbol{x}_i^{(T)} \geq \boldsymbol{x}_i^{(1)} \exp\left(\frac{\epsilon\alpha T}{8} - \frac{\epsilon\alpha T}{200}\right) \geq \frac{1}{m\left\|\boldsymbol{P}_{:,i}\right\|_\infty} \exp\left(\frac{\epsilon\alpha T}{10}\right).$$

If $T \geq \frac{100 \ln n \ln \frac{m}{\epsilon}}{\epsilon^2} \geq \frac{100 \ln \frac{m}{\epsilon}}{\epsilon\alpha}$, we have $\boldsymbol{x}_i^{(T)} \gg \frac{10 \ln n}{\epsilon\left\|\boldsymbol{P}_{:,i}\right\|_\infty}$. So the algorithm must have terminated since $\max_j(\boldsymbol{P}\boldsymbol{x})_j \gg \frac{10 \ln n}{\epsilon}$. ◀

Lemma 8 and Lemma 10 bound the total number of iterations in a phase by $\tilde{O}(\frac{1}{\epsilon^2})$, together with the bound on the number of phases, which is $\tilde{O}(\frac{1}{\epsilon})$, we guarantee the total number of iterations in Algorithm 1 is $\tilde{O}(\frac{1}{\epsilon^3})$ if the LP in (4) is feasible.

▶ **Theorem 11.** *Algorithm 1 solves the $(1+\epsilon)$-feasibility problem correctly. It runs in parallel time $\tilde{O}(1/\epsilon^3)$ with the total work $\tilde{O}(N/\epsilon^3)$, where N is the number of non-zero entries in the constraint matrix.*

Proof. The correctness and convergence follows from the lemmas in the prior sections. We only need to look at the running time and total work.

At each iteration, we compute all updated values in $O(\log N)$ parallel time. Since the total number of iterations is $\tilde{O}(\frac{1}{\epsilon^3})$, the algorithm terminates in parallel time $\tilde{O}(\frac{1}{\epsilon^3})$.

To see the total work, consider the following implementation. For each $i \in [m]$, we maintain $\boldsymbol{P}_{ji}\boldsymbol{x}_i$ if $\boldsymbol{P}_{ij} \neq 0$; similarly we maintain $\boldsymbol{C}_{ji}\boldsymbol{x}_i$ if $\boldsymbol{C}_{ij} \neq 0$. Besides, we maintain the values of $\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{P}^T\boldsymbol{y}, \boldsymbol{C}^T\boldsymbol{z}, \mathbf{1}^T\boldsymbol{y}$ and $\mathbf{1}^T\boldsymbol{z}$. When we update $\boldsymbol{x}_i$, we update these values accordingly, with work proportional to the number of non-zero entries in the ith column of the constraint matrix. For each fixed variable $\boldsymbol{x}_i$, the total time of updates is at most $\tilde{O}(\frac{1}{\epsilon^2})$. Thus, the work on this part is $\tilde{O}(\frac{N}{\epsilon^2})$. Additionally, we need to compute the $\boldsymbol{a}_i, \boldsymbol{b}_i$ for all variables at the beginning of each iteration to determine which variables to update, this takes $\tilde{O}(N)$ work each iteration, so the total work is $\tilde{O}(\frac{N}{\epsilon^3})$. ◀

▶ Remark. We see the majority of the work is actually on computing the gradients for the variables we may not update. We point out that we can implement the same lazy update as in [19], which on a high level is just that if a variable has a large $\frac{\boldsymbol{a}_i}{\boldsymbol{b}_i}$ in an iteration, and is not updated, we do not recompute its gradients, until $\frac{\mathbf{1}^T\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{z}}$ grows by more than a factor of $1+\epsilon$. This can reduce the work to $\tilde{O}(\frac{N}{\epsilon^2})$, but requires a centralized step to control the phases. We omit the details as it is a straightforward adaptation.

4 Pure Packing and Pure Covering LPs

We point out that in the case of pure packing or pure covering LPs, Algorithm 1 converges in $\tilde{O}(\frac{1}{\epsilon^2})$ iterations. This improves upon the result of [17], since our algorithm is deterministic, and does not need centralized steps.

The analysis follows the same approach as the mixed case, but the result of Lemma 10 can be improved to bound the total number of good iterations over all phases by $O(\ln n \ln(\frac{m}{\epsilon})/\epsilon^2)$, since for pure packing (or pure covering) LPs, the $\boldsymbol{a}_i$'s (or the $\boldsymbol{b}_i$'s) are constants through the algorithm. We omit the details as the proof is a straightforward adaptation.

Acknowledgments. We thank Richard Peng for helpful discussions on results and writing of the paper. DW was supported by ARO Grant W911NF-12-1-0541, SR was funded by NSF Grant CCF-1118083, and MM acknowledges the support of the NSF, AFOSR, and DARPA.

References

1. Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC'15, pages 229–236, 2015. Newer version available at `http://arxiv.org/abs/1411.1124`.
2. Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 1439–1456, 2015.
3. Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012. URL: `http://www.theoryofcomputing.org/articles/v008a006`, `doi:10.4086/toc.2012.v008a006`.
4. Baruch Awerbuch and Rohit Khandekar. Stateless distributed gradient descent for positive linear programs. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 691–700, 2008.
5. S. Boyd and L. Vandenberghe. *Convex Optimization*. Camebridge University Press, 2004.
6. Lisa Fleischer. A fast approximation scheme for fractional covering problems with variable upper bounds. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004, New Orleans, Louisiana, USA, January 11-14, 2004*, pages 1001–1010, 2004.
7. Christos Koufogiannakis and Neal E. Young. A nearly linear-time PTAS for explicit fractional packing and covering linear programs. *Algorithmica*, 70(4):648–674, 2014. `doi:10.1007/s00453-013-9771-6`.
8. Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in õ(sqrt(rank)) iterations and faster algorithms for maximum flow. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 424–433, 2014.
9. Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 230–249, 2015.
10. Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 448–457, 1993.
11. Faraz Makari Manshadi, Baruch Awerbuch, Rainer Gemula, Rohit Khandekar, Julián Mestre, and Mauro Sozio. A distributed algorithm for large-scale generalized matching. *PVLDB*, 6(9):613–624, 2013. Available at http://www.vldb.org/pvldb/vol6/p613-makarimanshadi.pdf.
12. Arkadi Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004. `doi:10.1137/S1052623403425629`.
13. Yurii Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005. `doi:10.1007/s10107-004-0552-5`.
14. Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012. `doi:10.1137/100802001`.
15. Richard Peng and Kanat Tangwongsan. Faster and simpler width-independent parallel algorithms for positive semidefinite programming. In *Proceedinbgs of the 24th ACM symposium on Parallelism in algorithms and architectures*, SPAA '12, pages 101–108, 2012. Available at http://arxiv.org/abs/1201.5135.

16 James Renegar. Efficient first-order methods for linear programming and semidefinite programming. *CoRR*, abs/1409.5832, 2014. URL: http://arxiv.org/abs/1409.5832.

17 Di Wang, Michael W. Mahoney, Nishanth Mohan, and Satish Rao. Faster parallel solver for positive linear programs via dynamically-bucketed selective coordinate descent. *CoRR*, abs/1511.06468, 2015. URL: http://arxiv.org/abs/1511.06468.

18 Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 538–546, 2001.

19 Neal E. Young. Nearly linear-time approximation schemes for mixed packing/covering and facility-location linear programs. *CoRR*, abs/1407.3015, 2014. URL: http://arxiv.org/abs/1407.3015.

20 Zeyuan Allen Zhu, Yin Tat Lee, and Lorenzo Orecchia. Using optimization to obtain a width-independent, parallel, simpler, and faster positive SDP solver. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1824–1831, 2016. Available at http://arxiv.org/abs/1507.02259.

21 Zeyuan Allen Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *CoRR*, abs/1407.1537, 2014. URL: http://arxiv.org/abs/1407.1537.

Optimization Algorithms for Faster Computational Geometry*

Zeyuan Allen-Zhu[1], Zhenyu Liao[2], and Yang Yuan[3]

1 **Princeton University, Princeton, USA**
`zeyuan@csail.mit.edu`
2 **Boston University, Boston, USA**
`zhenyul@bu.edu`
3 **Cornell University, Ithaka, USA**
`yangyuan@cs.cornell.edu`

Abstract

We study two fundamental problems in computational geometry: finding the maximum inscribed ball (MaxIB) inside a bounded polyhedron defined by m hyperplanes, and the minimum enclosing ball (MinEB) of a set of n points, both in d-dimensional space. We improve the running time of iterative algorithms on

MaxIB from $\widetilde{O}(md\alpha^3/\varepsilon^3)$ to $\widetilde{O}(md + m\sqrt{d}\alpha/\varepsilon)$, a speed-up up to $\widetilde{O}(\sqrt{d}\alpha^2/\varepsilon^2)$, and[1]
MinEB from $\widetilde{O}(nd/\sqrt{\varepsilon})$ to $\widetilde{O}(nd + n\sqrt{d}/\sqrt{\varepsilon})$, a speed-up up to $\widetilde{O}(\sqrt{d})$.

Our improvements are based on a novel saddle-point optimization framework. We propose a new algorithm `L1L2SPSolver` for solving a class of regularized saddle-point problems, and apply a randomized Hadamard space rotation which is a technique borrowed from compressive sensing. Interestingly, the motivation of using Hadamard rotation solely comes from our optimization view but not the original geometry problem: indeed, it is not immediately clear why MaxIB or MinEB, as a geometric problem, should be easier to solve if we rotate the space by a unitary matrix. We hope that our optimization perspective sheds lights on solving other geometric problems as well.

1998 ACM Subject Classification F.2 Analysis of Algorithms and Problem Complexity – geometrical problems and computations

Keywords and phrases maximum inscribed balls, minimum enclosing balls, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.53

1 Introduction

The goal of this paper is to bridge the fields of optimization and computational geometry using a simple unified saddle-point framework. As two immediate products of this new connection, we obtain faster iterative algorithms to approximately solve two fundamental problems in computational geometry: the maximum inscribed ball problem (MaxIB) and the

* This is an abstract of our full paper at `http://arxiv.org/abs/1412.1001` [3]. The first version of this paper appeared in December 2014 but contains only the smooth convex optimization based algorithms. The second version of this paper appeared in December 2015 and already contains all the technical details of this present paper.

[1] $\alpha \geq 1$ is the aspect ratio of the polyhedron. Throughout this paper we use the $\widetilde{O}$ notation to hide logarithm factors such as $\log m, \log d, \log \alpha$, and $\log(1/\varepsilon)$.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 53; pp. 53:1–53:6


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

minimum enclosing ball problem (MinEB). Our methods are composed of simple updating rules on vectors and therefore do not require geometric operations that are found in classical algorithms. This is another example of surprisingly good results obtained using optimization insights following the current trend of theoretical computer science.

In the rest of this introduction, we describe the definitions of the MaxIB and MinEB problems and review prior work. In the next three sections, we describe our saddle-point formulation and algorithms for MaxIB and MinEB.

Maximum Inscribed Ball (MaxIB). In the MaxIB problem, we are given a polyhedron P in $\mathbb{R}^d$ defined by m halfspaces $\{H_1, \ldots, H_m\}$. Each halfspace H_j is characterized by a linear constraint $\langle A_j, x\rangle + b_j \geq 0$. As in prior work [20], we assume that P is bounded (so $m \geq d$) and a common point is known to be contained in P – without loss of generality, let it be the origin O. Let $\alpha \geq 1$ be an upper bound on the aspect ratio of P, i.e., the ratio between the radii of the minimum enclosing ball and the maximum inscribed ball of P, and $\varepsilon > 0$ be a desired error bound.

The goal of MaxIB is to find a point $x \in P$ such that its minimum distance to all the bounding hyperplanes H_j is at least $(1-\varepsilon)r_{\mathsf{opt}}$, where r_{opt} is the radius of a maximum inscribed ball of P.

Besides the applications in computational geometry, MaxIB has also been used in the column generation method [13] and the sphere method [14] for linear programming, and the central cutting-plane method for convex programming [9].

When the dimension is a constant, the ε-kernel technique (see the survey [1]) yields a linear-time approximation algorithm for MaxIB based on core-set construction. However, its running time is proportional to $\varepsilon^{-\Omega(d)}$. In high dimensions, finding the maximum inscribed ball remains a challenging problem in theoretical computer science and operations research. One can reduce this problem to a linear program [9] and rely on existing LP solvers, however, the so-obtained algorithm can be too slow for practical purposes (although still in polynomial time).

In an influential paper, Xie, Snoeyink, and Xu [20] obtained an approximation algorithm for MaxIB with running time $O(md\alpha^3/\varepsilon^3 + md\alpha\log\alpha) = \widetilde{O}(md\alpha^3/\varepsilon^3)$. Their algorithm is based on a number of interesting geometric observations, as well as a dual transformation to reduce the MaxIB problem to a sequence of minimum enclosing ball (MinEB) instances, which they solve by applying known core-set techniques [6, 12]. Unfortunately, their cubic dependence on α and $1/\varepsilon$ undermines the practical applicability of their algorithm.

In Section 3, we use saddle-point optimization techniques to obtain an algorithm `MaxIBSPSolver` with running time $\widetilde{O}\big(md + m\sqrt{d}\alpha/\varepsilon\big)$. In other words, we reduce the dependence on both α and $1/\varepsilon$ from cubic to linear, and improve the running time by a factor up to $\sqrt{d}\alpha^2/\varepsilon^2$. We emphasize that our improvement could be significant in the views of theoretical computer scientists, operations researchers, as well as experimentalists:

- In theoretical computer science, one usually views α and ε as large constants so our improvement can be seen as $\widetilde{\Omega}(\sqrt{d})$ if one ignores the input reading time $O(md)$.
- In operations research or statistics, one usually concentrate on the convergence rate which is the ε dependence (recall that the seminal work of Nesterov is only to reduce $1/\varepsilon$ to $1/\sqrt{\varepsilon}$ [15]). Our improvement in this paper is from $1/\varepsilon^3$ to $1/\varepsilon$.
- In practice, if α is 10 for the polyhedron, ε is 10%, and the dimension $d = 100$, our method could potentially be 10^5 times faster than that of [20]. We leave it a future work to inspect the practical performance of our method on real-life datasets.

In the full version of this paper [3], we also apply convex (rather than saddle-point)

optimization and obtain a *parallel* algorithm `MaxIBConvexSolver` with slightly slower *total* running time $\widetilde{O}(md\alpha/\varepsilon)$. However, in terms of *parallel* running time (i.e., the number of parallelizable iterations, a classical benchmark used by iterative solvers [4]), `MaxIBConvexSolver` improves the result of [20] by a factor $\widetilde{\Omega}(\alpha^2/\varepsilon^2)$.

Minimum Enclosing Ball (MinEB). In the MinEB problem, we are given a set $\{a_1, a_2, \ldots, a_n\} \subseteq \mathbb{R}^d$ of points in the d-dimensional space and are asked to find a point $x \in \mathbb{R}^d$ so that its maximum distance to all the n points is at least $(1+\varepsilon)R_{\mathsf{opt}}$, where R_{opt} is the radius of a minimum enclosing ball that contains all the points in this set.

As originally studied by Sylvester in [18], the problem of MinEB has found numerous applications in fields such as data mining, learning, statistics, and computer graphics. In particular, the relationship between MinEB and support vector machines (SVMs) has been recently emphasized by [11, 10, 7, 17]. Efficient algorithms for this problem are both of theoretical and practical importance.

If the dimension d is constant, the algorithm of Welzl [19] solves MinEB exactly in linear time. Unfortunately, its dependency on d is exponential.

For large dimensions, a sequence of works based on the core-set technique [6, 12, 5, 21, 7] has given algorithms whose best known running time is $O(nd/\varepsilon)$. This running time is tight for the core-set technique, as, in the worst-case, the size of a coreset of MinEB is at least $\Omega(1/\varepsilon)$ [5]. Another type of algorithm due to Clarkson, Hazan, and Woodruff [8] achieves a running time of $\widetilde{O}(n/\varepsilon^2 + d/\varepsilon)$. This algorithm is fast for large values of ε, but may not be suitable for very small ε. All these cited algorithms converge at best in $O(1/\varepsilon)$ iterations.

Recently, Saha, Vishwanathan, and Zhang [17] designed two algorithms for MinEB that successfully overcame this $1/\varepsilon$ barrier. Using our ε-notation for multiplicative error, they give one algorithm which works in the ℓ_2-norm and achieves a running time of $O(ndQ/\sqrt{\varepsilon})$, and another algorithm which works in the ℓ_1-norm and achieves a running time of $O(nd\sqrt{\log n}L/\sqrt{\varepsilon})$. While the values of Q and L depend on the input structure, we observe that Q can be as large as $O(\sqrt{n})$, while L is never larger than a constant. In other words, their proposed algorithms have worst-case running times $O(n^{1.5}d/\sqrt{\varepsilon})$ and $O(nd\sqrt{\log n}/\sqrt{\varepsilon})$. The key component behind the result of Saha, Vishwanathan, and Zhang is the excessive gap framework of Nesterov [16], which is a primal-dual first-order approach for structured non-smooth optimization problems.

In Section 4, we rewrite MinEB as a saddle-point optimization problem, and obtain an algorithm `MinEBSPSolver` that runs in $\widetilde{O}(nd + n\sqrt{d}/\sqrt{\varepsilon})$. This is faster than the previous algorithm [17] by a factor up to $\sqrt{d}$, and faster than the popular core-set algorithm by a factor up to $\sqrt{d}/\sqrt{\varepsilon}$.

As an additional result, in the full version of this paper [3], we also observe that MinEB can be directly formulated as a convex (rather than saddle-point) optimization problem, and get an algorithm `MinEBConvexSolver` matching the running time of [17] but with much simpler analysis.

Remark. For both MaxIB and MinEB, one can also use interior-point types of algorithms to obtain a convergence rate of $\log(1/\varepsilon)$. However, this fast convergence rate comes at the cost of having expensive iterations: each iteration typically requires solving a linear equation system in the input size, making it impractical for very-large-scale inputs. Therefore, in this paper, we choose to focus on iterative methods whose iterations run in nearly-linear time.

1.1 Our Techniques

Our `MaxIBSPSolver` and `MinEBSPSolver` rely on (min-max) saddle-point optimization to solve MaxIB and MinEB respectively. More specifically, we reduce MaxIB and MinEB to solving the regularized saddle-point program:

$$\max_{x\in\mathbb{R}^d}\min_{y\in\Delta_m}\frac{1}{d}y^TAx+\frac{1}{d}y^Tb+\lambda H(y)-\frac{\gamma}{2}\|x\|_2^2\ ,$$

where $H(\cdot)$ is the entropy function over m-dimensional probabilities vectors, and $\lambda,\gamma>0$ are fixed regularization parameters. We call this ℓ_1-ℓ_2 saddle-point optimization because, borrowing language from optimization, this objective is strongly convex with respect to the ℓ_1 norm on the y side and strongly concave with respect to the ℓ_2 norm on the x side.

To solve this saddle-point problem efficiently, we iteratively update x and y. In particular, in each iteration we update x by a *random coordinate*, and update y fully using multiplicative weight updates. Therefore, this method can be viewed as an *accelerated, coordinate-based*, first-order method for saddle-point optimization. To the best of our knowledge, the only previously known accelerated, coordinate-based method on saddle-point optimization was SPDC [22], one of the state-of-the-art algorithms used for empirical risk minimizations in machine learning. We call our algorithm `L1L2SPSolver`.

A Surprising Hadamard Rotation. Unfortunately, solely applying `L1L2SPSolver` does not solve MinEB or MaxIB fast enough. In particular, the running time of `L1L2SPSolver` relies on the largest absolute values of A's entries. If the entries of A are very non-uniform – say, with a few very large entries and mostly small ones – the performance could be somewhat unsatisfactory. (In particular, we no longer have a $\sqrt{d}$ factor speed-up.)

To overcome this difficulty, we apply a randomized Hadamard transformation on A to uniformize its entries, so that all entries of A are relatively small. This transformation is inspired by the fast Johnson-Lindenstrauss transform [2] proposed for numerical linear algebra and compressive sensing purposes, and is another crucial ingredient behind our running time improvements.

Surprisingly, this Hadamard rotation comes solely from our optimization view but not the geometry. Indeed, it is not immediately clear why MaxIB or MinEB, as geometric problems, should be easier to solve if we rotate the space by a unitary (Hadamard) matrix.

Our Contributions. We summarize the main contributions of this paper as follows:

- We provide significantly faster algorithms on MaxIB and MinEB.
- This is the first time coordinate-based saddle-point optimization algorithm is applied to MaxIB, MinEB, or perhaps to any computational geometry problem.
- Since the ℓ_1-ℓ_2 saddle-point problem seems very natural, our `L1L2SPSolver` method can potentially lead to other applications in the future.
- The speed-up we obtained from the Hadamard rotation is an algebraic technique but applied to geometric problems. It sheds lights on solving perhaps more geometric problems faster using optimization insights.

2 Main Body of This Paper

We defer all the mathematical details of this paper to `http://arxiv.org/abs/1412.1001` [3].

Acknowledgements. We thank Lorenzo Orecchia for insightful comments and discussions. He took an active part in stages of this research, and yet declined to be a co-author. We thank Sepideh Mahabadi and Jinhui Xu for helpful conversations. We also thank the anonymous reviewers' valuable comments on the earlier versions of this paper.

References

1 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 52:1–30, 2005.

2 Nir Ailon and Bernard Chazelle. Faster dimension reduction. *Communications of the ACM*, 53(May):97–104, 2010. doi:10.1145/1646353.1646379.

3 Zeyuan Allen-Zhu, Zhenyu Liao, and Yang Yuan. Optimization Algorithms for Faster Computational Geometry. *ArXiv e-prints*, abs/1412.1001, December 2014.

4 Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the 26th ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, 2015.

5 Mihai Bădoiu and Kenneth L Clarkson. Optimal core-sets for balls. *Computational Geometry*, 40(1):14–22, 2008.

6 Mihai Bădoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing – STOC'02*, page 250, New York, New York, USA, 2002. ACM Press. doi:10.1145/509907.509947.

7 Kenneth L Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):63, 2010.

8 Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff. Sublinear optimization for machine learning. *Journal of the ACM*, 59(5):23:1–23:49, October 2012. doi:10.1145/2371656.2371658.

9 Jack Elzinga and Thomas G. Moore. A central cutting plane algorithm for the convex programming problem. *Mathematical Programming*, 8(1):134–145, 1975.

10 Bernd Gärtner and Martin Jaggi. Coresets for polytope distance. In *Proceedings of the 25th annual symposium on computational geometry*, pages 33–42. ACM, 2009.

11 Sariel Har-Peled, Dan Roth, and Dav Zimak. Maximum margin coresets for active and noise tolerant learning. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 836–841. Morgan Kaufmann Publishers Inc., 2007.

12 Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *Journal of Experimental Algorithmics*, 8:1–29, January 2003. doi:10.1145/996546.996548.

13 Chungmok Lee and Sungsoo Park. Chebyshev center based column generation. *Discrete Applied Mathematics*, 159(18):2251–2265, 2011.

14 Katta G. Murty. $o(m)$ bound on number of iterations in sphere methods for lp. *Algorithmic Operations Research*, 7(1):30–40, 2012.

15 Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. In *Doklady AN SSSR (translated as Soviet Mathematics Doklady)*, volume 269, pages 543–547, 1983.

16 Yurii Nesterov. Excessive Gap Technique in Nonsmooth Convex Minimization. *SIAM Journal on Optimization*, 16(1):235–249, January 2005. doi:10.1137/S1052623403422285.

17 Ankan Saha, S. V. N. Vishwanathan, and Xinhua Zhang. New Approximation Algorithms for Minimum Enclosing Convex Shapes. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms – SODA'11*, pages 1146–1160, September 2011. arXiv:0909.1062.

18 James Joseph Sylvester. A question in the geometry of situation. *Quarterly Journal of Pure and Applied Mathematics*, 1, 1857.

19 Emo Welzl. *Smallest enclosing disks (balls and ellipsoids)*. Springer, 1991.

20 Yulai Xie, Jack Snoeyink, and Jinhui Xu. Efficient algorithm for approximating maximum inscribed sphere in high dimensional polytope. In *Proceedings of the 22nd annual symposium on computational geometry – SCG'06*, 2006. doi:10.1145/1137856.1137861.

21 E. Alper Yildirim. Two algorithms for the minimum enclosing ball problem. *SIAM Journal on Optimization*, 19(3):1368–1391, 2008.

22 Yuchen Zhang and Lin Xiao. Stochastic Primal-Dual Coordinate Method for Regularized Empirical Risk Minimization. In *Proceedings of the 32nd International Conference on Machine Learning*, ICML 2015, 2015. arXiv:1409.3257.

A Fast Distributed Stateless Algorithm for α-Fair Packing Problems*

Jelena Marašević[1], Clifford Stein[2], and Gil Zussman[3]

1 **Columbia University, New York, NY, USA**
`jelena@ee.columbia.edu`
2 **Columbia University, New York, NY, USA**
`cliff@ieor.columbia.edu`
3 **Columbia University, New York, NY, USA**
`gil@ee.columbia.edu`

Abstract

We study weighted α-fair packing problems, that is, the problems of maximizing the objective functions (i) $\sum_j w_j x_j^{1-\alpha}/(1-\alpha)$ when $\alpha > 0$, $\alpha \neq 1$ and (ii) $\sum_j w_j \ln x_j$ when $\alpha = 1$, over linear constraints $Ax \leq b$, $x \geq 0$, where w_j are positive weights and A and b are non-negative. We consider the distributed computation model that was used for packing linear programs and network utility maximization problems. Under this model, *we provide a distributed algorithm for general* α that converges to an $\varepsilon-$approximate solution in time (number of distributed iterations) that has an inverse polynomial dependence on the approximation parameter ε and poly-logarithmic dependence on the problem size. *This is the first distributed algorithm for weighted α-fair packing with poly-logarithmic convergence in the input size.* The algorithm uses simple local update rules and is stateless (namely, it allows asynchronous updates, is self-stabilizing, and allows incremental and local adjustments). We also obtain a number of structural results that characterize α-fair allocations as the value of α is varied. These results deepen our understanding of fairness guarantees in α-fair packing allocations, and also provide insight into the behavior of α-fair allocations in the asymptotic cases $\alpha \to 0$, $\alpha \to 1$, and $\alpha \to \infty$.

1998 ACM Subject Classification F.2.1 [Theory of Computation] Analysis of Algorithms and Problem Complexity – Numerical Algorithms and Problems, G.1.6 [Mathematics of Computing] Numerical Analysis – Optimization, Convex programming, Gradient methods

Keywords and phrases Fairness, distributed and stateless algorithms, resource allocation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.54

1 Introduction

Over the past two decades, *fair resource allocation* problems have received considerable attention in many application areas, including Internet congestion control [32], rate control in software defined networks [36], scheduling in wireless networks [45], multi-resource allocation and scheduling in datacenters [12, 20, 24, 21], and a variety of applications in operations research, economics, and game theory [11, 23]. In most of these applications, positive linear (packing) constraints arise as a natural model of the allowable allocations.

* The full version of the paper can be found at `http://arxiv.org/abs/1502.03372v3`. This work was partially supported by the NSF grants CNS-14-23105, CCF-1349602, and CCF-1421161, and the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement n°[PIIF-GA-2013-629740].11.

In this paper, we focus on the problem of finding an *α-fair* vector on the set determined by packing constraints $Ax \leq \mathbb{1}, x \geq 0$ where all $A_{ij} \geq 0$.[1] We refer to this problem as α-fair packing. For a vector of positive weights w and $\alpha \geq 0$, an allocation vector x^* of size n is weighted α-fair , if it maximizes $p_\alpha(x) = \sum_j w_j f_\alpha(x_j)$ [38], where:

$$f_\alpha(x_j) = \begin{cases} \ln(x_j), & \text{if } \alpha = 1 \\ \frac{x_j^{1-\alpha}}{1-\alpha}, & \text{if } \alpha \neq 1 \end{cases}. \tag{1}$$

α-fairness provides a trade-off between efficiency (sum of allocated resources) and fairness (minimum allocated resource) as a function of α: the higher the α, the higher the fairness and the lower the efficiency [4, 11, 31]. Important special cases are max-min fairness ($\alpha \to \infty$) and proportional fairness ($\alpha = 1$). When $\alpha = 0$, we have the "unfair" case of linear optimization.

Distributed algorithms for α-fair packing are of particular interest, as many applications are inherently distributed (e.g., network congestion control), while others require parallelization due to the large problem size (e.g., resource allocation in datacenters). We adopt the model of distributed computation commonly used in packing linear programming (LP) algorithms [7, 3, 8, 29, 33, 41] and which generalizes the model from network congestion control [26]. In this model, an agent j controls the variable x_j and has information about: (i) the j^{th} column of the $m \times n$ constraint matrix A, (ii) the weight w_j, (iii) upper bounds on global problem parameters $m, n, w_{\max}$, and $A_{\max}$, where $w_{\max} = \max_j w_j$, and $A_{\max} = \max_{ij} A_{ij}$, and (iv) in each round, the relative slack of each constraint i in which x_j takes part.

Distributed algorithms for α-fair resource allocations have been most widely studied in the network congestion control literature, using a control-theoretic approach [25, 26, 45, 38, 32]. Such an approach yields continuous-time algorithms that converge after "finite" time; however, the convergence time of these algorithms as a function of the input size is poorly understood. Some other distributed pseudo-polynomial-time approximation algorithms that can address α-fair packing are described in Table 1. These algorithms all have convergence times that are at least linear in the parameters describing the problem.

No previous work has given truly fast (poly-log iterations) distributed algorithms for the general case of α-fair packing. Only for the unfair $\alpha = 0$ case (packing LPs), are such algorithms known [7, 33, 8, 46, 29, 3].

Our Results. *We provide the first efficient, distributed, and stateless algorithm for weighted α-fair packing*, namely, for the problem $\max\{p_\alpha(x) : Ax \leq \mathbb{1}, x \geq 0\}$, where distributed agents update the values of x_j's asynchronously and react only to the current state of the constraints. We assume that all non-zero entries A_{ij} of matrix A satisfy $A_{ij} \geq 1$. Considering such a normalized form of the problem is without loss of generality (see Appendix A in [35]).

The approximation provided by the algorithm, to which we refer as the ε-approximation, is (i) $(1+\varepsilon)$-multiplicative for $\alpha \neq 1$, and (ii) $W\varepsilon$-additive[2] for $\alpha = 1$, where $W = \sum_j w_j$. The main results are summarized in the following theorem, where, to unify the statement of the results, we treat α as a constant that is either equal to 1 or bounded away from 0 and 1, and we also loosen the bound in terms of $\varepsilon^{-1}, n, m, R_w = \max_{j,k} w_j/w_k$, and $A_{\max}$. For a more detailed statement, see Theorems 4.1–4.3.

[1] Although in the network congestion control literature A is commonly assumed to be a 0-1 matrix [25, 26, 45, 38, 32], important applications (such as, e.g., multi-resource allocation in datacenters) are modeled by a more general constraint matrix A with arbitrary non-negative elements [12, 20, 24, 21].

[2] Note that W cannot be avoided here, as additive approximation is not invariant to the objective scaling.

▶ **Theorem 1.1** (Main Result). *Given a weighted α-fair packing problem* $\max\{\sum_j w_j f_\alpha(x_j) : Ax \leq \mathbb{1}, x \geq 0\}$*, there exists a stateless and distributed algorithm (α-FairPSolver) that computes an ε-approximate solution in $O(\varepsilon^{-5}\ln^4(R_w nmA_{\max}\varepsilon^{-1}))$ rounds.*

To the best of our knowledge, for any constant approximation parameter ε, our algorithm is the first distributed algorithm for weighted α-fair packing problems with a poly-logarithmic convergence time.[3] The algorithm is *stateless* according to the definition given by Awerbuch and Khandekar [7, 6]: it starts from any initial state, the agents update the variables x_j in a cooperative but uncoordinated manner, reacting only to the current state of the constraints that they observe, and without access to a global clock. Statelessness implies various desirable properties of a distributed algorithm, such as: asynchronous updates, self-stabilization, and incremental and local adjustments [7, 6].

We also obtain the following structural results that characterize α-fair packing allocations as a function of the value of α:

- We derive a lower bound on the minimum coordinate of the α-fair packing allocation as a function of α and the problem parameters (Lemma 4.12). This bound deepens our understanding of how the fairness (a minimum allocated value) changes with α.
- We prove that for $\alpha \leq \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, α-fair packing can be $O(\varepsilon)-$approximated by any $\varepsilon-$approximation packing LP solver (Lemma 4.13).
- We show that for $|\alpha-1| = O(\varepsilon^2/\ln^2(\varepsilon^{-1}R_w mnA_{\max}))$, α-fair allocation is $\varepsilon-$approximated by a $1-$fair allocation returned by our algorithm (Lemmas 4.14 and 4.15).
- We show that for $\alpha \geq \ln(R_w nA_{\max})/\varepsilon$, the α-fair packing allocation x^* and the max-min fair allocation z^* are ε-close to each other: $(1-\varepsilon)z^* \leq x^* \leq (1+\varepsilon)z^*$ element-wise. This result is especially interesting as (i) max-min fair packing is not a convex problem, but rather a multi-objective problem (see, e.g., [27, 43]) and (ii) the result yields the first convex relaxation of max-min fair allocation problems with a $1 \pm \varepsilon$ gap.

We now overview some of the main technical details of α-FairPSolver. In doing so, we point out connections to the two main bodies of previous work, from packing LPs[7] and network congestion control [25]. We also outline the new algorithmic ideas and proofs.

The algorithm and KKT conditions. The algorithm maintains primal and dual feasible solutions and updates each primal variable x_j whenever a Karush-Kuhn-Tucker (KKT) condition ${x_j}^\alpha \sum_i y_i A_{ij} = w_j$ is not *approximately* satisfied. In previous work, relevant update rules include: [25] (for $\alpha = 1$), where the update of each variable x_j is proportional to the difference $w_j - x_j \sum_i y_i A_{ij}$, and [7] (for $\alpha = 0$), where each x_j is updated by a multiplicative factor $1 \pm \beta$, whenever $\sum_i y_i A_{ij} = w_j$ is not approximately satisfied. For our techniques (addressing a general α) such rules do not suffice and we introduce the following modifications: (i) in the $\alpha < 1$ case we use multiplicative updates by factors $(1+\beta_1)$ and $(1-\beta_2)$, where $\beta_1 \neq \beta_2$ and (ii) we use additional threshold values δ_j to make sure that x_j's do not become too small. These thresholds guarantee that we maintain a feasible solution, but they significantly complicate (compared to the linear case) the argument that each step makes a significant progress.

[3] The total amount of work per (distributed) round is linear in the number of non-zero entries of the constraint matrix A, which matches the best bound achieved in previous work on distributed packing LPs [7, 33, 8, 46, 29, 3] and distributed network utility maximization [9, 39].

Table 1 Comparison among distributed algorithms for α-fair packing.

Paper	Number of Distributed Iterations[4]	Statelessness	Notes
[15]	$\Omega(\varepsilon^{-1}nA_{\max})$	Semi-stateless[5]	Only for $\alpha = 1$
[9]	$\Omega(\varepsilon^{-1}mn{A_{\max}}^2)$	Not stateless	
[39]	$\text{poly}(\varepsilon^{-1}, m, n, A_{\max})$	Semi-stateless	
[this work]	$O(\varepsilon^{-5}\ln^4(R_w mnA_{\max}/\varepsilon))$	Stateless	

Dual Variables. In α-FairPSolver, a dual variable y_i is an exponential function of the i^{th} constraint's relative slack: $y_i(x) = C \cdot e^{\kappa(\sum_j A_{ij}x_j - 1)}$, where C and κ are functions of global input parameters $\alpha, w_{\max}, n, m$, and $A_{\max}$. Packing LP algorithms [7, 3, 42, 8, 18, 17, 28] use similar dual variables with $C = 1$. Our work requires choosing C to be a function of $\alpha, w_{\max}, n, m, A_{\max}$ rather than a constant.

Convergence Argument. The convergence analysis of α-FairPSolver relies on the appropriately chosen concave potential function that is bounded below and above for $x_j \in [\delta_j, 1]$, $\forall j$, and that increases with every primal update. The algorithm can also be interpreted as a gradient ascent on a regularized objective function (the potential function), using a generalized entropy regularizer (see [3, 1]). A similar potential function was used in many works on packing and covering linear programs, such as, e.g., in [7] and (implicitly) in [46]. The Lyapunov function from [25] is also equivalent to this potential function when $y_i(x) = C \cdot e^{\kappa(\sum_j A_{ij}x_j - 1)}$, $\forall i$. As in these works, the main idea in the analysis is to show that whenever a solution x is not "close" to the optimal one, the potential function increases substantially. However, our work requires several new ideas in the convergence proofs, the most notable being *stationary rounds*. A stationary round is roughly a time when the variables x_j do not change much and are close to the optimum. Poly-logarithmic convergence time is then obtained by showing that: (i) there is at most a poly-logarithmic number of non-stationary rounds where the potential function increases additively and the increase is "large enough", and (ii) in all the remaining non-stationary rounds, the potential function increases multiplicatively. Our use of stationary rounds is new, as is the use of Lagrangian duality and all the arguments that follow (see [35] for a detailed discussion).

Relationship to Previous Work. Very little progress has been made in the design of efficient distributed algorithms for the general class of α-fair objectives. Classical work on distributed rate control algorithms in the networking literature uses a control-theoretic approach to optimize α-fair objectives. While such an approach has been extensively studied [25, 26, 45, 38, 32], it has never been proven to lead to a polynomial convergence time.

Since α-fair objectives are concave, their optimization over a region determined by linear constraints is solvable in polynomial time in a centralized setting through convex programming (see, e.g., [13, 40]). Distributed gradient methods for network utility maximization problems, such as e.g., [9, 39] summarized in Table 1, can be applied to α-fair packing. However, the convergence times of these algorithms depend on the dual gradient's Lipschitz constant to produce good approximations. While [9, 39] provide a better dependence on the accuracy ε than our work, the dependence on the dual gradient's Lipschitz constant, in general, leads to at least linear convergence time as a function of n, m, and $A_{\max}$.

As mentioned before, some special cases have been addressed, particularly max-min fairness ($\alpha \to \infty$) and packing LPs ($\alpha = 0$). Relevant work on max-min fairness includes

[10, 22, 30, 27, 37, 34, 14], but none of these works have poly-logarithmic convergence time. There is a long history of interesting work on packing LPs in both centralized and distributed settings, e.g., [1, 42, 28, 18, 7, 33, 8, 46, 29, 3, 19]. Only a few of these works are stateless, including the packing LP algorithm of Awerbuch and Khandekar [7], flow control algorithm of Garg and Young [19], and the algorithm of Awerbuch, Azar, and Khandekar [5] for the special case of load balancing in bipartite graphs. Additionally, the packing LP algorithm of Allen-Zhu and Orecchia [3] is "semi-stateless"; the lacking property to make it stateless is that it requires synchronous updates. The $\alpha = 1$ case of α-fair packing problems is equivalent to the problem of finding an equilibrium allocation in Eisenberg-Gale markets with Leontief utilities [15]. Similar to the aforementioned algorithms, the algorithm from [15] converges in time linear in ε^{-1} but also (at least) linear in the input size (see Table 1).

2 Preliminaries

Weighted α-Fair Packing. Consider the following optimization problem with positive linear (packing) constraints: $(Q_\alpha) = \max\{p_\alpha(x) \equiv \sum_{j=1}^{n} w_j f_\alpha(x_j) : Ax \leq b, x \geq 0\}$, where $f_\alpha(x_j)$ is given by (1), $x = (x_1, \ldots, x_n)$ is the vector of variables, A is an $m \times n$ matrix with non-negative elements, and $b = (b_1, \ldots, b_m)$ is a vector with strictly positive[6] elements. We refer to (Q_α) as the weighted α-fair packing. The following definition and lemma introduced by Mo and Walrand [38] characterize weighted α-fair allocations. In the rest of the paper, we will use the terms weighted α-fair and α-fair interchangeably.

▶ **Definition 2.1** ([38]). Let w be a vector with positive entries and $\alpha > 0$. A vector x is weighted α-fair, if it is feasible and for any other feasible vector x: $\sum_{j=1}^{n} w_j \frac{x_j - x_j^*}{x_j^{*\alpha}} \leq 0$.

▶ **Lemma 2.2** ([38]). *A vector x^* solves (Q_α) if and only if it is weighted α-fair.*

Notice in (Q_α) that since $b_i > 0$, $\forall i$, and the partial derivative of the objective with respect to any of the variables x_j goes to ∞ as $x_j \to 0$, the optimal solution must lie in the positive orthant. Moreover, since the objective is strictly concave and maximized over a convex region, the optimal solution is unique and (Q_α) satisfies strong duality (see, e.g., [13]). The same observations are true for the scaled version of the problem denoted by (P_α) and introduced in the following subsection.

Normalized Form. We consider weighted α-fair packing in the normalized form:

$$(P_\alpha) = \max\{p_\alpha(x) : Ax \leq \mathbb{1}, x \geq 0\},$$

where $p_\alpha(x) = \sum_{j=1}^{n} w_j f_\alpha(x_j)$, f_α is defined by (1), $w = (w_1, \ldots, w_n)$ is a vector of positive weights, $x = (x_1, \ldots, x_n)$ is the vector of variables, A is an $m \times n$ matrix with non-negative entries, and $\mathbb{1}$ is a size-m vector of 1's. We let $A_{\max}$ denote the maximum element of the constraint matrix A, and assume that every entry A_{ij} of A is non-negative, and moreover, that $A_{ij} \geq 1$ whenever $A_{ij} \neq 0$. The maximum weight is denoted by $w_{\max}$ and the minimum weight is denoted by $w_{\min}$. The sum of the weights is denoted by W and the ratio $\frac{w_{\max}}{w_{\min}}$ by R_w. We remark that considering (Q_α) in the normalized form (P_α) is without loss of generality: any problem (Q_α) can be scaled to this form by (i) dividing both sides of each inequality i by b_i and (ii) working with scaled variables $c \cdot x_j$, where $c = \min\{1, \min_{\{i,j:A_{ij}\neq 0\}} \frac{A_{ij}}{b_i}\}$. Moreover, such scaling preserves the approximation (see [35]).

[6] If, for some i, $b_i = 0$, then trivially $x_j = 0$, for all j such that $A_{ij} \neq 0$.

Model of Distributed Computation. We adopt the same model of distributed computation as [7, 3, 8, 29, 33, 41], described as follows. We assume that for each $j \in \{1, \ldots, n\}$, there is an agent controlling the variable x_j. Agent j is assumed to have information about the following problem parameters: (i) the j^{th} column of A, (ii) the weight w_j, and (iii) (an upper bound on) $m, n, w_{\max}$, and $A_{\max}$. In each round, agent j collects the relative slack[7] $1 - \sum_{j=1}^{n} A_{ij}x_j$ of all constraints i for which $A_{ij} \neq 0$.

We remark that this model of distributed computation is a generalization of the model considered in network congestion control problems [26] where a variable x_j corresponds to the rate of node j, A is a 0-1 routing matrix, such that $A_{ij} = 1$ if and only if a node j sends flow over link i, and b is the vector of link capacities. Under this model, the knowledge about the relative slack of each constraint corresponds to each node collecting (a function of) congestion on each link that it utilizes. Such a model was used in network utility maximization problems with α-fair objectives [25] and general strongly-concave objectives [9].

KKT Conditions and Duality Gap. We will denote the Lagrange multipliers for (P_α) as $y = (y_1, \ldots, y_m)$ and refer to them as "dual variables". The KKT conditions for (P_α) are:

$$\sum_{j=1}^{n} A_{ij}x_j \leq 1,\ \forall i \in \{1, \ldots, m\};\ x_j \geq 0,\ \forall j \in \{1, \ldots, n\} \text{ (primal feasibility)} \tag{K1}$$

$$y_i \geq 0,\ \forall i \in \{1, \ldots, m\} \text{ (dual feasibility)} \tag{K2}$$

$$y_i \cdot \Big(\sum_{j=1}^{m} A_{ij}x_j - 1 \Big) = 0,\ \forall i \in \{1, \ldots, m\} \text{ (complementary slackness)} \tag{K3}$$

$${x_j}^{\alpha} \sum_{i=1}^{m} y_i A_{ij} = w_j,\ \forall j \in \{1, \ldots, m\} \text{ (gradient conditions)} \tag{K4}$$

The duality gap for $\alpha \neq 1$ is (see Appendix B in [35]):

$$G_\alpha(x, y) = \sum_{j=1}^{n} w_j \frac{{x_j}^{1-\alpha}}{1-\alpha} \left({\xi_j}^{\frac{\alpha-1}{\alpha}} - 1 \right) + \sum_{i=1}^{m} y_i - \sum_{j=1}^{n} w_j x_j^{1-\alpha} \cdot {\xi_j}^{\frac{\alpha-1}{\alpha}}, \tag{2}$$

where $\xi_j = \frac{{x_j}^{\alpha} \sum_{i=1}^{m} y_i A_{ij}}{w_j}$, while for $\alpha = 1$:

$$G_1(x, y) = -\sum_{j=1}^{n} w_j \ln \left(\frac{x_j \sum_{i=1}^{m} y_i A_{ij}}{w_j} \right) + \sum_{i=1}^{m} y_i - W. \tag{3}$$

3 Algorithm

The pseudocode for the α-FairPSolver algorithm run at each node j is provided in Fig. 1. The basic intuition is that the algorithm keeps KKT conditions (K1) and (K2) satisfied and works towards (approximately) satisfying the remaining two KKT conditions (K3) and (K4) to minimize the duality gap. The algorithm can run in the distributed setting described in Section 2. In each round, an agent j updates the value of x_j based on the relative slack of all the constraints in which j takes part, as long as the KKT condition (K4) of agent j is not approximately satisfied. The updates need not be synchronous: we will require that all agents make updates at the same speed, but without access to a global clock.

[7] The slack is "relative" because in a non-scaled version of the problem where one could have $b_i \neq 1$, agent j would need to have information about $\frac{b_i - \sum_{j=1}^{n} A_{ij}x_j}{b_i}$.

α-FairPSolver(ε)

(Parameters $\delta_j, C, \kappa, \gamma, \beta_1$, and β_2 are set as described in the text below the algorithm.)

In each round of the algorithm:

1: $x_j \leftarrow \max\{x_j, \delta_j\}$, $x_j = \min\{x_j, 1\}$

2: Update the dual variables: $y_i = C \cdot e^{\kappa\left(\sum_{j=1}^{n} A_{ij}x_j - 1\right)}$ $\forall i \in \{1, \ldots, m\}$

3: **if** $\frac{{x_j}^{\alpha} \cdot \sum_{i=1}^{m} y_i A_{ij}}{w_j} \leq (1-\gamma)$ **then**

4: $\quad x_j \leftarrow x_j \cdot (1+\beta_1)$

5: **else**

6: $\quad$ **if** $\frac{{x_j}^{\alpha} \cdot \sum_{i=1}^{m} y_i A_{ij}}{w_j} \geq (1+\gamma)$ **then**

7: $\quad\quad x_j \leftarrow \max\{x_j \cdot (1-\beta_2), \delta_j\}$

Figure 1 Pseudocode of α-FairPSolver algorithm.

To allow for self-stabilization and dynamic changes, the algorithm runs forever at all the agents, which is a standard requirement for self-stabilizing algorithms (see, e.g., [16]). The convergence of the algorithm is measured as the number of rounds between the round in which the algorithm starts from some initial solution and the round in which it reaches an $\varepsilon-$approximate solution, assuming that there are no hard reset events or node/constraint insertions/deletions in between.

Without loss of generality, we assume that the input parameter ε that determines the approximation quality satisfies $\varepsilon \leq \min\{\frac{1}{6}, \frac{9}{10\alpha}\}$ for any α, and $\varepsilon \leq \frac{1-\alpha}{\alpha}$ for $\alpha < 1$. The parameters $\delta_j, C, \kappa, \gamma$, β_1, and β_2 are set as follows. For technical reasons (mainly due to reinforcing dominant multiplicative updates of the variables x_j), we set the values of the lower thresholds δ_j below the actual lower bound of the optimal solution that we derive in Lemma 4.12:

$$\delta_j = \left(\frac{1}{2} \cdot \frac{w_j}{w_{\max}}\right)^{1/\alpha} \cdot \begin{cases} \left(\frac{1}{m \cdot n^2 \cdot A_{\max}}\right)^{1/\alpha}, & \text{if } 0 < \alpha \leq 1 \\ \frac{1}{m \cdot n^2 {A_{\max}}^{2-1/\alpha}}, & \text{if } \alpha > 1 \end{cases}.$$

We denote $\delta_{\max} \equiv \max_j \delta_j$, $\delta_{\min} \equiv \min_j \delta_j$. The constant C that multiplies the exponent in the dual variables y_i is chosen as $C = \frac{W}{\sum_{j=1}^{n} {\delta_j}^{\alpha}}$. Because δ_j only depends on w_j and on global parameters, we also have $C = \frac{w_j}{{\delta_j}^{\alpha}}$, $\forall j$. The parameter κ that appears in the exponent of the y_i's is chosen as $\kappa = \frac{1}{\varepsilon} \ln\left(\frac{CmA_{\max}}{\varepsilon w_{\min}}\right)$. The "absolute error" of (K4) γ is set to $\varepsilon/4$. For $\alpha \geq 1$, we set $\beta_1 = \beta_2 = \beta$, where the choice of β is described below. For $\alpha < 1$, we set $\beta_1 = \beta$, $\beta_2 = \beta^2(\ln(\frac{1}{\delta_{\min}}))^{-1}$.

Similar to [7], we choose the value of β so that if we set $\beta_1 = \beta_2 = \beta$, in any round the value of each $\frac{{x_j}^{\alpha} \sum_{i=1}^{m} y_i(x) A_{ij}}{w_j}$ changes by a multiplicative factor of at most $(1 \pm \gamma/4)$. Since the maximum increase over any x_j in each iteration is by a factor $1+\beta$, and x is feasible in each round (see Lemma 4.4), we have that $\sum_{j=1}^{n} A_{ij}x_j \leq 1$, and therefore, the maximum increase in each y_i is by a factor of $e^{\kappa\beta}$. A similar argument holds for the maximum decrease. Hence, we choose β so that:

$$(1+\beta)^{\alpha} e^{\kappa\beta} \leq 1 + \gamma/4 \quad \text{and} \quad (1-\beta)^{\alpha} e^{-\kappa\beta} \geq 1 - \gamma/4,$$

and it suffices to set:

$$\beta = \begin{cases} \frac{\gamma}{5(\kappa+1)}, & \text{if } \alpha \leq 1 \\ \frac{\gamma}{5(\kappa+\alpha)}, & \text{if } \alpha > 1 \end{cases}.$$

▶ Remark. In the $\alpha < 1$ cases, since $\beta_2 = \beta^2(\ln(1/\delta_{\min}))^{-1}$, the maximum decrease in $\frac{{x_j}^\alpha \sum_i y_i(x)A_{ij}}{w_j}$ is by a factor $(1 - (\gamma/4) \cdot \beta(\ln(1/\delta_{\min}))^{-1})$, $\forall j$.

4 Convergence Analysis

In this section, we analyze the convergence time of α-FairPSolver. We first state our main theorems and provide some general results that hold for all $\alpha > 0$. We show that starting from an arbitrary solution, the algorithm reaches a feasible solution within poly-logarithmic (in the input size) number of rounds, and maintains a feasible solution forever after. Similar to [7, 46, 25], we use a concave potential function that, for feasible x, is bounded below and above and increases with any algorithm update. Then, we sketch the proof of Theorem 4.3 ($\alpha > 1$), while we defer the full proofs of the three theorems to the full paper [35]. The main proof idea in all the cases is as follows. With an appropriate definition of a *stationary round* for each of the three cases $\alpha < 1$, $\alpha = 1$, and $\alpha > 1$, we show that in every stationary round, x approximates "well" the optimal solution by bounding the duality gap. On the other hand, for any non-stationary round, we show that the potential increases substantially. This large increase in the potential leads to the conclusion that there cannot be too many non-stationary rounds, thus bounding the overall convergence time.

We make a few remarks here. First, we require that α be bounded away from zero. This requirement is without loss of generality because we show that when $\alpha \leq \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, any $\varepsilon-$approximation LP provides a $3\varepsilon-$approximate solution to (P_α) (Lemma 4.13). Thus, when $\alpha \leq \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$ we can switch to the algorithm of [7], and when $\alpha > \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, the convergence time remains poly-logarithmic in the input size and polynomial in ε^{-1}. Second, the assumption that $\varepsilon \leq \frac{1-\alpha}{\alpha}$ in the $\alpha < 1$ case is also without loss of generality, because we show that when α is close to 1 (roughly, $1 - O(\varepsilon^2/\ln^2(R_w mnA_{\max}/\varepsilon))$), we can approximate (P_α) by switching to the $\alpha = 1$ case of the algorithm (Lemma 4.14). Finally, when $\alpha > 1$, the algorithm achieves an $\varepsilon-$approximation in time $O(\alpha^4\varepsilon^{-4}\ln^2(R_w nmA_{\max}\varepsilon^{-1}))$. We believe that a polynomial dependence on α is difficult to avoid in this setting, because by increasing α, the gradient of the α-fair utilities f_α blows up on the interval $(0, 1)$: as α increases, $f_\alpha(x)$ quickly starts approaching a step function that is equal to $-\infty$ on the interval $(0, 1]$ and equal to 0 on the interval $(1, \infty]$. To characterize the behavior of α-fair allocations as α becomes large, we show that when $\alpha \geq \varepsilon^{-1}\ln(R_w nA_{\max})$, all the coordinates of the α-fair vector are within a $1 \pm \varepsilon$ multiplicative factor of the corresponding coordinates of the max-min fair vector (Lemma 4.17).

Main Results. Our main results are summarized in the following three theorems. The objective is denoted by $p_\alpha(x)$, x^t denotes the solution at the beginning of round t, and x^* denotes the optimal solution.

▶ **Theorem 4.1** (Convergence for $\alpha < 1$)**.** α-FairPSolver *solves* (P_α) *approximately for* $\alpha < 1$ *in time that is polynomial in* $\frac{\ln(nmA_{\max})}{\alpha\varepsilon}$*. In particular, after at most*

$$O\left(\alpha^{-2}\varepsilon^{-5}\ln^2\left(R_w mnA_{\max}\right)\ln^2\left(\varepsilon^{-1}R_w mnA_{\max}\right)\right) \tag{4}$$

rounds, there exists at least one round t such that $p_\alpha(x^*) - p_\alpha(x^t) \leq \varepsilon p_\alpha(x^t)$. Moreover, the total number of rounds s in which $p_\alpha(x^*) - p_\alpha(x^s) > \varepsilon p_\alpha(x^s)$ is also bounded by (4).

▶ **Theorem 4.2** (Convergence for $\alpha = 1$)**.** α-FairPSolver *solves* (P_1) *approximately in time that is polynomial in* $\varepsilon^{-1}\ln(R_w nmA_{\max})$*. In particular, after at most*

$$O\left(\varepsilon^{-5}\ln^2\left(R_w nmA_{\max}\right)\ln^2\left(\varepsilon^{-1}R_w nmA_{\max}\right)\right) \tag{5}$$

rounds, there exists at least one round t such that $p(x^*) - p(x^t) \leq \varepsilon W$. Moreover, the total number of rounds s in which $p(x^*) - p(x^s) > \varepsilon W$ is also bounded by (5).

▶ **Theorem 4.3** (Convergence for $\alpha > 1$)**.** α-FairPSolver *solves (P_α) approximately for $\alpha > 1$ in time that is polynomial in $\varepsilon^{-1} \ln(nmA_{\max})$. In particular, after at most:*

$$O\left(\alpha^4 \varepsilon^{-4} \ln\left(R_w nmA_{\max}\right) \ln\left(\varepsilon^{-1} R_w nmA_{\max}\right)\right) \tag{6}$$

rounds, there exists at least one round t such that $p_\alpha(x^*) - p_\alpha(x^t) \leq \varepsilon(-p_\alpha(x^t))$. Moreover, the total number of rounds s in which $p_\alpha(x^*) - p_\alpha(x^s) > \varepsilon(-p_\alpha(x^s))$ is also bounded by (6).

Proofs of Theorem 4.1 and Theorem 4.2 are provided in the full paper [35]. We sketch the proof of Theorem 4.3 in Section 4.1.

Feasibility and Approximate Complementary Slackness. The following three lemmas are preliminaries for the convergence time analysis. Lemma 4.4 shows that starting from a feasible solution, the algorithm always maintains a feasible solution. Lemma 4.5 shows that any violated constraint becomes feasible within poly-logarithmic number of rounds, and remains feasible forever after. Combined with Lemma 4.4, Lemma 4.5 allows us to focus only on the rounds with feasible solutions x. Lemma 4.6 shows that after a poly-logarithmic number of rounds, approximate complementary slackness (KKT condition (K3)) holds in an aggregate sense: $\sum_{i=1}^m y_i(x)\left(\sum_{j=1}^n A_{ij}x_j - 1\right) \approx 0$. Proofs are provided in [35].

▶ **Lemma 4.4.** *If the algorithm starts from a feasible solution, then the algorithm maintains a feasible solution x: $x_j \geq 0$, $\forall j$ and $\sum_{j=1}^n A_{ij}x_j \leq 1$, $\forall i$, in each round.*

▶ **Lemma 4.5.** *If for any i: $\sum_{j=1}^n A_{ij}x_j > 1$, then after at most $\tau_1 = O(\frac{1}{\beta_2} \ln(nA_{\max}))$ rounds, it is always true that $\sum_{j=1}^n A_{ij}x_j \leq 1$.*

▶ **Lemma 4.6.** *If the algorithm starts from a feasible solution, then after at most $\tau_0 = \frac{1}{\beta} \ln\left(\frac{1}{\delta_{\min}}\right)$ rounds, it is always true that:*

1. *At least one constraint is approximately tight: $\max_i \left\{\sum_{j=1}^n A_{ij}x_j\right\} \geq 1 - (1 + 1/\kappa)\varepsilon$,*
2. *$\sum_{i=1}^m y_i \leq (1 + 3\varepsilon) \sum_{j=1}^n x_j \sum_{i=1}^m y_i A_{ij}$, and*
3. *$(1 - 3\varepsilon) \sum_{i=1}^m y_i \leq \sum_{j=1}^n x_j \sum_{i=1}^m y_i A_{ij} \leq \sum_{i=1}^m y_i$.*

Lemmas analogous to 4.4 and 4.6 also appear in [7]. However, the proofs of Lemmas 4.4 and 4.6 require new ideas compared to the proofs of the corresponding lemmas in [7]. We need to be much more careful in our choice of lower thresholds δ_j and constant C in the dual variables, particularly by choosing C as a function of several variables, rather than as a constant. The choice of δ_j's is also sensitive as smaller δ_j's would make the potential function range too large, while larger δ_j's would cause more frequent decrease of "small" variables. In either case, the convergence time would increase.

Decrease of Small Variables. The following lemma is also needed for the convergence analysis. It shows that if some variable x_j decreases by less than a multiplicative factor $(1 - \beta_2)$, i.e., $x_j < \frac{\delta_j}{1-\beta_2}$ and x_j decreases, then x_j must be part of at least one approximately tight constraint. This lemma will be used later to show that in any round the increase in the potential due to the decrease of "small" variables is dominated by the decrease of "large" variables (i.e., the variables that decrease by a multiplicative factor $(1 - \beta_2)$). The proof of Lemma 4.7 is provided in [35].

▶ **Lemma 4.7.** *Consider the rounds that happen after the initial* $\tau_1 = O(\frac{1}{\beta_2}\ln(nA_{\max}))$ *rounds. If in some round there is a variable* $x_j < \frac{\delta_j}{1-\beta_2}$ *that decreases, then in the same round for some* i *with* $A_{ij} \neq 0$ *it holds that:* $y_i(x) \geq \frac{\sum_{l=1}^{m} A_{lj} y_l(x)}{m A_{\max}}$ *and* $\sum_{k=1}^{n} A_{ik} x_k > 1 - \frac{\varepsilon}{2}$.

Potential. We use the following potential function to analyze the convergence time:

$$\Phi(x) = p_\alpha(x) - \frac{1}{\kappa}\sum_{i=1}^{m} y_i(x),$$

where $p_\alpha(x) = \sum_{j=1}^{n} w_j f_\alpha(x_j)$ and f_α is defined by (1). The potential function is strictly concave and its partial derivative with respect to any variable x_j is:

$$\frac{\partial \Phi(x)}{\partial x_j} = \frac{w_j}{{x_j}^\alpha} - \sum_{i=1}^{m} y_i(x) A_{ij} = \frac{w_j}{{x_j}^\alpha}\left(1 - \frac{{x_j}^\alpha \sum_{i=1}^{m} y_i(x) A_{ij}}{w_j}\right). \tag{7}$$

The following fact (given in a similar form in [7]), which follows directly from the Taylor series representation of concave functions, will be useful for the potential increase analysis:

▶ **Fact 4.8.** For a differentiable concave function $f : \mathbb{R}^n \to \mathbb{R}$ and any two points $x^0, x^1 \in \mathbb{R}^n$:

$$\sum_{j=1}^{n} \frac{\partial f(x^0)}{\partial x_j}(x_j^1 - x_j^0) \geq f(x^1) - f(x^0) \geq \sum_{j=1}^{n} \frac{\partial f(x^1)}{\partial x_j}(x_j^1 - x_j^0).$$

Using Fact 4.8 and (7), we show the following lemma:

▶ **Lemma 4.9.** *Starting with a feasible solution and throughout the course of the algorithm, the potential function* $\Phi(x)$ *never decreases. Letting* x^0 *and* x^1 *denote the values of* x *before and after a round update, respectively, the potential function increase is lower-bounded as:*

$$\Phi(x^1) - \Phi(x^0) \geq \sum_{j=1}^{n} w_j \frac{\left|x_j^1 - x_j^0\right|}{(x_j^1)^\alpha}\left|1 - \frac{(x_j^1)^\alpha \sum_{i=1}^{m} y_i(x^1) A_{ij}}{w_j}\right|.$$

4.1 Proof Sketch of Theorem 4.3

In this section, we outline the main ideas of the proof of Theorem 4.3, while the technical details are omitted and are instead provided in [35]. First, we show that in any round of the algorithm the variables that decrease by a multiplicative factor $(1 - \beta_2)$ dominate the potential increase due to *all the variables* that decrease (see Lemma 4.21 in [35]). This result is then used in Lemma 4.10 to show the following lower bound on the potential increase:

▶ **Lemma 4.10.** *Let* x^0 *and* x^1 *denote the values of* x *before and after any fixed round, respectively, and let* $S^+ = \{j : x_j^1 > x_j^0\}$, $S^- = \{j : x_j^1 < x_j^0\}$. *The potential increase in the round is lower bounded as:*

1. $\Phi(x^1) - \Phi(x^0) \geq \Omega(\beta\gamma)\sum_{j\in\{S^+\cup S^-\}} x_j^0 \sum_{i=1}^{m} y_i(x^0) A_{ij}$;
2. $\Phi(x^1) - \Phi(x^0) \geq \Omega\left(\frac{\beta}{(1-\beta)^\alpha}\right)\left(\sum_{j=1}^{n} x_j^0 \sum_{i=1}^{m} y_i(x^0) - (1+\gamma)\sum_{j=1}^{n} w_j (x_j^0)^{1-\alpha}\right)$;
3. $\Phi(x^1) - \Phi(x^0) \geq \Omega\left(\frac{\beta}{(1+\beta)^\alpha}\right)\left((1-\gamma)\sum_{j=1}^{n} w_j (x_j^0)^{1-\alpha} - \sum_{j=1}^{n} x_j^0 \sum_{i=1}^{m} y_i(x^0)\right)$.

Observe that for $\alpha > 1$ the objective function $p_\alpha(x)$, and, consequently, the potential function $\Phi(x)$, is negative for any feasible x. To yield a poly-logarithmic convergence time in R_w, m, n, and $A_{\max}$, the idea is to show that the negative potential $-\Phi(x)$ decreases by some

multiplicative factor whenever x is not a "good" approximation to x^* – the optimal solution to (P_α). This idea, combined with the fact that the potential never decreases (and therefore $-\Phi(x)$ never increases) and with upper and lower bounds on the potential then leads to the desired convergence time. Consider the following definition of a stationary round:

▶ **Definition 4.11** (Stationary round)**.** A round is stationary, if both:

1. $\sum_{j\in\{S^+\cup S^-\}} x_j^0 \sum_{i=1}^m y_i(x)A_{ij} < \gamma \sum_{j=1}^n w_j (x_j^0)^{1-\alpha}$, and
2. $(1-2\gamma)\sum_{j=1}^n w_j (x_j^0)^{1-\alpha} \le \sum_{j=1}^n x_j^0 \sum_{i=1}^m y_i(x^0)A_{ij}$,

where $S^+ = \{j : x_j^1 > x_j^0\}$, $S^- = \{j : x_j^1 < x_j^0\}$. Otherwise, the round is non-stationary.

Recall the expression for the negative potential: $-\Phi(x) = \frac{1}{\alpha-1}\sum_j w_j {x_j}^{1-\alpha} + \frac{1}{\kappa}\sum_i y_i(x)$. Then, using Lemma 4.10, it suffices to show that in a non-stationary round the decrease in the negative potential $-\Phi(x)$ is a multiplicative factor of the larger of the two terms $\frac{1}{\alpha-1}\sum_j w_j {x_j}^{1-\alpha}$ and $\frac{1}{\kappa}\sum_i y_i(x)$. The last part of the proof shows that the solution x that corresponds to any stationary round is close to the optimal solution. This part is done by appropriately upper-bounding the duality gap. Denoting by $S^+ \cup S^-$ the set of coordinates j for which x_j either increases or decreases in the observed stationary round and using Definition 4.11, we show that the terms $j \in \{S^+ \cup S^-\}$ contribute to the duality gap by no more than $O(\varepsilon\alpha)\cdot(-p_\alpha(x))$. The terms corresponding to $j \notin \{S^+ \cup S^-\}$ are bounded recalling (from α-FairPSolver) that for such terms $\frac{x_j^\alpha \sum_{i=1}^m y_i(x)A_{ij}}{w_j} \in (1-\gamma, 1+\gamma)$.

4.2 Structural Properties

Lower Bound on the Minimum Allocated Value. Recall (from Section 2) that the optimal solution x^* to (P_α) must lie in the positive orthant. We show in Lemma 4.12 that not only does x^* lie in the positive orthant, but the minimum element of x^* can be bounded below as a function of the problem parameters. This lemma motivates the choice of parameters δ_j in α-FairPSolver (Section 3). The proof is provided in [35].

▶ **Lemma 4.12.** *Let $x^* = (x_1^*, \ldots, x_n^*)$ be the optimal solution to (P_α). Then $\forall j \in \{1,\ldots,n\}$:*

- $x_j^* \ge \left(\frac{w_j}{w_{\max} M}\min_{i:A_{ij}\neq 0}\frac{1}{n_i A_{ij}}\right)^{1/\alpha}$, *if* $0 < \alpha \le 1$,
- $x_j^* \ge {A_{\max}}^{(1-\alpha)/\alpha}\left(\frac{w_j}{w_{\max} M}\right)^{1/\alpha}\min_{i:A_{ij}\neq 0}\frac{1}{n_i A_{ij}}$, *if* $\alpha > 1$,

where $n_i = \sum_{j=1}^n \mathbb{1}_{\{A_{ij}\neq 0\}}$[8] *is the number of non-zero elements in the i^{th} row of the constraint matrix A, and $M = \min\{m,n\}$.*

Asymptotics of α-Fair Allocations. The following lemma states that for sufficiently small (but not too small) α, the values of the linear and the α-fair objectives at their respective optimal solutions are approximately the same. This statement will then lead to a conclusion that to $\varepsilon-$approximately solve an α-fair packing problem for a very small α, one can always use an $\varepsilon-$approximation packing LP algorithm.

▶ **Lemma 4.13.** *Let (P_α) be an α-fair packing problem with optimal solution x^*, and (P_0) be the LP with the same constraints and the same weights w as (P_α) and an optimal solution z^*. Then if $\alpha \le \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, we have that $\sum_j w_j z_j^* \ge (1-3\varepsilon)\sum_j \frac{(x_j^*)^{1-\alpha}}{1-\alpha}$, where $\varepsilon \in (0, 1/6]$.*

[8] With the abuse of notation, $\mathbb{1}_{\{e\}}$ is the indicator function of the expression e, i.e., 1 if e holds, and 0 otherwise.

Observing that for any $\alpha \in (0,1)$, $\frac{(z_j^*)^{1-\alpha}}{1-\alpha} \geq z_j^*$ (since, due to the scaling, $z_j^* \in [0,1]$), a simple corollary of Lemma 4.13 is that an $\varepsilon-$approximation z to (P_0) ($\sum_j w_j z_j \geq (1-\varepsilon)\sum_j w_j z_j^*$) is also an $O(\varepsilon)-$approximation to (P_α), for $\alpha \leq \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$. Thus, to find an $\varepsilon-$approximate solution for $\alpha \leq \frac{\varepsilon/4}{\ln(nA_{\max}/\varepsilon)}$, the packing LP algorithm of [7] can be run, which means that there is a stateless distributed algorithm that converges in $\text{poly}(\ln(\varepsilon^{-1}R_w mnA_{\max})/\varepsilon)$ time for α arbitrarily close to zero.

The following two lemmas show that when α is sufficiently close to 1, (P_α) can be $\varepsilon-$approximated by $\varepsilon-$approximately solving (P_1) with the same constraints and weights.

▶ **Lemma 4.14.** *Let x be an $\varepsilon-$approximate solution to a 1-fair packing problem (P_1) returned by α-FairPSolver. Then, for any $\alpha \in [1-1/\tau_0, 1)$, where $\tau_0 = \frac{1}{\beta}\ln(\frac{1}{\delta_{\min}})$, x is also a $2\varepsilon-$approximate solution to (P_α), where the only difference between (P_1) and (P_α) is in the value of α in the objective.*

▶ **Lemma 4.15.** *Let x be an $\varepsilon-$approximate solution to a 1-fair packing problem (P_1) returned by α-FairPSolver. Then, for any $\alpha \in (1, 1+1/\tau_0]$, where $\tau_0 = \frac{1}{\beta}\ln(\frac{1}{\delta_{\min}})$, x is also a $2\varepsilon-$approximate solution to (P_α), where the only difference between (P_1) and (P_α) is in the value of α in the objective.*

Finally, we consider the asymptotics of α-fair allocations, as α becomes large. This result complements the result from [38] that states that α-fair allocations approach the max-min fair one as $\alpha \to \infty$ by showing how fast the max-min fair allocation is reached as a function of α, R_w, n, and $A_{\max}$. First, for completeness, we provide the definition of max-min fairness.

▶ **Definition 4.16.** (Max-min fairness [10].) Let $\mathcal{R} \subset \mathbb{R}_+^n$ be a compact and convex set. A vector $x \in \mathcal{R}$ is max-min fair on $\mathcal{R}$ if for any vector $z \in \mathcal{R}$ it holds that: if for some $j \in \{1,\ldots,n\}$ $z_j > x_j$, then there exists $k \in \{1,\ldots,n\}$ such that $z_k < x_k$ and $x_k \leq x_j$.

On a compact and convex set $\mathcal{R} \subset \mathbb{R}^n$, the max-min fair vector is unique [44, 43]. The following lemma shows that for $\alpha \geq \varepsilon^{-1}\ln(R_w nA_{\max})$, the α-fair vector and the max-min fair vector are $\varepsilon-$close to each other. Notice that because of a very large gradient of $p_\alpha(x)$ as α becomes large, the max-min fair solution gives only an $O(\varepsilon\alpha)-$approximation to (P_α).

▶ **Lemma 4.17.** *Let x^* be the optimal solution to $(P_\alpha) = \max\{p_\alpha(x) : Ax \leq 1, x \geq 0\}$, z^* be the max-min fair solution for the convex and compact set determined by the constraints from (P_α). Then if $\alpha \geq \varepsilon^{-1}\ln(R_w nA_{\max})$, we have that:*
1. *$p_\alpha(x^*) \leq (1-\varepsilon(\alpha-1))p_\alpha(z^*)$, i.e., z^* is an $\varepsilon(\alpha-1)-$approximate solution to (P_α), and*
2. *$(1-\varepsilon)z_j^* \leq x_j^* \leq (1+\varepsilon)z_j^*$, for all $j \in \{1,\ldots,n\}$.*

5 Conclusion

We presented an efficient stateless distributed algorithm for the class of α-fair packing problems. To the best of our knowledge, this is the first algorithm with poly-logarithmic convergence time in the input size. Additionally, we obtained results that characterize the fairness and asymptotic behavior of allocations in weighted α-fair packing problems that may be of independent interest. An interesting open problem is to determine the class of objective functions for which the presented techniques yield fast and stateless distributed algorithms, together with a unified convergence analysis. This problem is especially important in light of the fact that α-fair objectives are not Lipschitz continuous, do not have a Lipschitz gradient, and their dual gradient's Lipschitz constant scales at least linearly with n and $A_{\max}$.

Therefore, the properties typically used in fast first-order methods are lacking [40, 2]. Finally, for applications of α-fair packing that do not require stateless updates, it seems plausible that the dependence on ε^{-1} in the convergence bound can be improved from ε^{-5} to ε^{-3} by relaxing the requirement for asynchronous updates, similarly as was done in [3] over [7].

References

1 Zeyuan Allen-Zhu and Lorenzo Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *Proc. ACM STOC'15*, 2015.

2 Zeyuan Allen-Zhu and Lorenzo Orecchia. A novel, simple interpretation of Nesterov's accelerated method as a combination of gradient and mirror descent, Jan. 2015. arXiv preprint, `http://arxiv.org/abs/1407.1537`.

3 Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proc. ACM-SIAM SODA'15*, 2015.

4 Anthony B Atkinson. On the measurement of inequality. *J. Econ. Theory*, 2(3):244–263, 1970.

5 Baruch Awerbuch, Yossi Azar, and Rohit Khandekar. Fast load balancing via bounded best response. In *Proc. ACM-SIAM SODA'08*, 2008.

6 Baruch Awerbuch and Rohit Khandekar. Greedy distributed optimization of multi-commodity flows. In *Proc. ACM PODC'07*, 2007.

7 Baruch Awerbuch and Rohit Khandekar. Stateless distributed gradient descent for positive linear programs. *SIAM J. Comput.*, 38(6):2468–2486, 2009.

8 Yair Bartal, John Byers, and Danny Raz. Global optimization using local information with applications to flow control. In *Proc. IEEE FOCS'97*, 1997.

9 Amir Beck, Angelia Nedić, Asuman Ozdaglar, and Marc Teboulle. An $O(1/k)$ gradient method for network resource allocation problems. *IEEE Trans. Control Netw. Syst.*, 1(1):64–73, 2014.

10 Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall, 1992.

11 Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. On the efficiency-fairness trade-off. *Manag. Sci*, 58(12):2234–2250, 2012.

12 Thomas Bonald and James Roberts. Multi-resource fairness: Objectives, algorithms and performance. In *Proc. ACM SIGMETRICS'15*, 2015.

13 Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2009.

14 Anna Charny, David D Clark, and Raj Jain. Congestion control with explicit rate indication. In *Proc. IEEE ICC'95*, 1995.

15 Yun Kuen Cheung, Richard Cole, and Nikhil Devanur. Tatonnement beyond gross substitutes?: Gradient descent to the rescue. In *Proc. ACM STOC'13*, 2013.

16 Shlomi Dolev. *Self-stabilization*. MIT press, 2000.

17 Lisa Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM J. Discrete Math.*, 13(4):505–520, 2000.

18 Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007.

19 Naveen Garg and Neal Young. On-line end-to-end congestion control. In *Proc. IEEE FOCS'02*, 2002.

20 Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proc. USENIX NSDI'11*, 2011.

21 Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In *Proc. ACM STOC'14*, 2014.

22 Jeffrey Jaffe. Bottleneck flow control. *IEEE Trans. Commun.*, 29(7):954–962, 1981.

23 Kamal Jain and Vijay Vazirani. Eisenberg-gale markets: Algorithms and structural properties. In *Proc. ACM STOC'07*, 2007.

24 Carlee Joe-Wong, Soumya Sen, Tian Lan, and Mung Chiang. Multiresource allocation: Fairness-efficiency tradeoffs in a unifying framework. *IEEE/ACM Trans. Netw.*, 21(6):1785–1798, 2013.

25 Frank Kelly, Aman Maulloo, and David Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *J. Oper. Res. Soc.*, 49(3):237–252, 1998.

26 Frank Kelly and Elena Yudovina. *Stochastic networks*, volume 2. Cambridge University Press, 2014.

27 Jon Kleinberg, Yuval Rabani, and Éva Tardos. Fairness in routing and load balancing. In *Proc. IEEE FOCS'99*, 1999.

28 Christos Koufogiannakis and Neal Young. Beating simplex for fractional packing and covering linear programs. In *Proc. IEEE FOCS'07*, 2007.

29 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. The price of being near-sighted. In *Proc. ACM-SIAM SODA'06*, 2006.

30 Amit Kumar and Jon Kleinberg. Fairness measures for resource allocation. In *Proc. IEEE FOCS'00*, 2000.

31 Tian Lan, David Kao, Mung Chiang, and Ashutosh Sabharwal. An axiomatic theory of fairness in network resource allocation. In *Proc. IEEE INFOCOM'10*, 2010.

32 Steven Low, Fernando Paganini, and John Doyle. Internet congestion control. *IEEE Control Systems*, 22(1):28–43, 2002.

33 Michael Luby and Noam Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. ACM STOC'93*, 1993.

34 Jelena Marašević, Clifford Stein, and Gil Zussman. Max-min fair rate allocation and routing in energy harvesting networks: Algorithmic analysis. In *Proc. ACM MobiHoc'14*, 2014.

35 Jelena Marašević, Cliff Stein, and Gil Zussman. A fast distributed stateless algorithm for α-fair packing problems, Feb. 2016. arXiv preprint, `http://arxiv.org/abs/1502.03372v3`.

36 Bill McCormick, Frank Kelly, Patrice Plante, Paul Gunning, and Peter Ashwood-Smith. Real time alpha-fairness based traffic engineering. In *Proc. ACM HotSDN'14*, 2014.

37 Nimrod Megiddo. Optimal flows in networks with multiple sources and sinks. *Math. Program.*, 7(1):97–107, 1974.

38 Jeonghoon Mo and Jean Walrand. Fair end-to-end window-based congestion control. *IEEE/ACM Trans. Netw.*, 8(5):556–567, October 2000.

39 Damon Mosk-Aoyama, Tim Roughgarden, and Devavrat Shah. Fully distributed algorithms for convex optimization problems. In *Proc. DISC'07*, 2007.

40 Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

41 Christos Papadimitriou and Mihalis Yannakakis. Linear programming without the matrix. In *Proc. ACM STOC'93*, 1993.

42 Serge Plotkin, David Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Math. Oper. Res.*, 20(2):257–301, 1995.

43 Božidar Radunović and Jean-Yves Le Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Trans. Netw.*, 15(5):1073–1083, 2007.

44 Saswati Sarkar and Leandros Tassiulas. Fair allocation of discrete bandwidth layers in multicast networks. In *Proc. IEEE INFOCOM'00*, 2000.

45 Yung Yi and Mung Chiang. Stochastic network utility maximisation – a tribute to Kelly's paper published in this journal a decade ago. *Eur. Trans. Telecommun.*, 19(4):421–442, 2008.

46 Neal Young. Sequential and parallel algorithms for mixed packing and covering. In *Proc. IEEE FOCS'01*, 2001.

All-Pairs Approximate Shortest Paths and Distance Oracle Preprocessing

Christian Sommer

Apple Inc., Cupertino, CA, USA
csommer@apple.com

Abstract

Given an undirected, unweighted graph G on n nodes, there is an $O(n^2 \text{poly} \log n)$-time algorithm that computes a data structure called *distance oracle* of size $O(n^{5/3} \text{poly} \log n)$ answering approximate distance queries in constant time. For nodes at distance d the distance estimate is between d and $2d + 1$.

This new distance oracle improves upon the oracles of Pătraşcu and Roditty (FOCS 2010), Abraham and Gavoille (DISC 2011), and Agarwal and Brighten Godfrey (PODC 2013) in terms of preprocessing time, and upon the oracle of Baswana and Sen (SODA 2004) in terms of stretch. The running time analysis is tight (up to logarithmic factors) due to a recent lower bound of Abboud and Bodwin (STOC 2016).

Techniques include dominating sets, sampling, balls, and spanners, and the main contribution lies in the way these techniques are combined. Perhaps the most interesting aspect from a technical point of view is the application of a spanner without incurring its constant additive stretch penalty.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases graph algorithms, data structures, approximate shortest paths, distance oracles, distance labels

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.55

1 Introduction

Given a graph $G = (V, E)$ on $n := |V|$ vertices, the *All-Pairs Shortest Path (APSP)* problem asks for pairwise distances $d(u, v)$ among all pairs of nodes $u, v \in V$. The fastest known algorithms to compute these $\binom{n}{2}$ distance values (in this paper, we restrict ourselves to undirected, unweighted graphs) use fast matrix multiplication [16, 18, 29, 34] or run in roughly cubic time [12, 20, 33, 37]. Depending on the number of edges $m := |E|$, computing n independent shortest-path trees (breadth-first search trees for unweighted graphs) in $O(mn)$ may be faster.

Algorithms for All-Pairs *Approximate* Shortest Paths (*APASP*) trade accuracy for speed. Their worst-case approximation guarantee is called *stretch*. Stretch (α, β) means that for any pair of nodes $u, v \in V$ at distance d the algorithm's estimate is between d and $\alpha \cdot d + \beta$ (typically, when $\beta > 0$, graphs are assumed to be unweighted or β depends on the largest edge weight).

Aingworth, Chekuri, Indyk, and Motwani [5] introduced a technique to handle high-degree nodes using *dominating sets*. They apply their technique to derive a $(1, 2)$-stretch algorithm that runs in time $\tilde{O}(n^{5/2})$ (as usual, $\tilde{O}(\cdot)$ hides logarithmic factors). Their dominating-set technique has been used in many subsequent algorithms. Dor, Halperin, and Zwick [17] improved the running time to $\tilde{O}(\min\{n^{3/2}m^{1/2}, n^{7/3}\})$. Furthermore, they also gave a reduction, proving that an APASP algorithm with multiplicative stretch strictly less than 2

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 55; pp. 55:1–55:13

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

Table 1 A selection of related work on the all-pairs approximate shortest path problem and constant-time approximate distance oracles for unweighted graphs. Some results hold for weighted graphs as well, and some methods guarantee better bounds for sparse graphs. To simplify the comparison for dense graphs, the bounds in this table are for $m = n^2$ (except for the sub-quadratic algorithm of [6]). Stretch (α, β) means that for pairs at distance d the distance estimate is between d and $\alpha d + \beta$.

Stretch	Time $\tilde{O}(\cdot)$	Space $\tilde{O}(\cdot)$	Reference
(1,2)	$n^{5/2}$	n^2	[5]
(1,2)	$n^{7/3}$	n^2	[17]
(3,0)	n^2	n^2	[15]
(3,0)	$n^{5/2}$	$n^{3/2}$	[31]
(3,0)	n^2	$n^{3/2}$	[8, 9]
(3,10)	$n^{23/12} + m$	$n^{3/2}$	[6]
(2,1)	n^2	n^2	[10]
(2,1)	$n^{8/3}$	$n^{5/3}$	[7] (space implicit)
(2,3)	n^2	$n^{5/3}$	[7] (space implicit)
(2,1)	poly	$n^{5/3}$	[21]
(2,1)	n^2	$n^{5/3}$	Theorem 1

or constant additive stretch serves as an algorithm for boolean matrix multiplication. After this reduction, most research efforts have been focused on stretch 2 or higher. Cohen and Zwick [15] provided various tradeoffs such as an $\tilde{O}(n^2)$ algorithm with stretch $\alpha = 3$. Berman and Kasiviswanathan [10] further improved the stretch to $(2, 1)$. See also the survey by Zwick [38].

A *distance oracle* is a *compact* representation of the AP(A)SP matrix of a graph $G = (V, E)$. The main quantities of interest are the *construction time* (also called *preprocessing time*), the *space consumption*, the *query time*, and the *stretch*. Thorup and Zwick [31] coined the term *distance oracle*, and they also provided an oracle with constant query time, multiplicative stretch $\alpha = 3$, space $O(n^{3/2})$, and preprocessing time $\tilde{O}(mn^{1/2})$, as well as more general tradeoffs for all odd integers $\alpha > 3$, which are not discussed any further in this paper.[1] For $\alpha = 3$, their space bound is asymptotically optimal due to the existence of dense graphs with large girth [11, 24, 32]. The preprocessing time was subsequently improved in a line of work: Baswana and Sen [9] improved the preprocessing time to $O(n^2)$ for unweighted graphs, Baswana and Kavitha [8] obtained $O(n^2)$ preprocessing time for weighted graphs, and Baswana, Gaur, Sen, and Upadhyay [6] gave a sub-quadratic preprocessing algorithm for stretch $(3, 10)$. See also the survey by Sen [26].[2]

Is stretch $\alpha = 3$ the best we can expect of a distance oracle with sub-quadratic space? Pătraşcu and Roditty [21][3] provide an oracle with constant query time, stretch $(2, 1)$, space $O(n^{5/3})$, and polynomial preprocessing time. The tradeoff between stretch and space is asymptotically optimal assuming the hardness of set intersection [21, 22]. Less well known, the work of Baswana, Goyal, and Sen [7] contains a data structure that is remarkably similar

[1] For larger stretch values, Wulff-Nilsen [36] provides the fastest preprocessing times and Chechik [13, 14] provides the fastest query times; for a survey, see [27].

[2] Again for larger stretch values, Wulff-Nilsen [36] further improved the preprocessing times.

[3] See also Abraham and Gavoille [2] for a distributed distance oracle, also known as a *distance labeling scheme*, as well as Agarwal and Brighten Godfrey [4].

to the distance oracle in the result of Pătraşcu and Roditty [21], implicitly providing an $\tilde{O}(mn^{2/3})$ bound on the preprocessing time for $(2,1)$-approximate distance oracles.[4] They also provide an $\tilde{O}(n^2)$-time algorithm for a data structure with stretch $(2,3)$ instead of the optimal $(2,1)$. Again, the space bound is not stated explicitly as a distance oracle result. For an overview, see Table 1.

What is the fastest preprocessing time that can be achieved for distance oracles with multiplicative stretch $\alpha < 3$? We propose an algorithm that runs in quadratic time (up to polylogarithmic factors) and computes a $(2,1)$-approximate distance oracle. Analogous to the distance oracles by Abraham and Gavoille [2], the distance oracle can also be distributed as *distance labels*.

▶ **Theorem 1.** *Given an undirected, unweighted graph G on n nodes, there is an $\tilde{O}(n^2)$-time algorithm that computes a $(2,1)$-approximate distance oracle of size $\tilde{O}(n^{5/3})$ with query time $O(1)$.*

Previously known quadratic-time algorithms either produce data structures with quadratic space ([10, 15, 17], see also Section 2.4) or with worst-case stretch $(2,3)$ (see [7]). Probably the most interesting technical aspect of the result is the use of an additive spanner *free of charge*, i.e., without incurring its typical stretch penalty (the query time has a linear dependency on the additive stretch of the spanner). The argument works for any spanner with constant additive stretch, and hence is tight due to a recent lower bound by Abboud and Bodwin [1].

For general graphs, Theorem 1 seems hard to improve upon (modulo logarithmic factors) for two reasons (beyond the obvious reason that the algorithm needs to read the graph with potentially $\Omega(n^2)$ edges): the space-stretch tradeoff is essentially tight due to a set intersection lower bound [21, 22], and the preprocessing-stretch-query tradeoff is essentially tight as for distance oracles with quadratic preprocessing time and $\tilde{O}(1)$ query time (but independent of the space), any improvement of the multiplicative stretch below 2 would imply a (quasi-)quadratic algorithm for boolean matrix multiplication [3, 17] (the fastest known combinatorial algorithm runs in roughly cubic time [37]).

For sparse graphs, I am not aware of any arguments against even faster preprocessing time $\tilde{O}(m + n^{5/3})$ (and no additive $+1$ stretch if $m = \tilde{O}(n)$ [21]). As mentioned above, Baswana, Goyal, and Sen [7] implicitly prove an $\tilde{O}(mn^{2/3})$ bound on the preprocessing time (Theorem 5), which for $m = \tilde{O}(n)$ is asymptotically optimal. The optimal preprocessing time for m between $\tilde{\omega}(n)$ and $\tilde{o}(n^2)$ remains open.

The distance oracle of Theorem 1 roughly works as follows (the description omits $\tilde{O}$-notation and assumes familiarity with the techniques described in the preliminaries (Section 2)): short distances are exact by storing balls and their intersection (as in [2, 7, 21]); long distances are triangulations via *landmarks*. However, we cannot afford to compute exact distances for each node and all $n^{2/3}$ landmarks. Instead, we group landmarks by degree (using dominating sets), where paths from/to the 2^i landmarks with degree $n/2^i$ are computed in graphs with $n^2/2^i$ edges plus the edges of an $(1,6)$-spanner [35]. The spanner ensures that distances do not deteriorate too much. For each node v, we compute a set of *portals* by sparsifying its set of (up to $\log n$) nearest landmarks to include a landmark with degree $n/2^j$ if and only if it is closer to v than all landmarks with higher degree $n/2^i$ (i.e., for all $i < j$). At query time, a constant number of landmarks per node is sufficient to guarantee

[4] An important reference I unfortunately failed to include when writing [27].

stretch $(2, 1)$: the query algorithm triangulates via landmarks by increasing degree, and each subsequent landmark implies that the target distance increases by at least 1.

The overall framework is described in Section 3. The main part of the argument is formalized as a distance oracle for *heavy paths*, where a path is defined to be heavy if it passes through two consecutive high-degree nodes, see Section 4.

2 Preliminaries

2.1 Spanners

Spanners [23] are relatively sparse subgraphs with additional properties such as stretch bounds on shortest-path distances. A spanner is deemed to have *stretch* (α, β) if $d_H(u, v) \leqslant \alpha \cdot d_G(u, v) + \beta$. Note that, since H is a subgraph, $d_G(u, v) \leqslant d_H(u, v)$. While there are no known distance oracles or distance labeling schemes with constant additive stretch (and are unlikely to exist [19, 21, 28]), such spanners can be computed efficiently. The preprocessing algorithm in this paper invokes the following spanner construction as a subroutine.

▶ **Theorem 2** (Woodruff [35]). *Given an undirected, unweighted graph G on n nodes, there is an $\tilde{O}(n^2)$-time algorithm that computes a $(1, 6)$-spanner with $\tilde{O}(n^{4/3})$ edges.*

Recently, Abboud and Bodwin [1] proved that Theorem 2 is tight in a very strong way. Any spanner with constant additive stretch must have essentially $n^{4/3}$ edges. More precisely, they prove the following.

▶ **Theorem 3** (Abboud and Bodwin [1]). *For all $\epsilon > 0$ there exists a $\delta > 0$ and an infinite family of graphs $G = (V, E)$ such that for any subgraph $H = (V, E')$ with $|E'| = O(n^{4/3-\epsilon})$, there exist nodes $u, v \in V$ with $d_H(u, v) = d_G(u, v) + \Omega(n^{\delta})$.*

Their result illustrates the limitations of the construction in this paper. There is no better spanner than Woodruff's for our purposes (as long as we are indifferent to log factors in the $\tilde{O}(\cdot)$ notation).

2.2 Balls and Clusters

While long distances can be approximated well using triangulation via *landmarks*, many distance oracles handle short-range distances using *balls* and *clusters* [2, 21, 22, 31].

▶ **Definition 4** (Balls and Clusters). Given a graph $G = (V, E)$ and a set of landmarks $L \subseteq V$, the *ball* of a node v with respect to L is $B_L(v) := \{u : d(v, u) < d(v, L)\}$. The *cluster* of a node u with respect to L is $C_L(u) := \{v : d(v, u) < d(v, L)\}$.

Note that $v \in C_L(u)$ if and only if $u \in B_L(v)$.

For any $0 < \ell < n$, random sampling with probability ℓ/n yields ℓ landmarks (in expectation) and *average* ball and cluster sizes of $O(n/\ell)$. A sampling algorithm of Thorup and Zwick [30] computes a set of landmarks L such that no node cluster $C_L(u)$ is larger than $O(n/\ell)$ and $|L| = O(\ell \log n)$. The running time of their algorithm is bounded by $O((mn \log n)/\ell)$. Abraham and Gavoille [2] extend the Thorup-Zwick sampling algorithm to also guarantee worst-case bounds on balls $|B_L(v)| = O(n/\ell)$.

Roditty, Thorup, and Zwick [25] provide a deterministic algorithm (based on hitting sets) to select landmarks. They also claim (without proof due to lack of space) that their algorithm can be used to de-randomize the quadratic-time preprocessing algorithm of Baswana and Sen [9].

2.3 Stretch-2 Distance Oracles

In the following, we give a brief description of the distance oracles by [2, 4, 21] for unweighted graphs. Similar arguments were also used in the analysis for all-pairs approximate shortest path algorithms [7].

The preprocessing algorithm starts by selecting a set of landmarks L of size roughly $|L| = \tilde{O}(n^{2/3})$ such that $|B_L(u)|, |C_L(u)| = \tilde{O}(n^{1/3})$ for all $u \in V$ (see sampling algorithm above). Long-range distances (global queries) are answered by triangulation using the landmark closest to either of the query nodes. To prepare for those queries, the preprocessing algorithm computes distances for all pairs of nodes in $V \times L$. Short-range distances (local queries) are answered using *super balls* $S_L(\cdot)$. For each node u, the algorithm computes the distance to all nodes in its ball $B_L(u)$ as well as all nodes v whose balls $B_L(v)$ intersect with $B_L(u)$, i.e., $S_L(u) := B_L(u) \cup \bigcup_{w \in B_L(u)} C_L(w)$. Since the sizes of both $B_L(\cdot)$ and $C_L(\cdot)$ are bounded by $\tilde{O}(n^{1/3})$, super balls contain at most $|S_L(u)| = \tilde{O}(n^{2/3})$ nodes.

The query algorithm for source s and target t checks whether $t \in S_L(s)$ or $s \in S_L(t)$. If so, the exact distance has been pre-computed (a local query). Otherwise, let l_s and l_t denote the landmarks closest to s and t, respectively. The long-range distance $\min\{d(s, l_s) + d(l_s, t), d(s, l_t) + d(l_t, t)\}$ is returned.

For the stretch analysis, we need to bound the distance returned for global queries. Without loss of generality, let us assume $d(t, l_t) \leqslant d(s, l_s)$. Since $B_L(s)$ and $B_L(t)$ do not intersect, $d(s, t) + 1 \geqslant d(s, l_s) + d(t, l_t) \geqslant 2d(t, l_t)$. By the triangle inequality, $d(s, l_t) \leqslant d(s, t) + d(t, l_t)$, hence the distance estimate is at most $d(s, t) + 2d(t, l_t)$, which is bounded by $2d(s, t) + 1$.

Baswana, Goyal, and Sen [7, Theorem 5.1] provide a randomized preprocessing algorithm with expected running time $O(m^{2/3} n \log n + n^2)$. More generally, they prove the following.

▶ **Theorem 5** (Baswana, Goyal, and Sen [7]). *For any $p \in (0, 1)$, an undirected unweighted graph $G = (V, E)$ can be preprocessed in expected $O(m \log n + n^2 + (n/p^2) \log n + mnp \log n)$ time to build a data structure that can report $(2, 1)$-approximate distances in constant time.*

Instead of super balls, a so-called *overlap matrix* is pre-computed to decide whether two balls intersect $B_L(u) \cap B_L(v) \stackrel{?}{=} \emptyset$. The landmark set L is computed using the sampling algorithm of Thorup and Zwick [30] (see also Section 2.2) to ensure that balls and their intersections are not too large.

2.4 High-Degree Dominating Sets

Recall that a dominating set of a graph G is a subset of nodes $D \subseteq V(G)$ such that each node is in D or has a neighbor in D. Aingworth, Chekuri, Indyk, and Motwani [5] prove that for a set of high-degree nodes there is an algorithm that can find a small dominating set (the algorithm is a greedy algorithm with logarithmic approximation ratio).

▶ **Theorem 6** (Aingworth et al. [5]). *Let $G = (V, E)$ be an undirected graph with $n := |V|$ and $m := |E|$, and let $V_\delta := \{v \in V : \deg(v) \geqslant \delta\}$. There is an algorithm that finds a dominating set for V_δ with size $O((n \log n)/\delta)$ in time $O(m + n\delta)$.*

Building on this theorem, they derive an $\tilde{O}(n^{5/2})$-time algorithm computing all-pairs approximate shortest paths with stretch $(1, 2)$. The dominating-set technique has been used extensively in algorithms for APASP (and related algorithms). Dor, Halperin, and Zwick [17, Theorem 6.2] compute an $(1, 2)$-approximate *spanner* in time $\tilde{O}(n^2)$ as follows (Algorithm 1): for decreasing node degrees $(n/2^i)$, split the nodes into high-degree and low-degree nodes,

compute a dominating set for the high-degree nodes (as in Theorem 6), compute a BFS tree from each node in the dominating set in the subgraph with edges adjacent to at least one low-degree node, and return the union of all these BFS trees.

1: **for** $i \in \{0, 1, 2, 3, \ldots, \lceil \log_2 \sqrt{n} \rceil\}$ **do**
2: $\quad \delta_i = n/2^i$
3: $\quad V_i = \{v \in V : \deg(v) \geqslant \delta_i\}$, $E_i = \{uv \in E : \deg(u) < 2\delta_i \text{ or } \deg(v) < 2\delta_i\}$
4: $\quad D_i = \mathsf{dominate}(V_i)$ (as in Theorem 6)
5: $\quad$ **for** $p \in D_i$ **do**
6: $\quad\quad T_i(p) = \mathsf{bfs}_{(V,E_i)}(p)$ (where T_i denotes the edges of the breadth-first tree)
7: $\quad$ **end for**
8: **end for**
9: **return** $\bigcup_{i,l \in D_i} T_i(p)$

Algorithm 1: Spanner construction by Dor, Halperin, and Zwick [17, Figure 7].

Cohen and Zwick [15] and Baswana and Kavitha [8] also provide algorithms similar to Algorithm 1 with different parameters and analysis.

Yet another instantiation of the above algorithm and framework is due to Berman and Kasiviswanathan [10]. Their all-pairs approximate shortest path algorithm computes $(2, 1)$-approximate distances in time $O(n^2 \log^2 n)$. Their main loop is essentially as in Algorithm 1, Lines 1–8 for i up to $\lceil \log_2 n \rceil$. Each node v remembers its nearest landmark $l_i(v)$ (for each level i, and with arbitrary tie breaking). Distance estimates $\tilde{d}(s, t)$ are computed as the minimum over all i of the triangulation via the landmark of s or via the landmark of t. We apply and extend their argument in Section 4.2.2.

3 Proof of Theorem 1

We split the problem into a dense case and a sparse case, where sparse means $O(n^{4/3})$ edges. The overall distance oracle simply returns the minimum from both data structures.

As in most shortest-path algorithms using dominating sets (see Section 2.4), edges uv are classified by minimum degree of their adjacent vertices u and v:

▶ **Definition 7** (Edge Degree)**.** The *degree* of an edge $uv \in E(G)$, denoted by $\deg(uv)$, is defined as $\deg(uv) := \min_{u,v} \{\deg(u), \deg(v)\}$.

Let G_δ denote the subgraph of G induced by all edges uv for which $\deg(uv) \leqslant \delta$. Note that $|E(G_\delta)| = O(n \cdot \delta)$, since each edge contributes to the degree of two nodes. As degree threshold in our proof, let Δ be the smallest power of 2 greater than $n^{1/3}$.

For the sparse case, we run the $O(n^2 + m^{2/3} n \log n)$-time algorithm of Baswana, Goyal, and Sen [7] (Theorem 5) on G_Δ. For $m = O(n^{4/3})$, the running time is $O(n^2)$. As mentioned in the introduction, their data structure is similar to the distance oracles by Pătraşcu and Roditty [21] and Abraham and Gavoille [2]. See Section 2.3 for more details.

For the dense case, we compute a data structure for G called *Heavy Path Oracle*. It is essentially an approximate distance oracle but limited in the following way: it can only return *heavy* paths.

▶ **Definition 8** (Heavy Paths)**.** A path is called δ-*heavy* if and only if it contains an edge uv with degree $\deg(uv) \geqslant \delta$.

Quite naturally, and analogously to a shortest path, there is a shortest δ-heavy path between any pair of nodes $s, t \in V(G)$. Let the *shortest δ-heavy distance* between s and t be the length of a shortest δ-heavy path between s and t.

The distance oracle for the dense case handles those pairs correctly for which any shortest path is Δ-heavy. If no shortest path between s and t is Δ-heavy, any shortest path must be in G_Δ, and hence the distance oracle for the sparse case provides an accurate estimate.

The remainder of this paper is devoted to the proof of the following lemma, which concludes the proof of Theorem 1.

▶ **Lemma 9** (Heavy Path Oracle). *Given an undirected, unweighted graph G on n nodes, let Δ denote the smallest power of 2 greater than $n^{1/3}$. There is an $\tilde{O}(n^2)$-time algorithm that computes a data structure of size $\tilde{O}(n^{5/3})$ with query time $O(1)$ that returns $(2,1)$-approximations for shortest Δ-heavy distances.*

4 Heavy Path Oracle

We prove Lemma 9 by adapting the APASP algorithms of Dor, Halperin, and Zwick [17] and Berman and Kasiviswanathan [10].

As typical in distance oracles, each node $v \in V$ stores distances to a set of *landmarks*, chosen as the union of hierarchical dominating sets (details below). Furthermore, each node stores a constant-size subset of nearby landmarks called *portals*. At query time, triangulation happens via portals.

4.1 Preprocessing Algorithm

For a pseudo-code description, see Algorithm 2. Given $G = (V, E)$, the algorithm begins by computing a sparse $(1, 6)$-spanner $H = (V, S)$ as in Theorem 2. Recall that G_δ is defined to be the subgraph of G induced by all edges uv for which $\deg(uv) \leqslant \delta$. In the remainder of the preprocessing algorithm, each breadth-first search in a subgraph $G_\delta = (V, E_\delta)$ of $G = (V, E)$ shall also consider the edges of the spanner, i.e., the breadth-first search is executed in $G_\delta + H = (V, E_\delta \cup S)$.

Independent of the spanner, the edge set E is organized into $\lceil \log n^{2/3} \rceil$ hierarchical classes: for $\delta_i = n/2^i$, let $V_i := \{v \in V : \deg(v) \geqslant \delta_i\}$. Class E_i consists of all edges uv for which u or v has degree $< 2\delta_i$. The algorithm then finds a dominating set for V_i, denoted by L_i (the union of all L_i is the *landmark set*). For each node $\ell \in L_i$, the algorithm runs a breadth-first search in $(V, E_i \cup S)$, computing and storing distances $d_{(V,E_i \cup S)}(\ell, \cdot)$. Each node v remembers its nearest landmark as a *portal* $p_i(v)$ if and only if it is closer than all portals $p_j(v)$ on previous levels $j < i$. This selection of portals is essential for the query time and stretch analysis. Let $K \geqslant 12$ (twice the additive stretch of the spanner). For each node v, let $P(v)$ denote the set that contains the $K + 1$ portals closest to v.

4.1.1 Space and Running Time Analysis

We have $O(\log n)$ distance tables $L_i \times V$. The dominating sets are small $|L_i| \leqslant \tilde{O}(n^{2/3})$, hence the distance tables can be kept (and stored in a distributed way at the non-landmark node to also enable distance labels).

For each level i, each node v stores distances to all nodes of the dominating set L_i. This dominating set L_i has size $O(2^i \log n)$ due to Theorem 6. The last level has the largest dominating set of size $O(n^{2/3} \log n)$. The overall space consumption is thus bounded by $O(n^{5/3} \log^2 n)$.

.

```
1: (V, S) = Spanner(V, E) (as in Theorem 2)
2: for i ∈ {0, 1, 2, 3, ..., ⌈log_2 n^{2/3}⌉} do
3:    δ_i = n/2^i
4:    V_i = {v ∈ V : deg(v) ⩾ δ_i}
5:    L_i = DominatingSet(V_i) (as in Theorem 6)
6:    E_i = {uv ∈ E : deg(u) < 2δ_i or deg(v) < 2δ_i}
7:    for ℓ ∈ L_i do
8:       BreadthFirstSearch_{(V,E_i∪S)}(ℓ)
9:       for v ∈ V do
10:         store distance d_{(V,E_i∪S)}(v, ℓ) to landmark
11:         if d_{(V,E_i∪S)}(v, ℓ) < d_{(V,E_i∪S)}(v, ℓ_i(v)) then
12:            ℓ_i(v) = ℓ (remember nearest landmark per level)
13:         end if
14:      end for
15:   end for
16: end for
    Portal Selection
17: for v ∈ V do
18:   for i ∈ {0, 1, 2, 3, ..., ⌈log_2 n^{2/3}⌉} do
19:      if d_{(V,E_i∪S)}(v, ℓ_i(v)) < d_{(V,E_j∪S)}(v, ℓ_j(v)) for all j < i then
20:         store ℓ_i(v) as a candidate portal
21:      end if
22:   end for
23:   let P(v) be the first K + 1 portals closest to v
24: end for
```

Algorithm 2: Preprocessing of graph $G = (V, E)$

For each level i, the dominating set is computed in time at most $O(n^2)$ by Theorem 6. For each $\ell \in L_i$ the algorithm computes a breadth-first search in a graph with at most $|E_i| = O(n^2/2^i)$ plus $|S| = \tilde{O}(n^{4/3})$ edges (the latter are the edges of the $(1, 6)$-spanner). Hence, the running time per level is at most $O(n^2 + |L_i| \cdot (|E_i| + |S|)) = \tilde{O}(n^2)$.

Summing up over all levels, the running time is $\tilde{O}(n^2)$ plus the time to compute the spanner, which is $\tilde{O}(n^2)$ by Theorem 2, plus the time to select the portals per node, which is at most $O(n^2 \log n)$ (since they are already generated in sorted order).

4.2 Query Algorithm

Distance estimates $\tilde{d}(s, t)$ are computed as the minimum over all the triangulations via portals of s and t: $\min_{p \in P(s) \cup P(t)} \{\tilde{d}(s, p) + \tilde{d}(p, t)\}$, where $\tilde{d}(s, p)$ and $\tilde{d}(p, t)$ denote the distances stored at s and t, respectively, to/from landmark p.

4.2.1 Query Time

There are $K + 1$ portals per node, each triangulation requires two table lookups for landmark distances (one for $\tilde{d}(s, p)$ and the other one for $\tilde{d}(p, t)$), hence the query time is $O(K)$ (using perfect hashing). For the stretch analysis to work, we need $K \geqslant 12$ (twice the additive stretch of the spanner), hence the query time is constant.

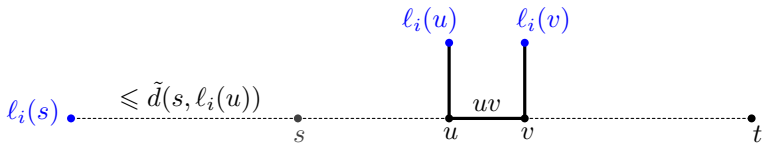


Figure 1 Worst-case stretch when triangulating via landmark $\ell_i(s)$. Distances are in $G = (V, E_i \cup S)$. Solid lines depict edges, dashed lines depict paths (which may have no, one, or multiple edge(s)). Edge uv is an edge on the shortest path with highest degree $\deg(uv)$. In this illustration, query node s is closer to uv than t. Since $\ell_i(s)$ is the nearest landmark for s, we have that $\tilde{d}(s, \ell_i(s)) \leqslant \tilde{d}(s, \ell_i(u))$.

4.2.2 Stretch Analysis

The stretch bound for the distance estimate $\tilde{d}(s,t)$ is derived as follows.

First, let us assume that, instead of triangulating via portals $P(\cdot)$ we were to triangulate via all closest landmarks of s and t, respectively. For each level i we compute the triangulation via the nearest landmark of s, i.e., $\tilde{d}(s, \ell_i(s)) + \tilde{d}(\ell_i(s), t)$, and via the nearest landmark of t, i.e., $\tilde{d}(s, \ell_i(t)) + \tilde{d}(\ell_i(t), t)$ and return the minimum.

The following argument is illustrated in Figure 1. Let E_i be the set that contains all the edges of the shortest s-t path and maximizes i. Let $uv \in E_i$ be an edge on this shortest path that is not in E_{i+1} (an edge between two high-degree nodes). The shortest-path length is $d(s,u) + 1 + d(v,t)$. Suppose $d(s,u) \leqslant d(v,t)$ (the other case is symmetric). Let us consider s and its nearest landmark $\ell_i(s)$. Since $u \in V_i$, u or its neighbor is in L_i, hence $\tilde{d}(s, \ell_i(s)) = d(s, \ell_i(s)) \leqslant d(s,u) + 1$. Triangulating via $\ell_i(s)$ increases the path length by at most $2d(s,u) + 2 \leqslant d(s,u) + d(v,t) + 2$, which yields stretch $(2,1)$. The same argument is also used by Berman and Kasiviswanathan [10].

However, we cannot afford to try all landmarks $\ell_i(\cdot)$ at query time since there may be logarithmically many for each query node. Instead, the preprocessing algorithm filters landmarks into a set of portals. For a node u, a landmark $\ell_i(u)$ is defined to *dominate* landmark $\ell_j(u)$ if $i < j$ and $\tilde{d}(u, \ell_i(u)) \leqslant \tilde{d}(u, \ell_j(u))$, which means that distances from $\ell_j(u)$ were computed in a subgraph of the one for which distances from $\ell_i(u)$ were computed. Let $D(u,i)$ denote the set of all landmarks $\ell_j(u)$ that dominate $\ell_i(u)$.

Again, let E_i be the set that contains all the edges of the shortest s-t path and maximizes i. We distinguish three cases. The first two cases are both fairly straightforward and could be combined. The most difficult case is the third, where query nodes are far away from the dense part of the graph, and hence also far away from the heavy edge uv.

- $\ell_i(s) \in P(s)$ and $\ell_i(t) \in P(t)$. The argument above applies since the query algorithm triangulates via both landmarks and computes a $(2,1)$-approximation.
- $\ell_i(s) \in P(s)$ or $D(s,i) \cap P(s) \neq \emptyset$, and the same holds for t. Let $\ell_j(s)$ be the landmark in $P(s)$ that dominates $\ell_i(s)$. When triangulating via $\ell_j(s)$, the stretch cannot increase since $d_{(V,E_j \cup S)}(s, \ell_j(s)) \leqslant d_{(V,E_i \cup S)}(s, \ell_i(s))$. In Figure 1, simply re-label $\ell_i(s)$ with $\ell_j(s)$. Since $E_i \subseteq E_j$ graph distances cannot increase. Analogous for t.
- Neither the landmark nor a dominating landmark are in the portal set. Now the spanner comes into play. At least one of the nodes must have had more than $K+1$ candidate portals (without loss of generality it is s), otherwise $\ell_i(s)$ or a landmark $\ell_j(s)$ dominating $\ell_i(s)$ would have to be in $P(s)$. The preprocessing algorithm truncated the portal list. Let $p^{(0)}(s), p^{(1)}(s), \ldots p^{(K)}(s)$ be the sequence of portals for s, ordered by increasing distance from s, and let $x := d(s, p^{(0)}(s))$. Each subsequent portal $p^{(j)}(s)$ increases the distance

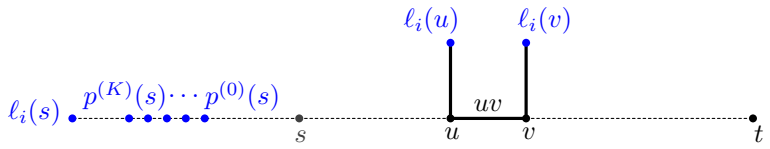


Figure 2 Worst-case stretch when triangulating via portal $p^0(s)$. There are $K+1$ portals closer than $\ell_i(s)$. When computing distances from/to these portals, the edges of the $(1,6)$-spanner were used, hence the distortion is bounded.

by at least 1, hence the last portal considered by the query algorithm is at distance at least $d(s, p^{(K)}(s)) \geqslant d(s, p^{(0)}(s)) + K$. Note that the best landmark for triangulation, $\ell_i(s)$, is even farther away, i.e., $d(s, \ell_i(s)) > d(s, p^{(K)}(s)) \geqslant d(s, p(0)(s)) + K = x + K$, otherwise it would have dominated $p^{(K)}(s)$. The shortest-path distance $d(s,t)$ is thus at least $x + K + y$, where $y := d(t, p^{(0)}(t))$. Without loss of generality, let us assume that s has a portal no farther than t has, i.e., $x \leqslant y$. Since the distance is long, we can safely triangulate using the first portal $p^{(0)}(s)$, which is relatively close to s. Due to always including the edges S of the $(1,6)$-spanner, the additive stretch when triangulating via $p^{(0)}(s)$ is at most $2(x+6)$, hence the estimate will be at most $(x + K + y) + 2(x + 6)$. Multiplicative stretch $(2,1)$ means the estimate can be $2(x + K + y) + 1$. Since $x \leqslant y$, the bound follows for $K \geqslant 12$. See Figure 2.

5 Conclusion

Distance oracles with stretch $(2,1)$ can be computed in quadratic time (modulo logarithmic factors) even for dense graphs. The main technical contribution is the use of a spanner without paying for its stretch penalty (the additive stretch of the spanner only affects the query time of the distance oracle). Previous attempts using a spanner increased the stretch to $(2,3)$, see [7].

The main difficulty lies in computing distances from/to landmarks. Since there are $n^{2/3}$ landmarks, we cannot afford to compute exact distances to all of them. Our algorithm allows constant additive distortion using a spanner. Other than pre-computing a spanner, the main bottleneck of the preprocessing algorithm in this paper is computing distances from $n^{2/3}$ nodes in a graph with $n^{4/3}$ edges. If there were a way to compute a spanner with additive stretch and fewer edges, there might be ways to push the preprocessing time below quadratic (if the number of edges m is sub-quadratic of course). However, Abboud and Bodwin [1] recently showed that Woodruff's spanner [35] is essentially as good as it gets for constant additive stretch. For stretch-3 distance oracles, there are sub-quadratic algorithms, increasing the stretch by a small additive constant [6]. For stretch-2 distance oracles, it seems like any further improvement to preprocessing times may have to find a way around landmarks or additive spanners.

Another open question is whether the result can be generalized to weighted graphs. The APASP algorithm of Berman and Kasiviswanathan [10] guarantees stretch $(2, w)$, where w denotes the largest edge length/weight on the shortest path. There are two main points where the algorithm and its analysis do not generalize: *a)* the additive spanner is insensitive to edge lengths, and *b)* each subsequent portal $p^{(i)}(\cdot)$ increases the distance by 1, which would not necessarily be true when edge lengths come into play.

Acknowledgments. Thanks to the anonymous reviewers for their feedback on earlier versions.

References

1. Amir Abboud and Greg Bodwin. The 4/3 additive spanner exponent is tight. In *48th ACM Symposium on Theory of Computing (STOC)*, 2016. To appear, available from: `http://arxiv.org/abs/1511.00700`.
2. Ittai Abraham and Cyril Gavoille. On approximate distance labels and routing schemes with affine stretch. In *25th International Symposium on Distributed Computing (DISC)*, pages 404–415, 2011. `doi:10.1007/978-3-642-24100-0_39`.
3. Rachit Agarwal. The space-stretch-time tradeoff in distance oracles. In *22nd European Symposium on Algorithms (ESA)*, pages 49–60, 2014. `doi:10.1007/978-3-662-44777-2_5`.
4. Rachit Agarwal and Philip Brighten Godfrey. Brief announcement: a simple stretch 2 distance oracle. In *32nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 110–112, 2013. `doi:10.1145/2484239.2484277`.
5. Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999. Announced at SODA 1996. `doi:10.1137/S0097539796303421`.
6. Surender Baswana, Akshay Gaur, Sandeep Sen, and Jayant Upadhyay. Distance oracles for unweighted graphs: Breaking the quadratic barrier with constant additive error. In *35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 609–621, 2008. `doi:10.1007/978-3-540-70575-8_50`.
7. Surender Baswana, Vishrut Goyal, and Sandeep Sen. All-pairs nearly 2-approximate shortest paths in $O(n^2 \text{poly} \log n)$ time. *Theoretical Computer Science*, 410(1):84–93, 2009. Announced at STACS 2005.
8. Surender Baswana and Telikepalli Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM Journal on Computing*, 39(7):2865–2896, 2010. Announced at FOCS 2006. `doi:10.1137/080737174`.
9. Surender Baswana and Sandeep Sen. Approximate distance oracles for unweighted graphs in expected $O(n^2)$ time. *ACM Transactions on Algorithms*, 2(4):557–577, 2006. Announced at SODA 2004. `doi:10.1145/1198513.1198518`.
10. Piotr Berman and Shiva Prasad Kasiviswanathan. Faster approximation of distances in graphs. In *10th International Workshop on Algorithms and Data Structures (WADS)*, pages 541–552, 2007.
11. William G. Brown. On graphs that do not contain a Thomsen graph. *Canadian Mathematical Bulletin*, 9:281–285, 1966.
12. Timothy M. Chan. Speeding up the four russians algorithm by about one more logarithmic factor. In *26th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 212–217, 2015.
13. Shiri Chechik. Approximate distance oracles with constant query time. In *46th ACM Symposium on Theory of Computing (STOC)*, pages 654–663, 2014. `doi:10.1145/2591796.2591801`.
14. Shiri Chechik. Approximate distance oracles with improved bounds. In *47th ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 2015. `doi:10.1145/2746539.2746562`.
15. Edith Cohen and Uri Zwick. All-pairs small-stretch paths. *Journal of Algorithms*, 38(2):335–353, 2001. Announced at SODA 1997. `doi:10.1006/jagm.2000.1117`.
16. Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. Announced at STOC 1987. `doi:10.1016/S0747-7171(08)80013-2`.

17 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000. Announced at FOCS 1996. doi:10.1137/S0097539797327908.

18 François Le Gall. Powers of tensors and fast matrix multiplication. In *39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

19 Cyril Gavoille and Christian Sommer. Sparse spanners vs. compact routing. In *23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 225–234, 2011. doi:10.1145/1989493.1989526.

20 Yijie Han and Tadao Takaoka. An $O(n^3 \log\log n/\log^2 n)$ time algorithm for all pairs shortest paths. In *13th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 131–141, 2012. doi:10.1007/978-3-642-31155-0_12.

21 Mihai Patrascu and Liam Roditty. Distance oracles beyond the Thorup-Zwick bound. *SIAM Journal on Computing*, 43(1):300–311, 2014. Announced at FOCS 2010. doi:10.1137/11084128X.

22 Mihai Patrascu, Liam Roditty, and Mikkel Thorup. A new infinity of distance oracles for sparse graphs. In *53rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 738–747, 2012.

23 David Peleg and Alejandro A. Schäffer. Graph spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi:10.1002/jgt.3190130114.

24 Istvan Reiman. Über ein Problem von K. Zarankiewicz. *Acta Mathematica Academiae Scientiarum Hungarica*, 9:269–273, 1958.

25 Liam Roditty, Mikkel Thorup, and Uri Zwick. Deterministic constructions of approximate distance oracles and spanners. In *32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 261–272, 2005. doi:10.1007/11523468_22.

26 Sandeep Sen. Approximating shortest paths in graphs. In *3rd International Workshop on Algorithms and Computation (WALCOM)*, pages 32–43, 2009. doi:10.1007/978-3-642-00202-1_3.

27 Christian Sommer. Shortest-path queries in static networks. *ACM Computing Surveys*, 46(4):45, 2014. doi:10.1145/2530531.

28 Christian Sommer, Elad Verbin, and Wei Yu. Distance oracles for sparse graphs. In *50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 703–712, 2009.

29 Andrew James Stothers. *On the Complexity of Matrix Multiplication.* PhD thesis, University of Edinburgh, 2010.

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

31 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, 2005. Announced at STOC 2001. doi:10.1145/1044731.1044732.

32 Rephael Wenger. Extremal graphs with no C^4's, C^6's, or C^{10}'s. *Journal of Combinatorial Theory, Series B*, 52:113–116, 1991.

33 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *46th ACM Symposium on Theory of Computing (STOC)*, pages 664–673, 2014. doi:10.1145/2591796.2591811.

34 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *44th Symposium on Theory of Computing (STOC)*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

35 David P. Woodruff. Additive spanners in nearly quadratic time. In *37th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 463–474, 2010. doi:10.1007/978-3-642-14165-2_40.

36 Christian Wulff-Nilsen. Approximate distance oracles with improved preprocessing time. In *23rd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 202–208, 2012. URL: http://portal.acm.org/citation.cfm?id=2095134&CFID=63838676&CFTOKEN=79617016.

37 Huacheng Yu. An improved combinatorial algorithm for boolean matrix multiplication. In *42nd International Colloquium on Automata, Languages, and Programming (ICALP 2015)*, pages 1094–1105, 2015. doi:10.1007/978-3-662-47672-7_89.

38 Uri Zwick. Exact and approximate distances in graphs – A survey. In *9th European Symposium on Algorithms (ESA)*, pages 33–48, 2001. doi:10.1007/3-540-44676-1_3.

Total Space in Resolution Is at Least Width Squared*

Ilario Bonacina†

KTH Royal Institute of Technology, School of Computer Science and Communication, Stockholm, Sweden
ilario@kth.se

Abstract

Given an unsatisfiable k-CNF formula φ we consider two complexity measures in Resolution: width and total space. The width is the minimal W such that there exists a Resolution refutation of φ with clauses of at most W literals. The total space is the minimal size T of a memory used to write down a Resolution refutation of φ, where the size of the memory is measured as the total number of literals it can contain. We prove that $T = \Omega((W-k)^2)$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems – Complexity of proof procedures

Keywords and phrases Resolution, width, total space

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.56

1 Introduction

Resolution is a well known propositional proof system introduced by Blake in [16] and proposed by Robinson in [38] for automated theorem proving. Since then this proof system became the most studied proof system in the sub-area of complexity theory that is *Proof Complexity*. Given a set of clauses φ, that is a set of disjunctions of literals or, equivalently, given a formula in Conjunctive Normal Form, *Resolution* is a method to infer new clauses according to the following inference rule:

$$\frac{C \vee x \quad D \vee \neg x}{C \vee D}, \tag{1}$$

where C, D are clauses and x is a variable. Resolution is sound and complete, that is it is possible to derive the empty clause $\perp$ if and only if φ is unsatisfiable. A Resolution *refutation* of φ is then just a sequence of clauses $C_1, \ldots, C_\ell$ with $C_\ell = \perp$ and each clause of the sequence is either a clause from φ or it is inferred by previous clauses in the sequence according to the inference rule in equation (1).

Nowadays, the main reason for the interest in Resolution comes from a practical perspective: it is at the core of most of the state-of-the-art SAT solvers since the introduction of the DPLL algorithm [22, 23] and its improvements, the so called *Conflict Driven Clause Learning* (CDCL) algorithms [4, 32, 40]. The track of the running of such algorithms on unsatisfiable instances produces a (particular form of) Resolution proofs. Hence Resolution is a valuable tool to study their performances and limitations. In this work we are interested in more theoretical questions about the Resolution proof system and the reader interested

* A full version of this paper is available online on ECCC [18].

† The author was was partially funded by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 56; pp. 56:1–56:13

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

in more details on the connections between Resolution and SAT solvers could look at the recent survey [35].

Given an unsatisfiable φ we are interested in measuring how complex a Resolution proof of φ must be. Certainly there are many ways of measuring the complexity of proofs and in this work we are interested in connecting two of such measures. The main complexity measure we can associate to φ in Resolution, and by far the most important, is the minimal length of a Resolution refutation of φ. This measure is denoted with $\mathsf{size}(\varphi \vdash \bot)$ and, since a long time now, we know that there are certain formulas φ_n requiring exponentially long proofs, e.g. the encodings of the Pigeonhole Principle [28] or Tseitin formulas [39, 42]. Another, easier to study, complexity measure is the *width*. Suppose that we focus on Resolution refutations of a formula φ with clauses up to a certain length w. The minimal w such that we have a refutation of φ with clauses of length at most w is the *width*, $\mathsf{width}(\varphi \vdash \bot)$. We have a trivial upper bound connecting size and width, that is for every set of clauses φ in n variables

$$\mathsf{size}(\varphi \vdash \bot) \leqslant n^{O(\mathsf{width}(\varphi \vdash \bot))},$$

and indeed this trivial bound could be asymptotically tight, cf. [3]. Another, more useful, connection between width and size is the following result from the seminal paper by Ben-Sasson and Wigderson [12]:

$$\log_2 \mathsf{size}(\varphi \vdash \bot) \geqslant \frac{(\mathsf{width}(\varphi \vdash \bot) - k)^2}{16n}, \tag{2}$$

where φ is a collection of clauses over n variables and each of them has at most k literals. Hence, if $\mathsf{width}(\varphi \vdash \bot) = \omega(\sqrt{n \log n})$ then, immediately by the previous inequality, $\mathsf{size}(\varphi \vdash \bot)$ is super-polynomial. Moreover this size-width inequality is essentially optimal [21].

Regarding the space complexity of proofs, its investigation was proposed in 1998 by Armin Haken as a natural analogue of the space complexity in the context of Turing machines and the first definitions of space measures in Resolution were given in [25, 1]. When talking about space, Resolution proofs are seen as a sequence of memory configurations $\mathfrak{M}_0, \ldots, \mathfrak{M}_\ell$, where each $\mathfrak{M}_i$ is a set of clauses, $\bot \in \mathfrak{M}_\ell$ and each $\mathfrak{M}_{i+1}$ derive from $\mathfrak{M}_i$ in one of the two following ways:

Axiom download: $\mathfrak{M}_{i+1} \subseteq \mathfrak{M}_i \cup \{C\}$, where $C \in \varphi$;

Inference: $\mathfrak{M}_{i+1} = \mathfrak{M}_i \cup \{D \vee E\}$, where both $D \vee x$ and $E \vee \neg x$ belong to $\mathfrak{M}_i$, for some variable x.

We then have some notions of how "spacious" a memory configuration can be. The most natural space measure for a memory configuration is of course the number of bits needed to write down it. Unfortunately it turns out that this notion of space is quite hard to study and hence some alternative notions of space were introduced [25, 1]. For example, the *clause space* of a memory configuration is the number of distinct clauses it can contain. The *total space*[1] of a memory configuration instead is the total number of literals it can contain. The minimal s such that we have a refutation of φ with memory configurations with total space at most s is the *Total Space* (needed to refute φ), $\mathsf{TSpace}(\varphi \vdash \bot)$. Similarly for the clause space we obtain $\mathsf{CSpace}(\varphi \vdash \bot)$. A more formal definition of $\mathsf{TSpace}(\varphi \vdash \bot)$, to avoid misunderstandings, is provided in Section 2.

[1] In [1] this space complexity measure is called *variable space*, but we follow [10, 9, 33, 11, 34] in calling it *total space*. This is due to distinguish it from a different space complexity measure in which different occurrences of the same variable are not counted, the *variable space*, investigated for instance in [43].

We now recall some known results about space complexity measures. Given any unsatisfiable set of clauses φ in n variables, in [25] it was proven that

$$\mathsf{CSpace}(\varphi \vdash \bot) \leqslant n + 1,$$

and, as a trivial consequence, we have that

$$\mathsf{TSpace}(\varphi \vdash \bot) \leqslant n(n + 1).$$

Both upper bounds are asymptotically tight, for example for random k-CNF formulas [8, 20]. Regarding lower bounds, in [2] it is proved that

$$\mathsf{CSpace}(\varphi \vdash \bot) \geqslant \mathsf{width}(\varphi \vdash \bot) - k + 1, \tag{3}$$

where φ consists of clauses of at most k literals. Clearly $\mathsf{TSpace}(\varphi \vdash \bot) \geqslant \mathsf{CSpace}(\varphi \vdash \bot)$ and whenever $\mathsf{TSpace}(\varphi \vdash \bot) = \omega(\mathsf{CSpace}(\varphi \vdash \bot))$ we say to have a *non-trivial* total space lower bound.

The total space measure was introduced in [1] and there the first non-trivial lower bounds were proven, for two particular class of formulas the Complete Tree formulas and the Pigeonhole Principle formulas. After that, in [20] it was introduced a technique to prove total space lower bounds in Resolution. That technique was sufficiently strong to prove asymptotically optimal total space lower bounds for instance for random k-CNFs [20, 13] but the proofs given there are quite long and involved. This paper, as a corollary, deeply simplify such proofs. Space complexity measures are also studied concerning trade-offs with other complexity measures, see for example [9, 36, 11, 34, 7, 5].

1.1 Contributions

This work is about proving an analogue of the inequality in (3) for the total space. This will add a nice bit to our knowledge of the lattice of relations between complexity measures in Resolution; it will simplify the proofs of existing total space lower bounds and it will imply new non-trivial total space lower bounds.

▶ **Theorem 1.** *Let φ be a k-CNF formula, then*

$$\mathsf{TSpace}(\varphi \vdash \bot) \geqslant \frac{1}{16} \left(\mathsf{width}(\varphi \vdash \bot) - k - 4\right)^2.$$

The general idea of the proof is the following: given a Resolution refutation, we identify a memory configuration where some small clause appear and then show that before that moment there must have been some memory configuration with a lot of clauses (and hence with large total space). This idea was originally used in [1] in some particular cases and in more generality in [20]. Indeed the proof we show has some close structural similarities with the total space lower bound from [20] and essentially it is a simplification of the proof of Theorem 2.5 from the author's PhD. Thesis [17]. The proof we give is not direct since it involves another, less studied, complexity measure: the *asymmetric width*, $\mathsf{awidth}(\varphi \vdash \bot)$, and families of assignments closely related to it. The asymmetric width was introduced in [29, 30] and the definition is quite technical so we defer it to Section 2 where we collect all the preliminary definitions and notations. The proof of Theorem 1 is purely combinatorial and implicitly uses some properties from a characterization of the asymmetric width from [15], cf. Section 3 for more details, together with a result tightly connecting the width and the asymmetric width, Theorem 2 (Lemma 8.5 from [29]).

Although defined quite differently, asymmetric width and width indeed share many properties. For instance, an analogue of the size-width inequality by Ben-Sasson and Wigderson [12]: given an unsatisfiable CNF formula φ in n variables

$$\ln\left(\mathsf{size}(\varphi \vdash \bot)\right) \geqslant \frac{\mathsf{awidth}(\varphi \vdash \bot)^2}{8n},$$

cf. Theorem 6.12 of [31]. For more information and history on the asymmetric width we refer to [15].

1.2 Examples of applications (and limitations) of Theorem 1

Since the seminal work of Ben-Sasson and Wigderson [12], the width measure has become one of the main tool to study Resolution proofs and their complexity. Hence we already have many relevant width lower bounds for many interesting class of formulas and then the range of applications of Theorem 1 is quite large. Below we recall some relevant examples.

Tseitin formulas. Given a d-regular graph G over n vertices, the Tseitin formula over G, $\mathsf{Tseitin}(G)$, is a CNF formula over $dn/2$ variables based on a propositional encoding of the fact that the total degree in any graph is even, see for example [12] for a formal definition. Such formulas were used by Tseitin to prove the first super-polynomial size lower bound for Resolution size [41]. Since then, Tseitin formulas became one of the standard tools in proof complexity to prove lower bounds and trade-offs, see for example [39, 43, 12, 25, 7]. In particular given a connected 3-regular graph G over n vertices which is an expander, we have that $\mathsf{width}(\mathsf{Tseitin}(G) \vdash \bot) \geqslant \Omega(n)$, cf. [12]. Hence, by Theorem 1, we have an asymptotically optimal total space lower bound: $\mathsf{TSpace}(\mathsf{Tseitin}(G) \vdash \bot) = \Theta(n^2)$. This answers the open question 4 from [1] in the case of Resolution.

Random k-CNFs. A random k-CNF with n variables and clause density Δ is a CNF formula picked as follows: choose independently uniformly at random Δn clauses from the set of all possible clauses in the variables $\{x_1, \ldots, x_n\}$ containing exactly k literals. If $\Delta = o(n^{1/4})$, Beame et al. [6] showed that random k-CNFs require exponential size Resolution proofs. Such result was simplified in [12] by showing a lower bound on width: if φ is a random k-CNF ($k \geqslant 3$) in n variables and Δn clauses, and Δ is a constant for simplicity, then with high probability $\mathsf{width}(\varphi \vdash \bot) \geqslant \Omega(n)$. Hence, by Theorem 1, with high probability $\mathsf{TSpace}(\varphi \vdash \bot) \geqslant \Omega(n^2)$. That is, almost every k-CNF require asymptotically optimal total space to be refuted in Resolution. This result was proven in [20] for $k \geqslant 4$ and for $k = 3$ in [13] with some explicit but quite involved constructions. Instead, as we saw, an asymptotically optimal total space lower bound for such formulas follows immediately from Theorem 1.

Formulas with short proofs. Bonet and Galesi [21] showed that the size-width inequality by Ben-Sasson and Wigderson [12] is essentially optimal. That is they showed that there are arbitrarily large 3-CNF formulas φ_n with $\Theta(n^3)$ clauses, $\Theta(n^2)$ variables and such that

- $\mathsf{width}(\varphi_n \vdash \bot) = \Theta(n)$,
- $\mathsf{CSpace}(\varphi_n \vdash \bot) = \Theta(n)$,

but φ_n has some Resolution proof of size $O(n^3)$, width $O(n)$ and clause space $O(n)$. Theorem 1 in this case tells us that $\mathsf{TSpace}(\varphi_n \vdash \bot) = \Omega(n^2)$, which is just a linear lower bound in the number of variables of φ_n. On the other hand this is a non-trivial total space lower bound since $\mathsf{TSpace}(\varphi_n \vdash \bot) = \omega(\mathsf{CSpace}(\varphi_n \vdash \bot))$.

Regarding the limitations, Theorem 1 suffers from the same kind of limitations of size-width inequality, equation (2), and the clause space-width inequality, equation (3). That is it became trivially vacuous for CNF formulas in n variables with clauses with many literals. For example we see such phenomenon when considering encodings of the negation of the Pigeonhole Principle as CNFs having clauses of n literals, the PHP_n^{n+1} formulas. For such formulas $\mathsf{width}(\mathsf{PHP}_n^{n+1} \vdash \bot) = \Theta(n)$ and hence no size lower bound or clause space lower bound could be implied directly from equations (2) - (3). The same applies for Theorem 1. On the other hand, by different techniques, we still have size lower bounds [42, 37], $\mathsf{size}(\mathsf{PHP}_n^{n+1} \vdash \bot) \geqslant 2^{\Omega(n)}$, clause space lower bounds [1], $\mathsf{CSpace}(\mathsf{PHP}_n^{n+1} \vdash \bot) \geqslant n$, and total space lower bounds [1, 20], $\mathsf{TSpace}(\mathsf{PHP}_n^{n+1} \vdash \bot) \geqslant \frac{1}{4}n^2$.

1.3 Organization of the paper

Section 2 contains all the preliminary definitions and notations needed for the proof of Theorem 1. In Section 3 we prove Theorem 1 and we give some more detailed comments on the proof. Section 4 contains some open questions about total space.

2 Preliminaries

We consider fixed a set of variables X and, given a natural number n, we denote as $[n]$ the set $\{1, \dots, n\}$. Given a set A, $\binom{A}{\leqslant 2}$ is the subset of the power set of A consisting of all the subsets of size at most 2.

Partial assignments. Given a set of variables X, a *partial (Boolean) assignment over* X is a function $\alpha : X \to \{0, 1\} \cup \{\star\}$. The *domain* of α is $\mathrm{dom}(\alpha) = \alpha^{-1}(\{0, 1\})$ and we say that α assigns a value to x if $x \in \mathrm{dom}(\alpha)$. Given two partial assignments over X, α and β we say that α extends β, $\beta \subseteq \alpha$, if for all $x \in X$, $\beta(x) \in \{\alpha(x), \star\}$. We denote by $\{x \mapsto b\}$ the partial assignment with domain the variable x mapped to $b \in \{0, 1\}$. Given two partial assignments α and β with disjoint domains, with $\alpha \cup \beta$ we denote the partial assignment with domain $\mathrm{dom}(\alpha) \cup \mathrm{dom}(\beta)$ such that for each $x \in \mathrm{dom}(\alpha) \cup \mathrm{dom}(\beta)$

$$\alpha \cup \beta(x) = \begin{cases} \alpha(x) & \text{if } x \in \mathrm{dom}(\alpha), \\ \beta(x) & \text{if } x \in \mathrm{dom}(\beta). \end{cases}$$

CNF formulas. A *literal* is a variable in X or the negation of a variable in X. A clause C is a formula of the form $\ell_1 \vee \cdots \vee \ell_k$, where the ℓ_i are literals and m is the *width* of the clause C, denoted as $|C|$. A formula in *Conjunctive Normal Form* (CNF) is a formula φ with variables in X of the form $C_1 \wedge \cdots \wedge C_m$, where the C_js are clauses. A k-CNF formula is a CNF formula where each clause has at most k distinct literals. With $\mathrm{var}(\varphi)$ we denote the set of variables occurring in the formula φ.

Given a CNF formula φ over a set of variables X and a partial assignment α over X, we can apply α to φ, obtaining a new CNF formula, denoted as $\varphi \restriction_\alpha$ or $\alpha(\varphi)$, in the following way: for each variable $x \in \mathrm{dom}(\alpha)$ substitute each occurrence of x in φ with $\alpha(x)$. Then simplify the resulting CNF according to the following rules: $\neg 0 \equiv 1$, $\neg 1 \equiv 0$, $0 \vee A \equiv A$, $1 \vee A \equiv 1$, $1 \wedge A \equiv A$, $0 \wedge A \equiv 0$. We say that α *satisfies* φ if $\alpha(\varphi) = 1$ and we say that α *falsifies* φ if $\alpha(\varphi) = 0$. Similarly, we can apply a partial assignment α to set of formulas $A = \{C_1, \dots, C_\ell\}$ component-wise: $A \restriction_\alpha = \{C_1 \restriction_\alpha, \dots, C_1 \restriction_\alpha\}$. Given a set of formulas F and a partial assignment α we say that α satisfies F, $\alpha \models F$, if and only if for every formula $\varphi \in F$, $\alpha(\varphi) = 1$.

Resolution proofs. A *Resolution derivation* of a clause C from a CNF formula φ is a sequence of clauses $\pi = (C_1, \ldots, C_\ell)$ such that $C_\ell = C$ and each C_i is either a clause from φ or it is inferred from C_j, C_k with $j, k < i$ and such that $\frac{C_j \quad C_k}{C_i}$ is a valid instance of the Resolution rule:

$$\frac{C \vee x \quad D \vee \neg x}{C \vee D}$$

where C, D are clauses and x is a variable; or, C_i is inferred from a C_j with $j < i$ and such that $\frac{C_j}{C_i}$ is a valid instance of the *weakening* inference rule[2]

$$\frac{C}{C \vee D}$$

where C, D are clauses. A *Resolution refutation* of a CNF formula φ is a Resolution derivation of the empty clause $\bot$ from φ. Resolution is sound and complete, that is it is possible to infer the empty clause $\bot$ from φ if and only if φ is unsatisfiable.

Width. Given a sequence of clauses $\pi = (C_1, \ldots, C_\ell)$ we recall that

$$\mathsf{width}(\pi) = \max_{C_j \in \pi} |C_j|$$

and the minimal width needed to refute φ in Resolution is

$$\mathsf{width}(\varphi \vdash \bot) = \min_{\pi} \mathsf{width}(\pi),$$

where the min is taken over all refutations of φ in Resolution[3].

Asymmetric width. The notion of *asymmetric width* was introduced in [30, 31]. Let φ be a CNF formula and $\pi = (C_1, \ldots, C_\ell)$ be a Resolution derivation from φ. To define the asymmetric width of π, $\mathsf{awidth}(\pi)$ we preliminary need the notion of *witness function*. A *witness function for* $\pi = (C_1, \ldots, C_\ell)$ is a function $\sigma : [\ell] \to \binom{[\ell]}{\leqslant 2} \cup \{\star\}$ witnessing the fact that π is a derivation from φ, that is such that

- $\sigma(i) = \{j, k\}$ implies that $j, k < i$ and $\frac{C_j \quad C_k}{C_i}$ is a valid instance of the inference rule of Resolution and if $j = k$ we require $\frac{C_j}{C_i}$ to be a valid instance of the weakening rule; and
- $\sigma(i) = \star$ implies that C_i is a clause from φ.

Given $\pi = (C_1, \ldots, C_\ell)$ a Resolution derivation from φ and a witness function σ for π, the *asymmetric width of* C_i *with respect to* π *and* σ, $aw_{\pi,\sigma}(C_i)$, is defined as follows

$$aw_{\pi,\sigma}(C_i) = \begin{cases} 0 & \text{if } \sigma(i) = \star, \text{ that is } C_i \in \varphi, \\ \min_{j \in \sigma(i)} |C_j| & \text{otherwise.} \end{cases}$$

Then $\mathsf{awidth}(\pi)$ is the minimum over all the possible functions σ witnessing the validity of π of the maximum over i of $aw_{\pi,\sigma}(C_i)$, that is

$$\mathsf{awidth}(\pi) = \min_{\sigma} \max_{C_i \in \pi} aw_{\pi,\sigma}(C_i).$$

---

[2] Notice that the weakening rule is not really needed but it will make simpler the exposition when dealing with restrictions of Resolution proofs.

[3] If φ is a satisfiable CNF formula then is customary to define $\mathsf{width}(\varphi \vdash \bot) = \infty$.

Finally, the asymmetric width needed to refute φ, $\mathsf{awidth}(\varphi \vdash \bot)$, is the minimum of $\mathsf{awidth}(\pi)$ over all possible sequence of clauses $\pi = (C_1, \ldots, C_\ell)$ that are Resolution refutations of φ.

Clearly it holds that $\mathsf{awidth}(\varphi \vdash \bot) \leqslant \mathsf{width}(\varphi \vdash \bot)$. Interestingly, the width cannot be much bigger than the asymmetric width.

▶ **Theorem 2** (Lemma 8.5 of [29]). *Let φ be an unsatisfiable k-CNF formula, then*

$$\mathsf{width}(\varphi \vdash \bot) \leqslant \mathsf{awidth}(\varphi \vdash \bot) + \max\{\mathsf{awidth}(\varphi \vdash \bot), k\}.$$

A self-contained proof of this result, essentially based on [14], is proven in the full version of this paper [18].

Total Space. As we saw in the introduction, a Resolution refutation of a CNF formula φ can be seen as a sequence of memory configurations $\pi = (\mathfrak{M}_0, \ldots, \mathfrak{M}_\ell)$, where each $\mathfrak{M}_i$ is a set of clauses, $\bot \in \mathfrak{M}_\ell$ and each $\mathfrak{M}_{i+1}$ derive from $\mathfrak{M}_i$ in one of the two following ways:
Axiom download: $\mathfrak{M}_{i+1} \subseteq \mathfrak{M}_i \cup \{C\}$, where $C \in \varphi$;
Inference: $\mathfrak{M}_{i+1} = \mathfrak{M}_i \cup \{D \vee E\}$, where both $D \vee x$ and $E \vee \neg x$ belong to $\mathfrak{M}_i$, for some variable x.
Given π as above, the *total space* of π is

$$\mathsf{TSpace}(\pi) = \max_{i \in [\ell]} \sum_{C \in \mathfrak{M}_i} |C|$$

and given an unsatisfiable CNF formula φ, the total space needed to refute φ in Resolution is

$$\mathsf{TSpace}(\varphi \vdash \bot) = \min_{\pi} \mathsf{TSpace}(\pi),$$

where the min is taken over all the possible Resolution refutations of φ given as a sequence of memory configurations[4].

3 Proof of Theorem 1

First let's prove the main result of this work, Theorem 1, for convenience of the reader restated below. We postpone more detailed comments on the proof after the proof itself.

▶ **Restated Theorem 1.** *Let φ be a k-CNF formula, then*

$$\mathsf{TSpace}(\varphi \vdash \bot) \geqslant \frac{1}{16} \left(\mathsf{width}(\varphi \vdash \bot) - k - 4\right)^2.$$

Proof. Let $\mathsf{awidth}(\varphi \vdash \bot) = r + 1$. We prove that

$$\mathsf{TSpace}(\varphi \vdash \bot) \geqslant \frac{1}{4}(r-1)^2,$$

or, more precisely, we prove that every Resolution refutation of φ must pass through a memory configuration of at least $(r-1)/2$ clauses each of width at least $(r-1)/2$. Once we prove this, the desired lower bound between total space and width follows:

$$\mathsf{TSpace}(\varphi \vdash \bot) \geqslant \frac{1}{4}(r-1)^2 \geqslant \frac{1}{16} \left(\mathsf{width}(\varphi \vdash \bot) - k - 4\right)^2,$$

[4] If φ is a satisfiable CNF formula then is customary to define $\mathsf{TSpace}(\varphi \vdash \bot) = \infty$.

where the last inequality uses that $\mathsf{width}(\varphi \vdash \bot) \leqslant 2(r+1)+k$, a consequence of Theorem 2.

Let Ξ and Ψ be two functions respectively mapping subsets of clauses into subsets of partial assignments and viceversa. Given a set of clauses A,

$$\Xi(A) = \{\alpha \text{ partial assignment} : \forall C \in A, \alpha(C) \neq 0\},$$

and given a set of partial assignments F,

$$\Psi(F) = \{C \text{ clause} : \exists \alpha \in F, \alpha(C) = 0\}.$$

Notice that, by construction, for every set of clauses A, $A \cap \Psi \circ \Xi(A) = \emptyset$ and $\bot \in \Psi(F)$ whenever F is non-empty. We consider the following special set:

$$W_r = \{C \text{ clause} : \mathsf{awidth}(\varphi \vdash C) \leqslant r\},$$

and its images $\Xi(W_r)$ and $\mathcal{S} = \Psi \circ \Xi(W_r)$. The main reason to consider the set $\Xi(W_r)$ is the following property:

▶ **Claim 3** (Extension Property of $\Xi(W_r)$)**.** *Let α be a $\subseteq$-maximal partial assignment in $\Xi(W_r)$ and x a variable not in $\mathrm{dom}(\alpha)$, then for every $\beta \subseteq \alpha$ such that $|\mathrm{dom}(\beta)| < r$ both $\beta \cup \{x \mapsto 0\}$ and $\beta \cup \{x \mapsto 1\}$ are in $\Xi(W_r)$.*

Proof. By contradiction let $\beta \subseteq \alpha$ such that $|\mathrm{dom}(\beta)| < r$ and $b \in \{0,1\}$ such that $\beta_b = \beta \cup \{x \mapsto b\} \notin \Xi(W_r)$. Without loss of generality we can restrict to consider $b = 0$. Since $\beta_0 \notin \Xi(W_r)$ it means that there exists a clause D in W_r such that $\beta_0(D) = 0$ but $\alpha(D) \neq 0$. This means that $D = D' \vee x$, $|D| \leqslant r$ and $\beta(D') = \alpha(D') = 0$. By maximality of α then both $\alpha_0 = \alpha \cup \{x \mapsto 0\} \notin \Xi(W_r)$ and $\alpha_1 = \alpha \cup \{x \mapsto 1\} \notin \Xi(W_r)$. In particular there exists a clause $E \in W_r$ such that $\alpha_1(E) = 0$, so, as before, we must have that $E = E' \vee \neg x$ and $\alpha(E') = 0$. But now

$$\frac{D' \vee x \quad E' \vee \neg x}{D' \vee E'}$$

is a valid instance of the Resolution rule. Hence, by definition of asymmetric width,

$$\mathsf{awidth}(\varphi \vdash D' \vee E') \leqslant \max\{\mathsf{awidth}(\varphi \vdash D), \mathsf{awidth}(\varphi \vdash E), r\} \leqslant r,$$

since both D and E belong to W_r and $|D| \leqslant r$. So $D' \vee E' \in W_r$ and $\alpha(D' \vee E') = 0$ which is a contradiction. ◀

Let $\pi = (\mathfrak{M}_0, \ldots, \mathfrak{M}_\ell)$ be a Resolution refutation of φ given as a sequence of memory configurations. By definition of W_r, $\bot \notin W_r$ and hence the empty partial assignment is in $\Xi(W_r)$, so, in particular $\bot \in \mathcal{S}$. Hence the following set is non-empty:

$$\mathcal{A} = \{i \in [\ell] \ : \ \exists C \in \mathfrak{M}_i \cap \mathcal{S}, \ |C| < (r-1)/2\}.$$

Let $t = \min \mathcal{A}$ and let $C \in \mathfrak{M}_t \cap \mathcal{S}$ be a clause of width less than $(r-1)/2$. Since $C \in \mathcal{S}$ there must exists a partial assignment $\alpha \in \Xi(W_r)$ that falsifies C and let α_C be the minimal partial assignment contained in α falsifying C. Notice that $|\mathrm{dom}(\alpha_C)| = |C| < (r-1)/2$. Our goal now is to show that there exists some $i < t$ such that $|\mathfrak{M}_i \cap \mathcal{S}| \geqslant (r-1)/2$. Since for every $i < t$ every clause in $\mathfrak{M}_i \cap \mathcal{S}$ has width at least $(r-1)/2$, this will give the desired result.

For sake of contradiction, suppose that for each $i < t$, $|\mathfrak{M}_i \cap \mathcal{S}| < (r-1)/2$. We inductively construct a sequence of assignments $\beta_0, \ldots, \beta_t$ in $\Xi(W_r)$ such that for each $i \leqslant t$ we have

that $\alpha_C \subseteq \beta_i$ and that $\beta_i \models \mathfrak{M}_i \cap \mathcal{S}$. This immediately give a contradiction when we reach β_t, since α_C falsifies the clause $C \in \mathfrak{M}_t \cap \mathcal{S}$ and $\beta_t \supseteq \alpha_C$.

The first memory configuration $\mathfrak{M}_0$ is empty, so we can put $\beta_0 = \alpha$. Supposing that $0 \leq i < t$ and that we already have a suitable β_i, we construct β_{i+1} distinguishing between two cases.

Axiom download case. $\mathfrak{M}_{i+1} \subseteq \mathfrak{M}_i \cup \{D\}$, where D is a clause from φ. Since each clause D from φ belongs to W_r and we have that $W_r \cap \mathcal{S} = \emptyset$, then $\mathfrak{M}_i \cap \mathcal{S} = \mathfrak{M}_{i+1} \cap \mathcal{S}$ and hence we can simply put $\beta_{i+1} = \beta_i$.

Inference case. $\mathfrak{M}_{i+1} \subseteq \mathfrak{M}_i \cup \{D \vee E\}$ where $D \vee E$ follows by Resolution on some variable x from two clauses $D \vee x$ and $E \vee \neg x$ in $\mathfrak{M}_i$. Then, by the inductive hypothesis, there exists $\beta_i \in \Xi(W_r)$ such that $\beta_i \models \mathfrak{M}_i \cap \mathcal{S}$, let $\bar{\beta}_i \in \Xi(W_r)$ be a $\subseteq$-maximal partial assignment containing β_i and let β be an assignment contained in $\bar{\beta}_i$ $\subseteq$-minimal such that $\alpha_C \subseteq \beta$ and $\beta \models \mathfrak{M}_i \cap \mathcal{S}$. We have that

$$|\mathrm{dom}(\beta)| \leqslant |\mathrm{dom}(\alpha_C)| + |\mathfrak{M}_i \cap \mathcal{S}| < (r-1)/2 + (r-1)/2 = r - 1,$$

where the first inequality follows easily from the fact that to satisfy a clause $F \in \mathfrak{M}_i \cap \mathcal{S}$ an assignment just have to satisfy a single literal in F. Notice that since $|\mathrm{dom}(\beta)| \leqslant r-2$ the extension property from Claim 3 can be applied twice and we will use this later. The main property of β that we now use is the following:

▶ **Claim 4.** *Let $\gamma \in \Xi(W_r)$ and F be any clause in $\mathfrak{M}_i$, if $\mathrm{var}(F) \subseteq \mathrm{dom}(\gamma)$ and $\beta \subseteq \gamma$, then $\gamma \models F$.*

Proof. Since $\mathrm{var}(F) \subseteq \mathrm{dom}(\gamma)$, then $\gamma(F) \in \{0, 1\}$. If by contradiction $\gamma(F) = 0$, then, by construction $F \in \mathcal{S}$ and, again by construction, $\beta \models \mathfrak{M}_i \cap \mathcal{S}$. So $\beta \models F$, which is is a contradiction since $\beta \subseteq \gamma$. ◀

The remaining part of the proof is just case analysis. If there is some variable y in $D \vee E$ unassigned by $\bar{\beta}_i$ then we can use the extension property (Claim 3) extending β to some $\beta' \in \Xi(W_r)$ setting y and satisfying $D \vee E$.

If $\mathrm{var}(D \vee E) \subseteq \mathrm{dom}(\beta)$ then we can extend β to some assignment $\beta' \in \Xi(W_r)$ setting x to some value (either by choosing $\bar{\beta}_i$ if $x \in \mathrm{dom}(\bar{\beta}_i)$ or otherwise by the extension property). Then $\mathrm{var}(D \vee x) \subseteq \mathrm{dom}(\beta')$, and, by the previous claim, $\beta' \models D \vee x$. The same happens for $E \vee \neg x$ and hence $\beta' \models D \vee E$ by the soundness of the Resolution rule.

The only remaining possibility is that $\mathrm{var}(D \vee E) \not\subseteq \mathrm{dom}(\beta)$ but $\mathrm{var}(D \vee E) \subseteq \mathrm{dom}(\bar{\beta}_i)$, and without loss of generality suppose that $\mathrm{var}(D) \not\subseteq \mathrm{dom}(\beta)$. If $x \in \mathrm{dom}(\bar{\beta}_i)$ then, by the previous claim $\bar{\beta}_i \models (D \vee x) \wedge (E \vee \neg x)$ so $\bar{\beta}_i \models D \vee E$. Suppose then that $x \notin \mathrm{dom}(\bar{\beta}_i)$. By Claim 3 we have that $\beta' = \beta \cup \{x \mapsto 0\} \in \Xi(W_r)$. Take a $\subseteq$-maximal assignment in $\Xi(W_r)$ containing β', let $\bar{\beta}'$ be such assignment. If $\mathrm{var}(D \vee x) \subseteq \mathrm{dom}(\bar{\beta}')$ then, by the previous claim, $\bar{\beta}_i \models D \vee x$, but $\beta'(x) = 0$ so $\bar{\beta}' \models D$ and hence $\bar{\beta}' \models D \vee E$. If $\mathrm{var}(D \vee x) \not\subseteq \mathrm{dom}(\bar{\beta}')$ then there is some variable y in $D \vee x$ not assigned by $\bar{\beta}'$ and since $|\mathrm{dom}(\beta')| = |\mathrm{dom}(\beta)| + 1 < r$ we can apply the extension property to β' extending it setting y and satisfying D. ◀

First of all notice that we proved something actually stronger, that is we proved that given an unsatisfiable CNF formula φ, every Resolution refutation of φ must pass through a memory configuration of at least $\frac{1}{2}(\mathsf{awidth}(\varphi \vdash \bot) - 2)$ clauses each of width at least $\frac{1}{2}(\mathsf{awidth}(\varphi \vdash \bot) - 2)$.

A crucial point in the proof of Theorem 1 is Claim 3. It is related with the following characterization of asymmetric width in Resolution by [15].

▶ **Theorem 5** (Theorem 22 from [15]). *Let φ be an unsatisfiable CNF formula, then the followings are equivalent:*

1. $\mathsf{awidth}(\varphi \vdash \bot) > r$.
2. *There exists a non-empty set $\mathcal{F}$ of partial assignments such that:*
   Consistency: *For every $\alpha \in \mathcal{F}$ and every clause C of φ, $\alpha(C) \neq 0$;*
   Extension: *If $\alpha \in \mathcal{F}$ and $\beta \subseteq \alpha$ is such that $|\mathrm{dom}(\beta)| < r$, then for every variable $x \notin \mathrm{dom}(\alpha)$ and for every $\epsilon \in \{0,1\}$ there exist $\beta_\epsilon \in \mathcal{F}$ with $\beta \subseteq \beta_\epsilon$ such that $\beta_\epsilon(x) = \epsilon$.*

Claim 3 is based on the proof of the implication from 1. to 2. in the previous theorem. The other implication (easier to prove) is not needed for Theorem 1. Indeed it is easy to see that given 1. the set of $\subseteq$-maximal partial assignments in $\Xi(W_r)$ satisfies the properties claimed in 2., and the crucial extension property is essentially Claim 3.

4 Open questions

We conclude this work with some open questions about the behaviour of the total space measure. Most of the questions are motivated by some analogy with the behaviour of the clause space measure.

On super-linear lower bounds. Is there any family of k-CNF formulas φ_n in n variables and $n^{O(1)}$ clauses such that $\mathsf{size}(\varphi_n \vdash \bot) = n^{O(1)}$ and $\mathsf{TSpace}(\varphi_n \vdash \bot) = \Theta(n^2)$?

For the formulas from [21] we saw in Section 1.2 we just have a linear total space lower bound. If we could find some formulas ψ_n with polynomial size Resolution proofs[5] but such that $\mathsf{width}(\psi_n \vdash \bot) = \omega(\sqrt{n})$ then, by Theorem 1, we would have that $\mathsf{TSpace}(\psi_n \vdash \bot) = \omega(n)$. This is anyway quite far from the question we are asking here and it seems that a positive answer should need some new techniques.

On simpler proofs for total space lower bounds. Is there a simpler more direct proof of a total space-width lower bound?

The clause space inequality $\mathsf{CSpace}(\varphi \vdash \bot) \geqslant \mathsf{width}(\varphi \vdash \bot) - k + 1$, where φ is a k-CNF, can be proven using some families of assignments and a characterization of Resolution width [2] or it can be proven (with some small loss in an additive constant) via some operation on Resolution proofs [26]. The proof we have of Theorem 1 is in a sense similar to the proof of the clause-width lower bound in [2] although not quite simple as that since we pass trough the asymmetric width and more complicated families of assignments.

Beyond Resolution. Space measures are defined in [1] also for proof systems stronger than Resolution. For example for Polynomial Calculus, a proof system where instead of clauses we infer polynomials, or $\mathsf{Res}(k)$, a (stronger) version of Resolution where instead of clauses k-DNF can be inferred, or Frege systems. In all such systems very little is known about space especially when it comes to total space. Regarding other space measures something is known for example for $\mathsf{Res}(k)$ [10, 24] and for Polynomial Calculus [1, 19, 27]. In [1] it is

[5] Equation 2 in this case implies that $\mathsf{width}(\psi_n \vdash \bot) = O(\sqrt{n \log n})$.

proven that in Frege system the total space is always at most linear and regarding Polynomial Calculus, the only lower bounds known for total space are from [1] and those are for the PHP_n^{n+1} formulas and the Complete Tree formulas. Is there any family of k-CNF formulas in n variables φ_n with $n^{O(1)}$ clauses requiring $\omega(n)$ total space to be refuted say in Polynomial Calculus?

Acknowledgements. The main question about width and total space is the result of some discussions had at the Dagstuhl seminar 15171: we want to thank Schloss Dagstuhl for the inspiring environment and the kind hospitality. We want to thank Jakob Nordström and Nicola Galesi for several long discussions on space complexity.

References

1 Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM J. Comput.*, 31(4):1184–1211, 2002. doi:10.1137/S0097539700366735.

2 Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *J. Comput. Syst. Sci.*, 74(3):323–334, 2008. doi:10.1016/j.jcss.2007.06.025.

3 Albert Atserias, Massimo Lauria, and Jakob Nordström. Narrow proofs may be maximally long. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 286–297. IEEE, 2014. doi:10.1109/CCC.2014.36.

4 Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In Benjamin Kuipers and Bonnie L. Webber, editors, *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island.*, pages 203–208. AAAI Press / The MIT Press, 1997. URL: http://www.aaai.org/Library/AAAI/1997/aaai97-032.php.

5 Paul Beame, Christopher Beck, and Russell Impagliazzo. Time-space tradeoffs in resolution: superpolynomial lower bounds for superlinear space. In Howard J. Karloff and Toniann Pitassi, editors, *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19-22, 2012*, pages 213–232. ACM, 2012. doi:10.1145/2213977.2213999.

6 Paul Beame, Richard M. Karp, Toniann Pitassi, and Michael E. Saks. On the complexity of unsatisfiability proofs for random k-CNF formulas. In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 561–571. ACM, 1998. doi:10.1145/276698.276870.

7 Chris Beck, Jakob Nordström, and Bangsheng Tang. Some trade-off results for polynomial calculus: extended abstract. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 813–822. ACM, 2013. doi:10.1145/2488608.2488711.

8 Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Struct. Algorithms*, 23(1):92–109, 2003. doi:10.1002/rsa.10089.

9 Eli Ben-Sasson and Jakob Nordström. Understanding space in resolution: optimal lower bounds and exponential trade-offs. In Peter Bro Miltersen, Rüdiger Reischuk, Georg Schnitger, and Dieter van Melkebeek, editors, *Computational Complexity of Discrete Problems, 14.09.–19.09.2008*, volume 08381 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, 2008. URL: http://drops.dagstuhl.de/opus/volltexte/2008/1781/.

10 Eli Ben-Sasson and Jakob Nordström. A space hierarchy for k-DNF resolution. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:47, 2009. URL: http://eccc.hpi-web.de/report/2009/047.

11 Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In Bernard Chazelle, editor, *Innovations in Computer Science – ICS 2010, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 401–416. Tsinghua University Press, 2011. URL: http://conference.itcs.tsinghua.edu.cn/ICS2011/content/papers/3.html.

12 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. *J. ACM*, 48(2):149–169, 2001. doi:10.1145/375827.375835.

13 Patrick Bennett, Ilario Bonacina, Nicola Galesi, Tony Huynh, Mike Molloy, and Paul Wollan. Space proof complexity for random 3-CNFs. *CoRR*, abs/1503.01613, 2015. URL: http://arxiv.org/abs/1503.01613.

14 Olaf Beyersdorff and Oliver Kullmann. Hardness measures and resolution lower bounds. *CoRR*, abs/1310.7627, 2013. URL: http://arxiv.org/abs/1310.7627.

15 Olaf Beyersdorff and Oliver Kullmann. Unified characterisations of resolution hardness measures. In Carsten Sinz and Uwe Egly, editors, *Theory and Applications of Satisfiability Testing – SAT 2014 – 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 170–187. Springer, 2014.

16 Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937. University of Chicago.

17 Ilario Bonacina. *Space in weak propositional proof systems*. PhD thesis, Department of Computer Science, Sapienza University of Rome, 12 2015.

18 Ilario Bonacina. Total space in resolution is at least width squared. *Electronic Colloquium on Computational Complexity (ECCC)*, 2016. http://eccc.hpi-web.de/report/2016/057/.

19 Ilario Bonacina and Nicola Galesi. A framework for space complexity in algebraic proof systems. *J. ACM*, 62(3):23, 2015. doi:10.1145/2699438.

20 Ilario Bonacina, Nicola Galesi, and Neil Thapen. Total space in resolution. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 641–650. IEEE Computer Society, 2014. doi:10.1109/FOCS.2014.74.

21 Maria Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, 2001. doi:10.1007/s000370100000.

22 Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962. doi:10.1145/368273.368557.

23 Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960. doi:10.1145/321033.321034.

24 Juan Luis Esteban, Nicola Galesi, and Jochen Messner. On the complexity of resolution with bounded conjunctions. *Theor. Comput. Sci.*, 321(2-3):347–370, 2004. doi:10.1016/j.tcs.2004.04.004.

25 Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Inf. Comput.*, 171(1):84–97, 2001. doi:10.1006/inco.2001.2921.

26 Yuval Filmus, Massimo Lauria, Mladen Mikša, Jakob Nordström, and Marc Vinyals. From small space to small width in resolution. In Ernst W. Mayr and Natacha Portier, editors, *31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014), STACS 2014, March 5-8, 2014, Lyon, France*, volume 25 of *LIPIcs*, pages 300–311. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.300.

27 Yuval Filmus, Massimo Lauria, Jakob Nordström, Noga Ron-Zewi, and Neil Thapen. Space complexity in polynomial calculus. *SIAM J. Comput.*, 44(4):1119–1153, 2015. doi:10.1137/120895950.

28 Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985. doi:10.1016/0304-3975(85)90144-6.

29 Oliver Kullmann. Investigating a general hierarchy of polynomially decidable classes of CNF's based on short tree-like resolution proofs. *Electronic Colloquium on Computational Complexity (ECCC)*, (41), 1999. URL: http://eccc.hpi-web.de/eccc-reports/1999/TR99-041/index.html.

30 Oliver Kullmann. An improved version of width restricted resolution. In *AMAI*, 2000. URL: http://rutcor.rutgers.edu/~amai/aimath00/regular/kullmann.ps.

31 Oliver Kullmann. Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems. *Ann. Math. Artif. Intell.*, 40(3-4):303–352, 2004. doi:10.1023/B:AMAI.0000012871.08577.0b.

32 Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*, pages 530–535. ACM, 2001. doi:10.1145/378239.379017.

33 Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution. *SIAM J. Comput.*, 39(1):59–121, 2009. doi:10.1137/060668250.

34 Jakob Nordström. Pebble games, proof complexity, and time-space trade-offs. *Logical Methods in Computer Science*, 9(3), 2013. doi:10.2168/LMCS-9(3:15)2013.

35 Jakob Nordström. On the interplay between proof complexity and SAT solving. *ACM SIGLOG News*, 2(3):19–44, August 2015.

36 Jakob Nordström and Johan Håstad. Towards an optimal separation of space and length in resolution. *Theory of Computing*, 9:471–557, 2013. doi:10.4086/toc.2013.v009a014.

37 Alexander A. Razborov. Proof complexity of pigeonhole principles. In Werner Kuich, Grzegorz Rozenberg, and Arto Salomaa, editors, *Developments in Language Theory, 5th International Conference, DLT 2001, Vienna, Austria, July 16-21, 2001, Revised Papers*, volume 2295 of *Lecture Notes in Computer Science*, pages 100–116. Springer, 2001.

38 John Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965. doi:10.1145/321250.321253.

39 Uwe Schöning. Resolution proofs, exponential bounds, and Kolmogorov complexity. In Igor Prívara and Peter Ruzicka, editors, *Mathematical Foundations of Computer Science 1997, 22nd International Symposium, MFCS'97, Bratislava, Slovakia, August 25-29, 1997, Proceedings*, volume 1295 of *Lecture Notes in Computer Science*, pages 110–116. Springer, 1997. doi:10.1007/BFb0029954.

40 João P. Marques Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48(5):506–521, 1999. doi:10.1109/12.769433.

41 G.S. Tseitin. On the complexity of derivation in propositional calculus. In Jörg H. Siekmann and Graham Wrightson, editors, *Automation of Reasoning*, Symbolic Computation, pages 466–483. Springer Berlin Heidelberg, 1983. URL: http://dx.doi.org/10.1007/978-3-642-81955-1_28, doi:10.1007/978-3-642-81955-1_28.

42 Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928.

43 Alasdair Urquhart. The depth of resolution proofs. *Studia Logica*, 99(1-3):349–364, 2011. doi:10.1007/s11225-011-9356-9.

Supercritical Space-Width Trade-Offs for Resolution

Christoph Berkholz[*1] **and Jakob Nordström**[†2]

1 **Humboldt-Universität zu Berlin, Berlin, Germany**
2 **KTH Royal Institute of Technology, Stockholm, Sweden**

Abstract

We show that there are CNF formulas which can be refuted in resolution in both small space and small width, but for which any small-width resolution proof must have space exceeding by far the linear worst-case upper bound. This significantly strengthens the space-width trade-offs in [Ben-Sasson 2009], and provides one more example of trade-offs in the "supercritical" regime above worst case recently identified by [Razborov 2016]. We obtain our results by using Razborov's new hardness condensation technique and combining it with the space lower bounds in [Ben-Sasson and Nordström 2008].

1998 ACM Subject Classification F.2.2 Analysis of Algorithms and Problem Complexity: Nonnumerical Algorithms and Problems – Complexity of proof procedures, F.1.3 Computation by Abstract Devices: Complexity Measures and Classes, I.2.3 Artificial Intelligence: Deduction and Theorem Proving, F.4.1 Mathematical Logic and Formal Languages: Mathematical Logic – computational logic

Keywords and phrases Proof complexity, resolution, space, width, trade-offs, supercritical

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.57

1 Introduction

Propositional proof complexity studies the problem of how to provide concise, polynomial-time checkable certificates that formulas in conjunctive normal form (CNF) are unsatisfiable. Research in this area was initiated in [20] as a way of attacking the problem of showing that NP $\neq$ coNP, and hence P $\neq$ NP, and it is therefore natural that the main focus has been on proving upper and lower bounds on proof length/size. More recently, however, other complexity measures have also been investigated, and this study has revealed a rich and often surprising web of connections.

Resolution Length, Width, and Space. Arguably the most thoroughly studied proof system in proof complexity is *resolution*, which appeared in [15] and began to be investigated in connection with automated theorem proving in the 1960s [21, 22, 33]. Because of its simplicity this proof system is well suited for proof search, and it lies at the heart of current state-of-the-art SAT solvers based on so-called conflict-driven clause learning [4, 26, 27].

* Part of the work of the first author was performed while at KTH Royal Institute of Technology supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD).
† The research of the second author was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 279611 and by Swedish Research Council grants 621-2010-4797 and 621-2012-5645.

It is not hard to show that any unsatisfiable CNF formula over n variables can be proven unsatisfiable, or *refuted*, by a resolution refutation containing $\exp(\mathrm{O}(n))$ clauses, and this holds even in the restricted setting of *tree-like resolution*, where each intermediate clause in the refutation has to be rederived from scratch every time it is used. In a breakthrough result, Haken [24] obtained a length lower bound on the form $\exp\big(\Omega\big(n^{\delta}\big)\big)$ for general resolution refutations of so-called pigeonhole principle formulas, and this paper was later followed by truly exponential lower bounds $\exp(\Omega(n))$ for other formula families in, e.g., [6, 18, 35].

In a seminal paper [12], Ben-Sasson and Wigderson identified *width*, measured as the largest size of any clause appearing in a refutation, as another interesting complexity measure for resolution. Clearly, any unsatisfiable CNF formula over n variables can be refuted in width at most n. Moreover, any refutation in width w need never be longer than $n^{\mathrm{O}(w)}$, since this is an upper bound on the number of distinct clauses of width w (and this naive counting argument is essentially tight [3]). What Ben-Sasson and Wigderson showed is that strong enough *lower* bounds on width also imply lower bounds on length; in particular that linear $\Omega(n)$ width lower bounds imply exponential $\exp(\Omega(n))$ length lower bounds. This connection can be used to rederive almost all currently known resolution length lower bounds.

Motivated by questions in SAT solving, where efficient memory management is a major concern, a more recent line of research in proof complexity has examined a third complexity measure on proofs, namely *space*. This study was initiated by Esteban and Torán [23], who defined the *(clause) space* of a resolution proof as the maximal number of clauses needed to be kept in memory during verification of the proof.[1] It can be shown that a CNF formula over n variables can always be refuted in space $n + \mathrm{O}(1)$ even in tree-like resolution [23], although the refutation thus obtained might have exponential length. Linear space lower bounds matching the worst-case upper bound up to constant factors were obtained for a number of formula families in [1, 9, 23].

These space lower bounds also matched known lower bounds on width, and in a strikingly simple and beautiful result Atserias and Dalmau [2] showed that in fact the resolution width of refuting a k-CNF formula F provides a lower bound for the clause space required. [2] This allows to recover the space lower bounds mentioned above as immediate consequences of width lower bounds shown in [12]. Furthermore, it follows from [2] that for $k = \mathrm{O}(1)$ any k-CNF formula that can be refuted by just keeping a constant number of clauses in memory can also be refuted in polynomial length and constant width. In the sequence of papers [28, 30, 10] it was shown, however, that there are formula families that have high space complexity although they have refutations in linear length and constant width.

Resolution Trade-offs. As was discussed above, a resolution proof in sufficiently small width will by necessity also be short, whereas the linear worst-case upper bound on space is achieved by a proof in exponential length. It is natural to ask, therefore, given a formula F, whether there exists a single refutation that can simultaneously optimize these different complexity measures. This question was first raised by Ben-Sasson [8], who gave a strong negative answer for space versus width. He showed that there are formulas which are refutable separately in constant width and in constant space, but for which any resolution proof minimizing one of the measures must exhibit almost worst-case linear behaviour with respect to the other.

---

[1] For completeness, we want to mention that for resolution there is also a *total space* measure counting the total number of literals in memory (with repetitions), which has been studied in [1, 13, 16, 17]. In this paper, however, "space" will always mean "clause space" in the sense of [23] unless otherwise stated.

[2] Note that this is a nontrivial connection since lower bound on width, i.e., the *number of literals* in a clause, is shown to imply essentially the same lower bound on the *number of clauses* needed.

A question that arises in the context of SAT solving is whether it is possible to simultaneously optimize size and space (corresponding to running time and memory usage). In [8] Ben-Sasson also proved a size-space trade-off for tree-like resolution, and building on [8, 10] it was shown in [11] that there are formulas which have refutations in linear length and also in small space, but for which any space-efficient refutation must have superpolynomial or even exponential length in general resolution. Beame et al. [5] and Beck et al. [7] exhibited formulas over n variables refutable in length polynomial in n where bringing the space down to linear, or even just shaving a constant factor of the polynomial space bound that follows immediately from the length bound, incurs a superpolynomial penalty in proof length.

Regarding length versus width, what was shown in [12] is that a short refutation can be converted to a refutation of small width, but this conversion blows up the length exponentially. Thapen [34] proved that this is inherent by exhibiting formulas refutable in small width and small length, but for which any small-width refutation has to have exponential length. For the restricted case of tree-like resolution, Razborov [32] recently showed that there are formulas refutable in small width for which any tree-like refutation even doing slightly better than the trivial linear upper bound with respect to width must by necessity have doubly exponential length.

We want to highlight an intriguing property of the trade-off results in [5, 7, 32] that sets them apart from the other trade-offs surveyed above. Namely, for most trade-off results between complexity measures it is the case that the trade-off plays out in the region between the worst-case upper bounds for the measures, where as one measure decreases the other measure has to approach its critical worst-case value. However, the short resolution proofs in [5, 7] require space even polynomially larger than the worst-case upper bound, and the small-width tree-like proofs in [32] require proofs of length exponential in the exponential upper bound for tree-like length. To underscore the dramatic nature of such trade-off results, Razborov refers to them as *ultimate* in the preliminary version [31] of [32], although in this paper we will instead use the term *supercritical trade-offs* to indicate that one of the complexity measures is pushed up into the supercritical regime above worst case when the other measure is decreased.

Our Contribution. Answering Razborov's call in [32] for more examples of the type of trade-offs discussed above, we prove a supercritical trade-off between space and width in resolution. As already observed, any refutation in width w of a CNF formula over n variables in general resolution need not contain more than $\mathrm{O}(n^w)$ clauses, which is also a trivial upper bound on the space complexity of such a refutation. Our main result is that this bound is essentially tight, and is also somewhat robust. Namely, we show that there are n-variable formulas that can be refuted in width w, but for which any refutation in width even up to almost a multiplicative logarithmic factor larger than this requires space $n^{\Omega(w)}$.

▶ **Theorem 1.1.** *For any constant $\varepsilon > 0$ and any non-decreasing function $\ell(n)$, $6 \leq \ell(n) \leq n^{\frac{1}{2}-\epsilon}$, there is a family $\{F_n\}_{n\in\mathbb{N}}$ of n-variable CNF formulas which can be refuted in resolution width $\ell(n)$ but for which any refutation in width $\mathrm{o}(\ell(n)\log n)$ requires space $n^{\Omega(\ell(n))}$.*

Techniques. In one sentence, we obtain our results by using Razborov's hardness condensation technique in [32] and combining it with the space lower bounds in [10].

In slightly more detail, our starting point are the so-called *pebbling formulas* defined in [12]. These formulas are refutable in constant width, but it was observed in [8] that space lower bounds for pebble games on directed acyclic graphs (DAGs) carry over to lower bounds on the number of *variables* kept simultaneously in memory in resolution refutations

of pebbling formulas defined over these DAGs. It was shown in [10] that substituting every variable in such formulas by an exclusive or of two new variables and expanding out to CNF produces a new family of formulas which are still refutable in constant width but for which the variable space lower bounds have been amplified to clause space lower bounds.

The result in [10] is one of several examples of how *XOR substitution*, or *XORification*, has been used to amplify weak proof complexity lower bounds to much stronger lower bounds. In all of these applications distinct variables of the original formula are replaced by disjoint sets of new variables. The wonderfully simple (with hindsight) but powerful new idea in [32] is to instead do XOR substitution with overlapping sets of variables from a much smaller variable pool (but with exclusive ors of higher arity).

This recycling of variables has the consequence that hardness amplification as in [10] no longer works, since it crucially depends on the fact that all new substitution variables are distinct. What Razborov showed in [32] was essentially that if the pattern of overlapping variable substitutions is described by a strong enough bipartite expander, then locally there are enough distinct new variables to make tree-like amplification lower bounds as in [8] go through over a fairly wide range of the parameter space, yielding supercritical trade-offs between width and tree-like length. Since in addition the number of variables in the formula has decreased significantly, this can be viewed as a kind of *hardness condensation*.

We use Razborov's idea of XORification with recycled variables, but since we want to obtain results for general, DAG-like resolution the technical details of our proofs are somewhat different. At a high level, we start with formulas over N variables that are refutable in constant width but require space $\Omega(N/\log N)$, to which we apply w-wise XORification using a much smaller set of n variables. We then show that from any refutation in width $O(w)$ of this new, XORified formula it is possible to recover a refutation of the original formula with comparable space complexity. But this means that any small-width refutation of the XORified formula must have space complexity roughly $\Omega(N/\log N)$. Choosing parameters so that $N \approx n^w$ yields the bound stated in Theorem 1.1.

We should point out that compared to [32] we get significantly less robust trade-offs, which break down already for a multiplicative logarithmic increase in width. This is mainly due to the fact that we deal not with tree-like resolution as in [32], but with general, DAG-like resolution. We share with [32] the less desirable feature that although our formulas only have n variables they contain on the order of n^w clauses. Thus, measured in terms of formula size our space-width trade-offs do not improve on [8], and the width of our formulas is not constant but scales linearly with w. Still, since the number of variables provides a worst-case upper bound on space (independently of formula size), measured in terms of variables it seems fair to say that the trade-off result in Theorem 1.1 is fairly dramatic.

Organization of This Paper. We start by reviewing some preliminaries in Section 2. In Section 3 we prove our main result assuming a hardness condensation lemma, and this lemma is then established in Section 4. We conclude in Section 5 with a discussion of possible directions for future research. Due to space constraints, we omit some of the proofs in this extended abstract, referring the reader to the upcoming full-length version for the missing details.

2 Preliminaries

A *literal* over a Boolean variable x is either the variable x itself (a *positive literal*) or its negation $\overline{x}$ (a *negative literal*). We define $\overline{\overline{x}} = x$. A *clause* $C = a_1 \vee \cdots \vee a_k$ is a disjunction

of literals over pairwise disjoint variables. A clause C' *subsumes* another clause C if every literal from C' also appears in C. A *k-clause* is a clause that contains at most k literals. A *CNF formula* $F = C_1 \wedge \cdots \wedge C_m$ is a conjunction of clauses. A *k-CNF formula* is a CNF formula consisting of k-clauses. We write $\mathit{Vars}(F)$ to denote the set of variables appearing in a formula F. We think of clauses and CNF formulas as sets: the order of elements is irrelevant and there are no repetitions.

A *resolution refutation* $\pi : F \vdash \bot$ of an unsatisfiable CNF formula F, which can also be referred to as a *resolution proof* for (the unsatisfiability of) F, is an ordered sequence of clauses $\pi = (D_1, \ldots, D_\tau)$ such that $D_\tau = \bot$ is the empty clause containing no literals, and each clause D_i, $i \in [\tau] = \{1, \ldots, \tau\}$, is either one of the clauses in F (an *axiom*) or is derived from clauses D_j, D_k in π with $j, k < i$ by the *resolution rule*

$$\frac{B \vee x \quad C \vee \overline{x}}{B \vee C} \ . \tag{2.1}$$

For technical reasons, it will also be convenient to permit a *weakening rule*

$$\frac{B}{B \vee C} \tag{2.2}$$

allowing to derive a strictly weaker clause from a clause already derived, although this rule is not essential.

With every resolution proof π we can associate a DAG G_π by having a sequence of vertices v_i on a line in order of increasing i, labelled by the clauses $D_i \in \pi$, and with directed edges (v_j, v_i) and (v_k, v_i) if the clause D_i was derived by resolution from D_j and D_k or an edge (v_j, v_i) if D_i was derived from D_j by weakening. Note that there might be several occurrences of a clause D in the proof π, and if so each occurrence gets its own vertex in G_π.

The *length* $L(\pi)$ of a resolution proof π is the number of clauses in it (counted with repetitions). The *width* $W(C)$ of a clause C is $|C|$, i.e., the number of literals, and $W(\pi)$ is the size of a largest clause in π. The *(clause) space* at step i is the number of clauses C_j, $j < i$, with edges to clauses C_k, $k \geq i$ in G_π plus 1 for the clause C_i derived at this step. Intuitively, space measures the number of clauses we need to keep in memory at step i, since they were derived before step i but are used to infer new clauses at or after step i. The space $\mathit{Sp}(\pi)$ of a proof π is the maximum space over all steps in π. Taking the minimum over all refutations, we define the length, width, and space of refuting F, respectively, as $L(F \vdash \bot) = \min_{\pi : F \vdash \bot}\{L(\pi)\}$, $W(F \vdash \bot) = \min_{\pi : F \vdash \bot}\{W(\pi)\}$, and $\mathit{Sp}(F \vdash \bot) = \min_{\pi : F \vdash \bot}\{\mathit{Sp}(\pi)\}$. We remark that any applications of the weakening rule (2.2) can always be eliminated from a refutation without increasing the length, width, or space.

When reasoning about space, it is sometimes convenient to use a slightly different, but equivalent, description of resolution that makes explicit what clauses are in memory at each point in time. We say that a *configuration-style resolution refutation* is a sequence $(\mathbb{D}_0, \ldots, \mathbb{D}_\tau)$ of sets of clauses, or *configurations*, such that $\mathbb{D}_0 = \emptyset$, $\bot \in \mathbb{D}_\tau$, and for all $t \in [\tau]$ the configuration $\mathbb{D}_t$ is obtained from $\mathbb{D}_{t-1}$ by one of the following *derivation steps*:

Axiom Download $\mathbb{D}_t = \mathbb{D}_{t-1} \cup \{C\}$, where C is a clause $C \in F$.

Inference $\mathbb{D}_t = \mathbb{D}_{t-1} \cup \{D\}$ for a clause D derived by resolution or weakening from clauses in $\mathbb{D}_{t-1}$.

Erasure $\mathbb{D}_t = \mathbb{D}_{t-1} \setminus \mathbb{D}'$ for some $\mathbb{D}' \subseteq \mathbb{D}_{t-1}$.

The length of a configuration-style refutation $\pi = (\mathbb{D}_0, \ldots, \mathbb{D}_\tau)$ is the number of axiom downloads and inference steps, the width is the size of a largest clause, as before, and the space is $\max_{t \in [\tau]}\{|\mathbb{D}_t|\}$. Given a refutation as an ordered sequence of clauses $\pi = (D_1, \ldots, D_\tau)$, we can construct a configuration-style refutation in the same length, width, and space by

deriving each clause D_i via an axiom download or inference step, and interleave with erasures of clauses C_j, $j < i$, as soon as they have no edges to clauses C_k, $k \geq i$ in the associated DAG G_π. In the other direction, taking a configuration-style refutation and listing the sequence of axiom download and inference steps yields a standard resolution refutation in the same length, width and space (assuming that clauses are erased as soon as possible). Thus, we can switch freely between these two ways of describing resolution refutations.

In fact, it will be convenient for us to limit our attention to a (slightly non-standard) restricted form of resolution refutations as described next. Let us say that a *homogeneous resolution refutation* is a refutation where every resolution rule application is of the form

$$\frac{C \vee x \quad C \vee \overline{x}}{C} \ . \tag{2.3}$$

The requirement of homogeneity is essentially without loss of generality, since we need to insert at most two weakening steps before each application of the resolution rule, which increases the width by at most 1, and the weakened clauses can then immediately be forgotten. We state this observation formally for the record.

▶ **Observation 2.1.** *If a CNF formula F has a standard resolution refutation without weakening steps in length L, width w, and space s, then it has a homogeneous refutation in length at most $3L$, width at most $w + 1$, and space at most $s + 2$.*

As already mentioned, a useful trick to obtain hard CNF formulas for different proof systems and complexity measures, which will play a key role also in this paper, is *XOR-ification*, i.e., substituting variables by exclusive ors of new variables and expanding out in the canonical way to obtain a new CNF formula. For example, the standard way to define binary XOR substitution for a positive literal x is

$$x[\oplus_2] = (x_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2) \ , \tag{2.4}$$

for a negative literal $\overline{y}$ we have

$$\overline{y}[\oplus_2] = (y_1 \vee \overline{y}_2) \wedge (\overline{y}_1 \vee y_2) \ , \tag{2.5}$$

and applying binary XOR substitution to the clause $x \vee \overline{y}$ we obtain the CNF formula

$$\begin{aligned}(x \vee \overline{y})[\oplus_2] = x[\oplus_2] \vee \overline{y}[\oplus_2] = \quad & (x_1 \vee x_2 \vee y_1 \vee \overline{y}_2) \wedge (x_1 \vee x_2 \vee \overline{y}_1 \vee y_2) \\ & \wedge (\overline{x}_1 \vee \overline{x}_2 \vee y_1 \vee \overline{y}_2) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{y}_1 \vee y_2) \ .\end{aligned} \tag{2.6}$$

The XORification of a CNF formula F is the conjunction of all the formulas corresponding to the XORified clauses of F. We hope that the reader excuses our slightly informal definition by example and has no problems generalizing it to substitutions with XOR of arbitrary arity (but see, e.g., Definition 2.12 in [29] for a more rigorous treatment).

Usually, the way XORification is done is that any two variables in the original formula are replaced by exclusive ors over disjoint sets of new variables. Razborov [32] observed that it can sometimes be useful to allow XORification with overlapping sets of variables. Let us define this concept more carefully.

▶ **Definition 2.2** (XORification with recycling [32])**.** Let F be a CNF formula over the set of variables $u_1, \ldots, u_N$ and let $\mathcal{G} = (U \dot\cup V, E)$ be a bipartite graph with left vertex set $U = \{u_1, \ldots, u_N\}$ and right vertex set $V = \{v_1, \ldots, v_n\}$. Then for the variables u_i we define the XORified literals $u_i[\mathcal{G}] = \bigoplus_{v \in \mathcal{N}(u_i)} v$ and $\overline{u}_i[\mathcal{G}] = \neg \bigoplus_{v \in \mathcal{N}(u_i)} v$ (where $\mathcal{N}(u_i)$ denotes the neighbours in V of u_i), for clauses $C \in F$ we define $C[\mathcal{G}] = \bigvee_{a \in C} a[\mathcal{G}]$ expanded out to CNF as in (2.6), and the *XORification of F with respect to $\mathcal{G}$* is defined to be $F[\mathcal{G}] = \bigwedge_{C \in F} C[\mathcal{G}]$.

Note that if F is an N-variable k-CNF with m clauses and $\mathcal{G} = (\{u_1, \ldots, u_N\} \dot\cup \{v_1, \ldots, v_n\}, E)$ is a bipartite graph of left degree d, then $F[\mathcal{G}]$ is an n-variable kd-CNF formula with most $2^{d-1}m$ clauses. We conclude this section with two simple observations that will be useful in what follows.

▶ **Observation 2.3.** *If F has a (homogeneous) resolution refutation in width w and $\mathcal{G}$ has left degree bounded by d, then $F[\mathcal{G}]$ can be refuted in (homogeneous) resolution in width $2dw$.*

This is not hard to show, and follows, e.g., from Theorem 2 in [11] (strictly speaking, this theorem is for XORification *without* recycling, but recycling can only decrease the width).

▶ **Observation 2.4.** *If F has a (homogeneous) resolution refutation π such that the associated DAG G_π has depth (i.e., longest path) s, then π can be carried out (in homogeneous resolution) in space $s + 2$ (possibly by repeating and/or reordering clauses in π).*

This second observation is essentially due to [23]. The proof DAG G_π can be turned into a binary tree of the same depth by repeating vertices/clauses, and it is then straightforward to show that any tree-like proof DAG in depth s can be realized in space at most $s + 2$.

3 Proof of Main Theorem

In this section we present a proof of Theorem 1.1. The proof makes use of the following hardness condensation lemma, which will be established in the next section and is the main technical contribution of the paper.

▶ **Lemma 3.1** (Hardness condensation lemma). *For all $k \in \mathbb{N}^+$ and $\varepsilon > 0$ there exist $n_0 \in \mathbb{N}^+$ and $\delta > 0$ such that the following holds. Let ℓ and n be integers satisfying $n \geq n_0$ and $k \leq \ell \leq n^{\frac{1}{2}-\varepsilon}$, and suppose that F is an unsatisfiable k-CNF formula over $N = \lfloor n^{\delta\ell} \rfloor$ variables which requires width $W(F \vdash \bot) = k$ and space $Sp(F \vdash \bot) = s$ to be refuted in resolution.*

Then there is a bipartite graph $\mathcal{G} = (U \dot\cup V, E)$ with $|U| = N$ and $|V| = n$ such that the n-variable CNF formula $F[\mathcal{G}]$ has the following properties:

- *$F[\mathcal{G}]$ can be refuted in width ℓ.*
- *Any refutation $\pi : F[\mathcal{G}] \vdash \bot$ in width $w \leq \frac{\ell}{4k} \log n$ requires space $Sp(\pi) \geq (s - w - 3)2^{-w}$.*

We want to apply this lemma to formulas of low width complexity but high space complexity as stated next.

▶ **Theorem 3.2** ([10]). *There is a family $\{F_N\}_{N\in\mathbb{N}}$ of N-variable 6-CNF formulas of size $\Theta(N)$ which can be refuted in width 6 but require space $Sp(F_N \vdash \bot) = \Omega(N/\log N)$.*

Combining Lemma 3.1 and Theorem 3.2, we can prove our main result.

Proof of Theorem 1.1. Recall that we want to prove that for any constant $\varepsilon > 0$ and any non-decreasing function $\ell(n) \leq n^{\frac{1}{2}-\epsilon}$ there is a family $\{F_n\}_{n\in\mathbb{N}}$ of n-variable CNF formulas which have a resolution refutation of width $\ell(n)$ but for which any refutation of width $o(\ell(n) \log n)$ requires clause space $n^{\Omega(\ell(n))}$.

From Theorem 3.2 we obtain constants $\varepsilon' > 0$ and $N_0 \in \mathbb{N}^+$ and a family of N-variable 6-CNF formulas F_N that require clause space $\varepsilon' N/\log N$ for all $N \geq N_0$. We want to apply Lemma 3.1 to these formulas. Let $\varepsilon > 0$ be given in Theorem 1.1 and fix $k = 6$. Plugging this into Lemma 3.1 yields $\delta > 0$ and $n_0 \in \mathbb{N}^+$, where in addition we choose n_0 large enough so that $\lfloor n_0^{\delta\ell(n_0)} \rfloor \geq N_0$ (this is always possible since $\delta\ell(n_0) \geq 6\delta > 0$).

For any $n \geq n_0$, set $N = \lfloor n^{\delta\ell(n)} \rfloor \geq N_0$ and let $\mathcal{G} = (U \dot\cup V, E)$ with $|U| = N$ and $|V| = n$ be the bipartite graph guaranteed by Lemma 3.1. Then the lemma says that $F_N[\mathcal{G}]$ is an n-variable formula which can be refuted in width ℓ, but for for which every refutation of width $w \leq \frac{\ell}{4k} \log n$ requires clause space of $(s - w - 3)2^{-w}$, where $s \geq \varepsilon' N / \log N = \varepsilon' \lfloor n^{\delta\ell(n)} \rfloor / \log \lfloor n^{\delta\ell(n)} \rfloor$ is the space lower bound for F_N. Choosing $w \leq \min\left(\frac{1}{4k}, \frac{\delta}{2}\right) \cdot \ell(n) \log n$ (recall that $w = \mathrm{o}(\ell(n) \log n)$ by assumption), the sequence of calculations

$$(s - w - 3)2^{-w} \geq \left(\varepsilon' \lfloor n^{\delta\ell(n)} \rfloor / \log \lfloor n^{\delta\ell(n)} \rfloor - \tfrac{\delta}{2}\ell(n) \log n\right) 2^{-\frac{\delta}{2}\ell(n)\log n} \geq \Omega\left(n^{\frac{\delta}{2}\ell(n)}\right) \quad (3.1)$$

yields the desired space lower bound. ◀

If one looks more closely at what is going on inside the proof of Theorem 1.1, where Lemma 3.1 and Theorem 3.2 come together, one can make the following somewhat intriguing observation. As discussed in the introduction, Theorem 3.2 is shown by using so-called pebbling formulas. Given a DAG $\mathcal{D}$ with sources S and a unique sink z, and with all non-sources having fan-in 2, we let every vertex in $\mathcal{D}$ correspond to a variable and define the *pebbling formula* $Peb_{\mathcal{D}}$ to consist of the following clauses:

- for all $s \in S$, a clause s,
- For all non-source vertices v with predecessors u_1, u_2, the clause $\overline{u}_1 \vee \overline{u}_2 \vee v$,
- for the sink z, the clause $\overline{z}$.

Applying standard binary XOR substitution (without recycling) as in (2.6) to these formulas amplifies weak lower bounds on the number of variables in memory $\mathit{VarSp}(\mathit{Peb}_{\mathcal{D}} \vdash \bot)$ (which follow from properties of the chosen DAG $\mathcal{D}$) to stronger lower bounds on the number of clauses $\mathit{Sp}(\mathit{Peb}_{\mathcal{D}}[\oplus_2] \vdash \bot)$. In Lemma 3.1 we then do another round of XOR substitution, this time with recycling, to decrease the number of variables while maintaining the space lower bound for small-width refutations. It is not entirely clear why we would need two separate rounds of XORification to achieve this result. In one sense, it would seem more satisfying to get a clean one-shot argument that just takes pebbling formulas and yields the supercritical trade-offs by only one round of XORification.

In fact, if we are willing to accept a slightly weaker bound, we could make such a one-shot argument and apply substitution with recycling directly to the pebbling formulas. The reason for this is that one can actually prove a somewhat stronger version of hardness condensation than in Lemma 3.1, as we will see in Section 4. There is no need to require that the original formula should have high space complexity unconditionally, but it suffices that the formula exhibits a strong trade-off between width and clause space. Since the number of clauses times the maximal width of any clause is an upper bound on the total number of distinct variables in memory, for any resolution refutation π we have the inequality $\mathit{Sp}(\pi) \cdot W(\pi) \geq \mathit{VarSp}(\pi)$. In [8] a variable space lower bound $\mathit{VarSp}(\mathit{Peb}_{\mathcal{D}} \vdash \bot) = \Omega(N/\log N)$ was presented (for appropriately chosen DAGs $\mathcal{D}$), implying that any width-w refutation requires clause space at least $\Omega(N/(w \log N))$. Since our hardness condensation step incurs a loss of a factor $1/2^w$, by starting with standard pebbling formulas and applying XORification with recycling directly we could obtain asymptotically similar bounds in one shot.

However, one can also argue that by combining Lemma 3.1 and Theorem 3.2 in the way done above one obtains a more modular proof, which shows that any formulas satisfying the conditions in Theorem 3.2 can be used for hardness condensation in a black-box fashion. This is why we chose to present the proof in this way.

4 Hardness Condensation

Let us now prove the hardness condensation lemma. We prove a slightly stronger version of the lemma below, which clearly subsumes Lemma 3.1.

▶ **Lemma 4.1** (Hardness condensation lemma, strong version). *For all $k \in \mathbb{N}^+$ and $\varepsilon > 0$ there are $n_0 \in \mathbb{N}^+$ and $\delta > 0$ such that the following holds. Let ℓ and n be integers satisfying $n \geq n_0$ and $k \leq \ell \leq n^{\frac{1}{2}-\varepsilon}$ and suppose that F is an unsatisfiable k-CNF formula over $N = \lfloor n^{\delta\ell} \rfloor$ variables which requires width $W(F \vdash \bot) = k$ to be refuted in resolution.*

Then there is a bipartite graph $\mathcal{G} = (U \dot\cup V, E)$ with $|U| = N$ and $|V| = n$ such that the n-variable CNF formula $F[\mathcal{G}]$ has the following properties:

- *$F[\mathcal{G}]$ can be refuted in width ℓ.*
- *Any refutation $\pi : F[\mathcal{G}] \vdash \bot$ of the XORified formula $F[\mathcal{G}]$ in width $w \leq \frac{\ell}{4k} \log n$ requires space $Sp(\pi) \geq (s - w - 3)2^{-w}$, where s is the minimal space of any refutation $\pi' : F \vdash \bot$ of the original formula F in width at most w.*

Clearly, the key to obtain Lemma 4.1 is to choose the right kind of graphs. As in [32], we use boundary expander graphs where the right-hand side is significantly smaller than the left-hand side. Let us start by giving a proper definition of these graphs and reviewing the properties of them that we need. Most of our discussion of boundary expanders can be recovered from [32], but since our setting of parameters is slightly different we give a self-contained presentation below. We refer to the full-length version of this paper for any missing proofs.

In what follows, we will let $\mathcal{G} = (U \dot\cup V, E)$ denote a bipartite graph with left vertices U and right vertices V. We write $\mathcal{N}(U') = \{v \mid \{u, v\} \in E(\mathcal{G}), u \in U'\}$ to denote the set of right neighbours of a left vertex subset $U' \subseteq U$ (and vice versa for right vertex subsets).

▶ **Definition 4.2** (Boundary expander). A bipartite graph $\mathcal{G} = (U \dot\cup V, E)$ is an $N \times n$ (r, c)-*boundary expander*, or *unique neighbour expander*, if $|U| = N$, $|V| = n$, and for every set $U' \subseteq U$, $|U'| \leq r$, it holds that $|\partial(U')| \geq c|U'|$, where $\partial(U') = \{v \in \mathcal{N}(U') : |\mathcal{N}(v) \cap U'| = 1\}$ is the *boundary* or the set of *unique neighbours* of U'. An (r, d, c)-boundary expander is an (r, c)-boundary expander where additionally $|\mathcal{N}(u)| \leq d$ for all $u \in U$, i.e., the left degree is bounded by d.

For a right vertex subset $V' \subseteq V$ in $\mathcal{G} = (U \dot\cup V, E)$ we define the *kernel* $\mathrm{Ker}(V') \subseteq U$ of V' to be the set of all left vertices whose entire neighbourhood is contained in V', i.e., $\mathrm{Ker}(V') = \{u \in U \mid \mathcal{N}(u) \subseteq V'\}$. We write $\mathcal{G} \setminus V'$ to denote the subgraph of $\mathcal{G}$ induced on $(U \setminus \mathrm{Ker}(V')) \dot\cup (V \setminus V')$. That is, we obtain $\mathcal{G} \setminus V'$ from $\mathcal{G}$ by first deleting V' and afterwards all isolated vertices from U.

A key property of boundary expanders is that for any small enough right vertex set V' we can always find a *closure* $\gamma(V') \supseteq V'$ with a small kernel on the left such that the subgraph $\mathcal{G} \setminus \gamma(V')$ has good boundary expansion. This is very similar to an analogous lemma in [32]. We omit the proof due to space constraints.

▶ **Lemma 4.3** ([32]). *Let $\mathcal{G}$ be an $(r, 2)$-boundary expander. Then for every $V' \subseteq V$ with $|V'| \leq r/2$ there exists a subset $\gamma(V') \supseteq V'$ such that $|\mathrm{Ker}(\gamma(V'))| \leq |V'|$ and the induced subgraph $\mathcal{G} \setminus \gamma(V')$ is an $(r/2, 1)$-boundary expander.*

The next lemma states that there exists $N \times n$ $(r, d, 2)$-boundary expanders where the size n of the right-hand side is significantly smaller than the size $N = n^{\Theta(d)}$ of the left-hand side. This can be proven by a standard application of the probabilistic method.

▶ **Lemma 4.4.** *Fix constants $\varepsilon, \delta, d_0 > 0$ such that $\delta + \frac{1}{d_0} < \varepsilon/2$. Then there exists an $n_0 \in \mathbb{N}^+$ such that for all n, d, and r satisfying $n \geq n_0$, $d_0 \leq d \leq n^{\frac{1}{2}-\varepsilon}$, and $r \leq d \log n$ there are $\lfloor n^{\delta d} \rfloor \times n$ $(r, d, 2)$-boundary expanders.*

After this review of boundary expanders and their properties we now come to the core argument of the paper, namely that space lower bounds are preserved for small-width proofs when we apply XORification as in Definition 2.2 with respect to an $(r, 2)$-boundary expander. To get cleaner technical arguments in the proofs we will restrict our attention to homogeneous resolution refutations as in (2.3), which for our purposes is without loss of generality by Observation 2.1.

▶ **Lemma 4.5.** *Let F be a CNF-formula and $\mathcal{G}$ an $(r, 2)$-boundary expander and suppose that $\pi : F[\mathcal{G}] \vdash \bot$ is a homogeneous resolution refutation in width $w \leq r/2$ of the XORified formula $F[\mathcal{G}]$. Then there is a homogeneous refutation $\pi' : F \vdash \bot$ of the original formula F in width at most w and space $Sp(\pi') \leq 2^w Sp(\pi) + w + 3$.*

Proof. Assume that $\pi = (\mathbb{C}_0, \mathbb{C}_1, \ldots, \mathbb{C}_\tau)$ is a configuration-style homogeneous resolution refutation of $F[\mathcal{G}]$ in width $W(\pi) = w \leq r/2$. We will show how to transform π into a refutation π' of the original formula F in width and space as claimed in the lemma. To help the reader navigate the proof, we remark that in what follows we will use the notational conventions that B and C denote clauses over $\mathit{Vars}(F[\mathcal{G}])$, D denotes a clause over $\mathit{Vars}(F)$, and A denotes an axiom clause from the original formula F before XORification.

Recall that for clauses $C \in F[\mathcal{G}]$ we have $\mathit{Vars}(C) \subseteq V$. For convenience, we will overload notation and write $\mathrm{Ker}(C) = \mathrm{Ker}(\mathit{Vars}(C))$, which is a subset of the variables U of the original formula F. Furthermore, for every clause $C \in \pi$ we fix $\gamma(C) := \gamma(\mathit{Vars}(C)) \subseteq V$ to be a minimal closure with properties as guaranteed by Lemma 4.3 (such closures exist since all clauses $C \in \pi$ have width at most w). An important notion in what follows will be that of *simultaneous falsifiability*, where we say that two CNF formulas F and G are *simultaneously falsifiable* if there is a truth value assignment that at the same time falsifies both F and G.

To transform the resolution refutation π of $F[\mathcal{G}]$ into a refutation π' of F we let $\mathbb{D}_t$ be obtained from $\mathbb{C}_t$ by replacing every clause $C \in \mathbb{C}_t$ by the set of clauses

$$\mathcal{G}^{-1}(C) := \{D \mid \mathit{Vars}(D) = \mathrm{Ker}(\gamma(C));\ D[\mathcal{G}] \text{ and } C \text{ are simultaneously falsifiable}\} \quad (4.1)$$

and defining

$$\mathbb{D}_t := \bigcup_{C \in \mathbb{C}_t} \mathcal{G}^{-1}(C) \quad (4.2)$$

(where the notation $\mathcal{G}^{-1}(C)$ is chosen to suggest that this is in some intuitive sense the inverse operation of XORification with respect to $\mathcal{G}$).

Every clause in $D \in \mathcal{G}^{-1}(C)$ has width at most w, because $|\mathit{Vars}(D)| = |\mathrm{Ker}(\gamma(C))| \leq W(C) \leq w$, where the first inequality is guaranteed by Lemma 4.3. Furthermore, we have $|\mathcal{G}^{-1}(C)| \leq 2^w$, since all clauses in $\mathcal{G}^{-1}(C)$ are over the same set of variables and each variable appears positively or negatively in every clause, and hence $\left|\mathbb{D}_t\right| \leq 2^w Sp(\pi)$. We want to argue that the sequence $\mathbb{D}_0, \mathbb{D}_1, \ldots, \mathbb{D}_\tau$ is the "backbone" of a resolution refutation π' of F, by which we mean that for every t it holds that $\mathbb{D}_{t+1}$ can be derived from $\mathbb{D}_t$ by a sequence of intermediate steps without affecting any proof complexity measure too much.

To show that this is so, we first observe that for $\mathbb{C}_0 = \emptyset$ we obviously get $\mathbb{D}_0 = \emptyset$ by (4.2). Moreover, it holds that $\mathcal{G}^{-1}(\bot) = \{\bot\}$ and hence $\bot \in \mathbb{D}_\tau$, since the unique minimal closure of the empty set is the empty set itself. We want to show that for every $0 \leq t < \tau$ the configuration $\mathbb{D}_{t+1}$ can be obtained from $\mathbb{D}_t$ by a resolution derivation

$(\mathbb{D}_t = \mathbb{D}_t^0, \mathbb{D}_t^1, \ldots, \mathbb{D}_t^{j_t} = \mathbb{D}_{t+1})$ where the space of every intermediate configuration is bounded by $\max\{Sp(\mathbb{D}_t), Sp(\mathbb{D}_{t+1}) + w\}$.

If $\mathbb{C}_{t+1}$ is obtained from $\mathbb{C}_t$ by erasing a clause C, then $\mathbb{D}_{t+1}$ can be obtained from $\mathbb{D}_t$ by erasing all clauses $\mathcal{G}^{-1}(C) \setminus \mathbb{D}_{t+1}$. Suppose that $\mathbb{C}_{t+1}$ is obtained from $\mathbb{C}_t$ by downloading an axiom $C \in F[\mathcal{G}]$. We claim that every clause in $\mathcal{G}^{-1}(C)$ is either an axiom or a weakening of an axiom from F. By the definition of $F[\mathcal{G}]$, every axiom $C \in F[\mathcal{G}]$ is a clause in the CNF formula $A[\mathcal{G}]$ for some original axiom $A \in F$. Fix any axiom $A \in F$ such that $C \in A[\mathcal{G}]$. Then for all $D \in \mathcal{G}^{-1}(C)$ it holds by (4.1) that $\mathit{Vars}(D) = \mathrm{Ker}(\gamma(C)) \supseteq \mathrm{Ker}(C) \supseteq \mathit{Vars}(A)$ and that there is an assignment falsifying both $D[\mathcal{G}]$ and C. To see that this implies that A subsumes D, suppose that there is a variable x appearing positively in A such that $\overline{x} \in D$. Any truth value assignment falsifying $D[\mathcal{G}]$ must falsify $a[\mathcal{G}]$ for all literals $a \in D$, and hence in particular $\overline{x}[\mathcal{G}]$. But this means that $x[\mathcal{G}]$ is satisfied by the same assignment, and then so is all of the formula $A[\mathcal{G}]$ including C. But this is a contradiction, and so not only does it hold that $\mathit{Vars}(A) \subseteq \mathit{Vars}(D)$ but A is in fact a subclause of D as claimed. From this we see that we can add the clauses $\mathcal{G}^{-1}(C)$ to $\mathbb{D}_t$ using axiom download and weakening. After applying a weakening step we immediately delete the old clause. Hence, the additional weakening might increase the space by at most one. It follows that the space of the intermediate configurations need never exceed $Sp(\mathbb{D}_{t+1}) + 1$.

It remains to check that $\mathbb{D}_{t+1}$ can be derived from $\mathbb{D}_t$ when $\mathbb{C}_{t+1}$ is obtained from $\mathbb{C}_t$ by an inference step. This is stated in the following two claims regarding applications of the resolution and weakening rules. Here graph expansion comes heavily into play, but due to space constraints we have to defer the proofs to the full-length version of this paper.

▶ **Claim 4.6.** *Every clause $D \in \mathcal{G}^{-1}(C)$ can be derived from $\mathcal{G}^{-1}(C \vee x) \cup \mathcal{G}^{-1}(C \vee \overline{x})$ by a homogeneous resolution derivation of width w and depth $w + 1$.*

▶ **Claim 4.7.** *For any two clauses $B \subseteq C$ it holds that every clause $D \in \mathcal{G}^{-1}(C)$ can be derived from $\mathcal{G}^{-1}(B)$ by a homogeneous derivation of width w and depth $w + 1$.*

Because the depth of a refutation is an upper bound on the clause space by Observation 2.4, it follows that in both cases we can derive all clauses in the clause set $\mathcal{G}^{-1}(C)$ one by one by using additional space $w+3$ to perform the derivations in depth $w+1$. It follows that F has a homogeneous resolution refutation π' of width w and clause space $Sp(\pi') \leq 2^w Sp(\pi) + w + 3$.

This concludes the proof of Lemma 4.5. ◀

We can now combine the construction in Lemma 4.5 with the existence of good boundary expanders in Lemma 4.4 to prove the hardness condensation in Lemma 4.1.

Proof of Lemma 4.1. Given $\varepsilon > 0$ and $k \in \mathbb{N}^+$ we choose $\delta := \frac{\varepsilon}{10k}$. Suppose ℓ and n are parameters such that $k \leq \ell \leq n^{\frac{1}{2}-\varepsilon}$ and let F be an unsatisfiable k-CNF formula over $N = \lfloor n^{\delta\ell} \rfloor$ variables that can be refuted in width k. To apply Lemma 4.4 we set $d_0 := \frac{5}{\varepsilon}$ and verify that $\delta + \frac{1}{d_0} = \frac{\varepsilon}{10k} + \frac{\varepsilon}{5} < \frac{\varepsilon}{2}$. We choose the degree of the expander to be $d := \frac{\ell}{2k}$ and set the expansion guarantee to $r := d \log n$. By the bound on ℓ we have $d \leq \ell \leq n^{\frac{1}{2}-\varepsilon}$.

Now we have two cases. The first, and interesting, case is when $d \geq d_0$ holds. Then Lemma 4.4 guarantees that there exists an $N \times n$ $(r, d, 2)$-boundary expander $\mathcal{G}$. Applying XORification with respect to $\mathcal{G}$, we obtain an ℓ-CNF formula $F[\mathcal{G}]$ with n variables. By Observation 2.3 it holds that $F[\mathcal{G}]$ has a resolution refutation of width $2dk \leq \ell$. Now suppose that $\pi : F[\mathcal{G}] \vdash \bot$ is a refutation of width w. Because $w \leq \frac{\ell}{4k} \log n = r/2$ the space lower bound follows from Lemma 4.5.

The second case is when $d < d_0$. Then we do not actually need any XORification but can use the original formula. Formally, let $\mathcal{G} = (U \dot\cup (V \cup V'), E)$ be a matching between two sets

U and V of size $|U| = |V| = N$ plus some isolated vertices V' on the right-hand side such that $|V \cup V'| = n$. To check that this is well defined we have to verify that $N \leq n$, which follows from the calculations $N = \lfloor n^{\delta\ell} \rfloor = \lfloor n^{\frac{\varepsilon}{10k}2kd} \rfloor \leq \lfloor n^{\frac{\varepsilon}{10k}2kd_0} \rfloor = \lfloor n^{\frac{\varepsilon}{10k}2k\frac{5}{\varepsilon}} \rfloor = n$. Thus, we obtain $F[\mathcal{G}] = F$ (plus some left-over dummy variables) and we have $W(F[\mathcal{G}] \vdash \bot) = W(F \vdash \bot) = k \leq \ell$ as well as $Sp(\pi) \geq s \geq (s - w - 3)2^{-w}$. ◀

5 Concluding Remarks

In this paper we prove that there are CNF formulas over n variables exhibiting an $n^{\Omega(w)}$ clause space lower bound for resolution refutations in width w. This lower bound is optimal (up to constants in the exponent) as every refutation in width w has length, and hence space, at most $n^{\mathrm{O}(w)}$. Our lower bounds do not only hold for the minimal refutation width w but remain valid for any refutations in width $\mathrm{o}(w \log n)$. Measured in terms of the number of variables n, this is a major improvement over the previous space-width trade-off result in [8], and provides another example of trade-offs in the supercritical regime above worst-case recently identified in [32].

A first open problem is whether the range of applicability can be extended even further so that the space lower bound holds true up to width $\mathrm{o}(n)$. It is clear that the lower bound has to break down at some point, since if one is allowed maximal width n any formula can be refuted in clause space $n + 2$. A supercritical trade-off on resolution proof depth over width ranging from w all the way up to $n^{(1-\epsilon)}/w$ was shown in [32], suggesting that the above goal might not be completely out of reach.

Another intriguing open problem from the complexity-theoretic point of view is to prove space trade-offs that are superlinear not only in terms of the number of variables but measured also in formula size. Such lower bounds cannot be obtained by the techniques used in this paper, but they are likely to exist as the following argument shows (see [25] for a more detailed discussion). Suppose that every refutation in width $w(n)$ can be transformed into a refutation that has width $w(n)$ and clause space polynomial in the size of the formula. Then we can find such a refutation non-deterministically in polynomial space by keeping the current configuration in memory and guessing the inference steps. Thus, by Savitch's theorem, finding refutations of width $w(n)$ would be in deterministic PSPACE. On the other hand, it has been shown by the first author that the problem of finding resolution refutations of bounded width is EXPTIME-complete [14]. Hence, unless EXPTIME = PSPACE there are formulas where every refutation of minimal width needs clause space that is superpolynomial in the size of the formula.

Finally, it would be interesting to study if the supercritical trade-offs between clause space and width in resolution shown in this paper could be extended to similar trade-offs between monomial space and degree for polynomial calculus or polynomial calculus resolution as defined in [1, 19].

Acknowledgements. We wish to thank Alexander Razborov for patiently explaining the hardness condensation technique in [32] during numerous and detailed discussions.

References

1 Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. Preliminary version in *STOC'00*.

2 Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, May 2008. Preliminary version in *CCC'03*.

3 Albert Atserias, Massimo Lauria, and Jakob Nordström. Narrow proofs may be maximally long. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity (CCC'14)*, pages 286–297, June 2014.

4 Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97)*, pages 203–208, July 1997.

5 Paul Beame, Chris Beck, and Russell Impagliazzo. Time-space tradeoffs in resolution: Superpolynomial lower bounds for superlinear space. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC'12)*, pages 213–232, May 2012.

6 Paul Beame, Richard Karp, Toniann Pitassi, and Michael Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, 31(4):1048–1075, 2002. Preliminary versions of these results appeared in *FOCS'96* and *STOC'98*.

7 Chris Beck, Jakob Nordström, and Bangsheng Tang. Some trade-off results for polynomial calculus. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC'13)*, pages 813–822, May 2013.

8 Eli Ben-Sasson. Size-space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, May 2009. Preliminary version in *STOC'02*.

9 Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Structures and Algorithms*, 23(1):92–109, August 2003. Preliminary version in *CCC'01*.

10 Eli Ben-Sasson and Jakob Nordström. Short proofs may be spacious: An optimal separation of space and length in resolution. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS'08)*, pages 709–718, October 2008.

11 Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS'11)*, pages 401–416, January 2011.

12 Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC'99*.

13 Patrick Bennett, Ilario Bonacina, Nicola Galesi, Tony Huynh, Mike Molloy, and Paul Wollan. Space proof complexity for random 3-CNFs. Technical Report 1503.01613, arXiv.org, April 2015.

14 Christoph Berkholz. On the complexity of finding narrow proofs. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*, pages 351–360, October 2012.

15 Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.

16 Ilario Bonacina. Total space in resolution is at least width squared. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP'16)*, July 2016. To appear.

17 Ilario Bonacina, Nicola Galesi, and Neil Thapen. Total space in resolution. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, pages 641–650, October 2014.

18 Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

19 Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC'96)*, pages 174–183, May 1996.

20 Stephen A. Cook and Robert Reckhow. The relative efficiency of propositional proof systems. *Journal of Symbolic Logic*, 44(1):36–50, March 1979.

21 Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

22 Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

23 Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001. Preliminary versions of these results appeared in *STACS'99* and *CSL'99*.

24 Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

25 Alexander Hertel. *Applications of Games to Propositional Proof Complexity*. PhD thesis, University of Toronto, May 2008. Available at `http://www.cs.utoronto.ca/~ahertel/`.

26 João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD'96*.

27 Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)*, pages 530–535, June 2001.

28 Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution. *SIAM Journal on Computing*, 39(1):59–121, May 2009. Preliminary version in *STOC'06*.

29 Jakob Nordström. Pebble games, proof complexity and time-space trade-offs. *Logical Methods in Computer Science*, 9:15:1–15:63, September 2013.

30 Jakob Nordström and Johan Håstad. Towards an optimal separation of space and length in resolution. *Theory of Computing*, 9:471–557, May 2013. Preliminary version in *STOC'08*.

31 Alexander A. Razborov. An ultimate trade-off in propositional proof complexity. Technical Report TR15-033, Electronic Colloquium on Computational Complexity (ECCC), March 2015.

32 Alexander A. Razborov. A new kind of tradeoffs in propositional proof complexity. *Journal of the ACM*, 63:16:1–16:14, April 2016.

33 John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

34 Neil Thapen. A trade-off between length and width in resolution. Technical Report TR14-137, Electronic Colloquium on Computational Complexity (ECCC), October 2014.

35 Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

Deterministic Time-Space Trade-Offs for k-SUM

Andrea Lincoln[*1], **Virginia Vassilevska Williams**[†2], **Joshua R. Wang**[‡3], **and R. Ryan Williams**[§4]

1 **Computer Science Department, Stanford University, Stanford, USA**
andreali@cs.stanford.edu
2 **Computer Science Department, Stanford University, Stanford, USA**
virgi@cs.stanford.edu
3 **Computer Science Department, Stanford University, Stanford, USA**
joshua.wang@cs.stanford.edu
4 **Computer Science Department, Stanford University, Stanford, USA**
rrw@cs.stanford.edu

Abstract

Given a set of numbers, the k-SUM problem asks for a subset of k numbers that sums to zero. When the numbers are integers, the time and space complexity of k-SUM is generally studied in the word-RAM model; when the numbers are reals, the complexity is studied in the real-RAM model, and space is measured by the number of reals held in memory at any point.

We present a time and space efficient deterministic self-reduction for the k-SUM problem which holds for both models, and has many interesting consequences. To illustrate:

- 3-SUM is in deterministic time $O(n^2 \lg\lg(n)/\lg(n))$ and space $O\left(\sqrt{\frac{n\lg(n)}{\lg\lg(n)}}\right)$. In general, any polylogarithmic-time improvement over quadratic time for 3-SUM can be converted into an algorithm with an identical time improvement but low space complexity as well.
- 3-SUM is in deterministic time $O(n^2)$ and space $O(\sqrt{n})$, derandomizing an algorithm of Wang.
- A popular conjecture states that 3-SUM requires $n^{2-o(1)}$ time on the word-RAM. We show that the 3-SUM Conjecture is in fact equivalent to the (seemingly weaker) conjecture that every $O(n^{.51})$-space algorithm for 3-SUM requires at least $n^{2-o(1)}$ time on the word-RAM.
- For $k \geq 4$, k-SUM is in deterministic $O(n^{k-2+2/k})$ time and $O(\sqrt{n})$ space.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems

Keywords and phrases 3SUM, kSUM, time-space trade-off, algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.58

1 Introduction

We consider the k-SUM problem: given a list S of n values, determine whether there are distinct $a_1, \ldots, a_k \in S$ such that $\sum_{i=1}^{k} a_i = 0$. This classic problem is a parameterized version of the Subset Sum problem, which is among Karp's original NP-Complete problems.[1]

The brute-force algorithm for k-SUM runs in $O(n^k)$ time, and it is known [22] that an $n^{o(k)}$ time algorithm (where the little-o depends on k) would violate the Exponential Time

* Supported by a Stanford Graduate Fellowship.
† Supported by NSF Grants CCF-1417238, CCF-1528078 and CCF-1514339, and BSF Grant BSF:2012338.
‡ Supported by a Stanford Graduate Fellowship.
§ Supported in part by NSF CCF-1552651. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

[1] Karp's definition of the Knapsack problem is essentially Subset Sum [19].

Hypothesis [18]. A faster meet-in-the-middle algorithm reduces the k-SUM problem on n numbers to 2-SUM on $O(n^{\lceil k/2 \rceil})$ numbers, which can then be solved by sorting and binary search in $O(n^{\lceil k/2 \rceil} \log n)$ time. The belief that this meet-in-the-middle approach is essentially time-optimal is at the heart of many conditional 3-SUM-hardness results in computational geometry (e.g. [15]) and string matching (e.g. [5, 2]).

The space usage of the meet-in-the-middle approach is prohibitive: the $O(n \log n)$ time solution for 2-SUM uses linear space, which causes the fast k-SUM algorithm to need $\Omega(n^{\lceil k/2 \rceil})$ space. However, the brute-force algorithm needs only $O(k \log n)$ space. This leads to the natural question: *how well can one trade off time and space in solving k-SUM?*

Schroeppel and Shamir [23] first studied time-space trade-off algorithms for Subset Sum. They showed how to reduce Subset Sum to an instance of k-SUM for any $k \geq 2$: split the elements into k sets of n/k elements each; for each set, compute $2^{n/k}$ sums corresponding to the subsets of the set; this forms a k-SUM instance of size $2^{n/k}$. Since the k-SUM instance does not have to be explicitly stored, any time $T(N)$, space $S(N)$ algorithm for $k-$SUM immediately implies a time $T(2^{n/k})$, space $S(2^{n/k})$ algorithm for Subset Sum. Furthermore, Schroeppel and Shamir gave a deterministic $\tilde{O}(n^2)$ time, $\tilde{O}(n)$ space algorithm for 4-SUM, implying a $O^*(2^{n/2})$ time, $O^*(2^{n/4})$ space algorithm for Subset Sum.[2] They also generalized the algorithm to provide a smooth time-space trade-off curve, with extremal points at $O^*(2^{n/2})$ time, $O^*(2^{n/4})$ space and $O^*(2^n)$ time, $O^*(1)$ space.

A recent line of work leading up to Austrin et al. [6] has improved this long-standing trade-off curve for Subset Sum via *randomized* algorithms, resulting in a more complex curve. Wang [25] moved these gains to the k-SUM setting. In particular, for 3-SUM he obtains an $\tilde{O}(n^2)$ time, $\tilde{O}(\sqrt{n})$ space Las Vegas algorithm.

Despite the recent progress on the problem, all of the improved algorithms for the general case of k-SUM have heavily relied on randomization, either utilizing hashes or random prime moduli. These improvements also all rely heavily on the values in the lists being integers. For the general case of k-SUM, the previous best *deterministic* k-SUM results (even for integer inputs) are the brute-force algorithm, the meet-in-the-middle algorithm, and the Schroeppel and Shamir 4-SUM algorithm, and simple combinations thereof.

1.1 Our Results

We consider new ways of trading time and space in solving k-SUM, on both integer and real inputs (on the word-RAM and real-RAM respectively), without the use of randomization. Our improvements for k-SUM naturally extend to improvements to Subset Sum as well.

Our main result is a deterministic self-reduction for k-SUM. Informally, we show how to deterministically decompose a list of n numbers into a small collection of shorter lists, such that the k-SUM solution is preserved. This result is shown for $k = 3$ in Section 4. It is shown for general k in Section 5.

▶ **Theorem 1.** *Let g be any integer between 1 and n. k-SUM on n numbers can be reduced to $O(kg^{k-1})$ instances of k-SUM on n/g numbers. The reduction uses $O(ng^{k-1})$ additional time and $O(n/g)$ additional words of space.*

Theorem 1 has several interesting applications. First, it leads to more efficient k-SUM algorithms. For example, Gold and Sharir, building on other recent advances, report a deterministic algorithm for 3-SUM that works in both the word-RAM and real-RAM models

[2] The notation $\tilde{O}$ suppresses polylogarithmic factors in n, and O^* suppresses polynomial factors in n.

and which runs in time $O(n^2 \lg\lg(n)/\lg(n))$ [16]. However, this algorithm uses a considerable amount of space to store a table of permutations. Applying Theorem 1 in multiple ways and calling their algorithm, we recover the *same* asymptotic running time but with drastically better space usage:

▶ **Theorem 2.** *There is an $O(n^2 \lg\lg(n)/\lg(n))$ time deterministic algorithm for 3-SUM that stores at $O(\sqrt{\frac{n\lg(n)}{\lg\lg(n)}})$ numbers in memory at point. (An analogous statement holds for 3-SUM over the integers.)*

Theorem 1 also directly leads to a derandomization of Wang's space-efficient algorithm for 3-SUM:

▶ **Theorem 3.** *For all $s \in [0, 1/2]$ there is a deterministic time $O(n^{3-2s})$, algorithm which uses $O(n^s)$ words of space for 3-SUM.*

From Theorem 1 we can also derive a more space-efficient algorithm for 4-SUM, and lift it to a new algorithm for k-SUM:

▶ **Theorem 4.** *For $k \geq 4$, k-SUM is solvable in deterministic $O(n^{k-2+2/(k-3)})$ time and $O(\sqrt{n})$ space in terms of words.*

A more plausible 3-SUM conjecture

A rather popular algorithmic conjecture is the *3-SUM Conjecture* that 3-SUM on n integers requires $n^{2-o(1)}$ time on a word-RAM with $O(\log n)$ bit words. This conjecture has been used to derive conditional lower bounds for a variety of problems [15, 5, 2, 20, 3], and appears to be central to our understanding of lower bounds in low-polynomial time. To refute the conjecture, one could conceivably construct an algorithm that runs in $O(n^{1.99})$ time, but utilizes $\Omega(n^{1.99})$ space in some clever way. Here we consider a seemingly weaker (and thus more plausible) conjecture:

▶ **Conjecture 5** (The Small-Space 3-SUM Conjecture)**.** *On a word-RAM with $O(\log n)$-bit words, there exists an $\epsilon > 0$ such that every algorithm that solves 3-SUM in $O(n^{1/2+\epsilon})$ space must take at least $n^{2-o(1)}$ time.*

This conjecture looks weaker than the original 3-SUM Conjecture, because we only have to prove a quadratic-time lower bound for all algorithms that use slightly more than $\sqrt{n}$ space. Proving time lower bounds is generally much easier when space is severely restricted (e.g. [9, 14, 12, 26, 8]).

Our self-reduction for 3-SUM yields the intriguing consequence that the original 3-SUM Conjecture is *equivalent* to the Small-Space 3-SUM conjecture! That is, the non-existence of a truly subquadratic-time 3-SUM algorithm is equivalent to the non-existence of a truly subquadratic-time $n^{0.51}$-space 3-SUM algorithm, even though the latter appears to be a more plausible lower bound. We prove:

▶ **Theorem 6.** *If 3-SUM is solvable in time $O(n^{2-\epsilon})$ time, then for every $\alpha > 0$ there is a $\delta > 0$ such that 3-SUM is solvable in $O(n^{2-\delta})$ time and space $O(n^{1/2+\alpha})$ in terms of words.*

Theorem 6 is interesting, regardless of the veracity of the 3-SUM conjecture. On the one hand, the theorem reduces the difficulty of proving the 3-SUM Conjecture if it is true, because we only have to rule out small-space sub-quadratic time algorithms. On the other hand, the theorem means that refuting the 3-SUM conjecture immediately implies a truly-subquadratic time algorithm for 3-SUM using small space as well, which would be an algorithmic improvement.

2 Preliminaries

2.1 k-SUM and Selection

We will use the following version of the k-SUM problem:

▶ **Definition 7.** In the *k-SUM problem*, we are given an unsorted list L of n values (over $\mathbb{Z}$ or $\mathbb{R}$) and want to determine if there are $a_1, \ldots, a_k \in L$ such that $\sum_{i=1}^{k} a_i = 0$.

One fundamental case is the 3-SUM problem. Sometimes 3-SUM is presented with three separate lists, which we denote as 3-SUM', but the two are reducible to each other in linear time, and with no impact on space usage.

▶ **Definition 8.** In the *3-SUM problem*, we are given an unsorted list L of n values and want to know if there are $a, b, c \in L$ such that $a + b + c = 0$. In the *3-SUM' problem*, we are given three unsorted lists A, B, and C of values, where $|A| = |B| = |C| = n$, and want to know if there are $a \in A, b \in B, c \in C$ such that $a + b + c = 0$.

As part of our k-SUM algorithms, the classical *Selection Problem* will also arise:

▶ **Definition 9.** In the *s-Select problem*, we are given an unsorted list L of n values and a natural number s, and want to determine the s^{th} smallest value in L.

2.2 Computational Model

As standard when discussing sub-linear space algorithms, the input is provided in read-only memory, and the algorithm works with auxiliary read/write memory which counts towards its space usage.

Computation on Integers. When the input values are integers, we work in the word-RAM model of computation: the machine has a word size w, and we assume all input numbers can be represented with w bits so that they fit in a word. Arithmetic operations $(+, -, *)$ and comparisons on two words are assumed to take $O(1)$ time. Space is counted in terms of the number of words used.

Computation on Reals. When the input values are real numbers, we work in a natural *real-RAM* model of computation, which is often called the *comparison-addition model* (see, for example, [21]). Here, the machine has access to registers that can store arbitrary real numbers; addition of two numbers and comparisons on real numbers take $O(1)$ time. Space is measured in terms of the number of reals stored.

Time-Space Complexity Notation. We say that k-SUM is solvable in $\mathsf{TISP}(T(n), S(n))$ if k-SUM on lists of length n can be solved by a single algorithm running in deterministic $O(T(n))$ time and $O(S(n))$ space simultaneously on the real-RAM (and if the lists contain integers, on the word-RAM).

2.3 Other Prior Work

Baran, Demaine and Patrascu [7] obtained randomized slightly subquadratic time algorithms for Integer 3-SUM in the word-RAM. Grønlund and Pettie [17] studied 3-SUM over the reals, presenting an $O(n^2/(\log n/\log\log n))$ time randomized algorithm, as well as a deterministic algorithm running in $O(n^2/(\log n/\log\log n)^{2/3})$ time. Recently, Gold and Sharir [16]

improved this deterministic running time to $O(n^2/(\log n/\log\log n))$. Abboud, Lewi and Williams [1] showed that Integer k-SUM is W[1]-complete under randomized FPT reductions (and under some plausible derandomization hypotheses, the reductions can be made deterministic). In the linear decision tree model of computation, k-SUM over the reals is known to require $\Omega(n^{\lceil k/2\rceil})$ depth k-linear decision trees [13, 4], but the problem can be solved with $O(n^{k/2}\sqrt{\log n})$ depth $(2k-2)$-linear decision trees [17]. The randomized decision tree complexity was improved by Gold and Sharir [16] to $O(n^{k/2})$.

3 Building Blocks

In this section, we describe two tools we use to obtain our main self-reduction lemma for k-SUM and 3-SUM. The first tool helps us guarantee that we don't have to generate too many subproblems in our reduction; the second will allow us to find these subproblems in a time and space efficient way.

3.1 Domination Lemma

Our deterministic self-reduction for k-SUM will split lists of size n into g sublists of size n/g, then solve subproblems made up of k-tuples of these sublists. Naively, this would generate g^k subproblems to enumerate all k-tuples. In this section, we show that we only need to consider $O(kg^{k-1})$ subproblems.

First, we define a partial ordering on k-tuples on $[n]^k$. For $t, t' \in [n]^k$, we say that $t \prec t'$ if $t[i] < t'[i]$ for all $i = 1, \ldots, k$. (Geometrically, the terminology is that t' *dominates* t.)

▶ **Lemma 10** (Domination Lemma). *Suppose all tuples in a subset $S \subseteq [n]^k$ are incomparable with respect to $\prec$. Then $|S| \leq kn^{k-1}$.*

The Domination Lemma can be seen as an extension of a result in [24] (also used in [11] in a different context) which covers the $k = 3$ case.

Proof. We will give a cover of all elements in $[n]^k$ with few chains under $\prec$. Then by Dilworth's theorem, any set of incomparable elements under $\prec$ can only have one element from each chain.

Take any k-tuple $t \in [n]^k$ such that $t[i] = 1$ for some $i = 1, \ldots, k$. Letting $\ell \in [n]$ be the largest element in t, we define the chain $C(t) = \{t_0, t_1, \ldots, t_{n-\ell}\}$, where each t_j is given by $t_j[i] = t[i] + j$ for all $i = 1, \ldots, k$. Clearly $C(t)$ forms a chain in $[n]^k$ under $\prec$. Moreover these chains cover all elements of $[n]^k$: observe that the tuple t appears in the chain $C(t')$ where $t'[i] = t[i] - \min_j t[j] + 1$ for all $i = 1, \ldots, k$.

The number of chains is exactly the number of k-tuples with a 1 in at least one coordinate. This number is less than k times the number of tuples that have a 1 in dimension i. The number of tuples with a 1 in dimension i is n^{k-1}. Thus, the total number of chains is $\leq kn^{k-1}$. ◀

The Domination Lemma can be applied to show that in any list of numbers, not too many k-SUM subproblems can have k-SUM solutions. In the following, let g divide n for simplicity. Given a list L of n numbers divided into g groups of size n/g, a *subproblem of L* is simply the union of a k-tuple of groups from L. Note that a subproblem contains at most kn/g numbers.

▶ **Corollary 11.** *Given a k-SUM instance L, suppose L is divided into g groups $L_1, \ldots, L_g$ where $|L_i| = n/g$ for all i, and for all $a \in L_i$ and $b \in L_{i+1}$ we have $a \leq b$. Then there are*

$O(k \cdot g^{k-1})$ subproblems L' of L such that the smallest k-sum of L' is less than zero and the largest k-sum of L' is greater than zero. Furthermore, if some subproblem of L has its largest or smallest k-sum equal to 0, then the corresponding k-SUM solution can be found in $O(g^k)$ time.

Proof. We associate each subproblem of L with a corresponding k-tuple $(x_1, \ldots, x_k) \in [g]^k$ corresponding to the k sublists $(L_{x_1}, \ldots, L_{x_k})$ of L.

Let $m[i]$ be the element in position $i \cdot (n/g)$ when L is in sorted order. Consider any subproblem with $\sum_{i=1}^{k} m[x_i] > 0$ (smallest k-sum greater than zero) or $\sum_{i=1}^{k} m[x_i + 1] < 0$ (largest k-sum less than zero). We call such a subproblem *trivial*, since it cannot contain k-SUM solutions.

In $O(g^k)$ time, we can determine whether any subproblem has $\sum_{i=1}^{k} m[x_i] = 0$, and return the corresponding k-SUM solution if this is the case. Otherwise, we can assume that for each subproblem either it is trivial, or $\sum_{i=1}^{k} m[x_i] < 0 < \sum_{i=1}^{k} m[x_i + 1]$.

Consider the set of non-trivial subproblems. Because for all $a \in L_i$ and $b \in L_{i+1}$ we have $a \leq b$, if for two subproblem k-tuples we have $t \prec t'$, then the smallest k-sum of the subproblem t' is at least the largest k-sum of the subproblem t. This implies that at least one of the two subproblems must be trivial. In other words, the set of nontrivial problems corresponds to a set of incomparable k-tuples in $[g]^k$. Applying Lemma 10, the number of nontrivial subproblems is $O(kg^k)$. ◀

3.2 Bucket Retrieval and Space-Efficient Selection

A randomized algorithm for k-SUM can partition a list of numbers by choosing a hash function at random, then loop over the hash function range to partition a given list into smaller buckets. Given a hash and a bucket number, it is easy to retrieve the contents of that bucket by scanning the list.

To derandomize this process, we could try to create small "hash" buckets by grouping the n/g smallest elements together, then the next n/g smallest elements, and so on, *without actually sorting the list*. However, retrieving the contents of a bucket may now be difficult to do with small space: we need to know the smallest and largest elements of a bucket to retrieve its elements, and we may not be able to store all of these extrema. We require an efficient algorithm to compute the largest element of a bucket, given the smallest element and the bucket size.

This problem is equivalent to the *selection problem*, also known as s-SELECT, which asks for the s^{th} smallest element of a list, when we set $s = n/g$. To reduce from our problem to s-SELECT, pretend that every entry less than our smallest element is ∞. (To reduce from s-SELECT to our problem, we can pretend our smallest element is $-\infty$.)

The classic median-of-median algorithm can solve s-SELECT in $O(n)$ time and $O(n)$ space [10]. Since we care about space usage, we provide an algorithm below which has $O(n)$ running time, but uses much less space. This algorithm turns out to be optimal for our purposes, since retrieving the bucket afterwards will already take $O(n)$ time and $O(s)$ space.

▶ **Lemma 12.** *s-SELECT can be solved in $O(n)$ time and $O(s)$ space.*

Proof. The plan is to scan through the elements of the list, inserting them to a data structure D which will allow us to track the smallest s elements. We perform n insertions, then query D to ask for the smallest s elements it contains. To get the claimed algorithm for selection, we give a data structure can handle these operations in $O(1)$ amortized update time and $O(s)$ query time, with a data structure using only $O(s)$ space.

One first attempt might be to build a heap of $s+1$ elements, which throws away the largest element whenever it gets full. Since heaps have logarithmic update time and linear space usage, this results in $O(\log s)$ update time, $O(s)$ query time, and $O(s)$ space.

We can improve the update time by batching when we throw out large elements. Suppose instead we keep an array which can hold up to $2s$ elements. When the array gets full, we throw out the largest s elements. To do this, we first compute the $(s+1)^{th}$ smallest element in the array. This can be done in $O(s)$ time and $O(s)$ space via the classical median-of-medians algorithm. We then do a linear scan of the array, and write all elements strictly less than the median to a new array. To handle ties, we write a copy of the median to the new array, until it has s elements. When we are given our final query, we again throw out large elements so that we only have s elements left, and then return those.

Updates now take amortized constant time: after s updates, we take $O(s)$ time to clear out the large elements. The final query takes $O(s)$ time, since we again need to throw out large elements. The space usage is $O(s)$ since we store up to $2s$ elements, and running median-of-medians takes $O(s)$ space. This completes the proof. ◀

We will call the above algorithm NextGroup. NextGroup takes as input a value v, a natural number s, and a list of numbers L, and outputs the next s elements of L in sorted order after the value v.

4 Subquadratic 3-SUM implies Subquadratic small-space 3-SUM

We will begin by using our building blocks to prove a self reduction for 3-SUM. Then we will show three intriguing consequences of this self reduction. First, the self reduction can be used to show a general theorem that takes subquadratic algorithms for 3-SUM and produces subquadratic time algorithms that run in nearly $\sqrt{n}$ space. Second, we show that algorithms for 3-SUM that are subquadratic by polylog factors can be used to obtain 3-SUM algorithms with the same asymptotic running time and simultaneously small space. Finally, we will prove that the Small-Space 3-SUM conjecture is equivalent to the 3-SUM conjecture.

4.1 3-SUM Self Reduction

We now proceed to solve 3-SUM using our bucket retrieval subroutine. We will use $\max S$ and $\min S$ to refer to the maximum and minimum elements of a list S, respectively.

As anticipated, we split the three arrays into groups of size n/g, and solve 3-SUM on subproblems of this size. Naively there are $O(g^3)$ subproblems to solve, but we use Corollary 11 to argue we only get $O(g^2)$ subproblems.

▶ **Theorem 13** (3-SUM Self-Reduction Theorem). *If 3-SUM is solvable in* $\mathit{TISP}(T(n), S(n))$ *then for any g, 3-SUM can be solved in* $\mathit{TISP}(g^2(n + T(n/g)), n/g + S(n/g))$.

Proof. Consider the following algorithm.

Algorithm 1: 3-SUM Algorithm

```
Set preva = −∞;
for i ∈ [0, g − 1] do
    Set A′ = NextGroup(A, preva, n/g + 1);
    Set prevb = −∞;
    for j ∈ [0, g − 1] do
        Set B′ = NextGroup(B, prevb, n/g + 1);
        Set C′ = NextGroup(C, − max A′ − max B′, n/g + 1);
        while min C′ ≤ − min A′ − min B′ do
            if 3-SUM(A′, B′, C′) returns true then
                return true;
            Set C′ = NextGroup(C, max C′, n/g + 1);
        Set prevb = max B′;
    Set preva = max A′;
return false;
```

Algorithm 1 is correct because we consider all possible elements of C where the sum of elements from A' and B' could land, and the choices of A' and the choices of B' cover all of A and B, respectively. If there are multiple copies of a value in a list we will fail to list all copies only if it already appeared in a previous sublist. This will not affect correctness because the value will have already been analyzed.

It's easy to see that the algorithm calls NextGroup $O(g)$ times for A', $O(g^2)$ times for B'. We claim that we also only call it $O(g^2)$ times for C'. To show this, we want to apply Corollary 11. Unfortunately, the groups of C that we extract don't always line up with our ideal n/g division; since we start at $-\max A' - \max B'$, we may not align at the endpoints of blocks. Fortunately, we've only introduced an extra $O(1)$ possibilities of C' for every (A', B') pair, or $O(g^2)$ extras total. Hence we still only make $O(g^2)$ calls to NextGroup. By Lemma 12, these calls will require $O(ng^2)$ time and $O(n/g)$ space.

Our algorithm also calls the $\mathsf{TISP}(T(n), S(n))$ algorithm for 3-SUM $O(g^2)$ times on instances of size $O(n/g)$, which requires $O(g^2 T(n/g))$ time and $O(S(n/g))$ space.

We have shown Algorithm 1 is correct and has the desired runtime and space usage, so this completes the proof. ◀

4.2 General Theorem for Space Reduction

Our self-reduction for 3-SUM yields the following intriguing consequence: *subquadratic-time algorithms for* 3*-SUM imply subquadratic-time small-space algorithms for* 3*-SUM*. Plugging this connection into known 3-SUM algorithms, we can automatically obtain more space-efficient 3-SUM algorithms for free. From a complexity-theoretic point of view, the consequence is perhaps even more intriguing: it means that the 3-SUM Conjecture is *equivalent* to the statement that there is no subquadratic-time $n^{0.51}$-space 3-SUM algorithm, even though the latter appears to be a more plausible lower bound(!).

We begin by stating our generic space reduction theorem.

▶ **Theorem 14** (3-SUM Space Reduction)**.** *Suppose* 3*-SUM is solvable in* $n^2/f(n)$ *time, where* $1 \le f(n) \le n$*. Then* 3*-SUM is solvable by an algorithm running in* $O(n^2/f(n/g))$ *time and*

$O(n/h)$ space simultaneously, where $g(n), h(n) \in [1, n]$ satisfy the relations

$$g(n) \geq \Omega\left(\sqrt{\frac{n}{f(n/g(n))}}\right) \quad \text{and} \quad h^2 + \frac{n}{f(n/(hg(n/h)))} \leq O\left(\frac{n}{f(n/g(n))}\right).$$

Proof. We will apply our Self-Reduction Theorem for 3-SUM (Theorem 13) in two different ways. First, we will use the self-reduction (and the constraint on $g(n)$) to convert our 3-SUM algorithm into a linear-space algorithm, with a modest increase in running time (if at all). Pushing the linear-space algorithm through the self-reduction once more will reduce the space bound further, without increasing the running time asymptotically (using the constraint on $h(n)$).

Let $T(n) := n^2/f(n)$. Set the parameter $g(n) \geq 1$ to satisfy

$$T(n/g) = \frac{n^2/g^2}{f(n/g)} = O(n); \quad \text{or, equivalently} \quad g = \Omega\left(\sqrt{\frac{n}{f(n/g)}}\right). \tag{1}$$

Assuming g satisfies (1), applying the 3-SUM Self-Reduction (Theorem 13) with $T(n) = S(n)$ and g, we can then solve 3-SUM in

$$\mathsf{TISP}\left(g^2(n + T(n/g)), n/g + T(n/g)\right) = \mathsf{TISP}\left(\frac{n^2}{f(n/g)}, n\right). \tag{2}$$

Now, set new time and space bounds $T(n) := n^2/f(n/g(n))$, $S(n) = n$ from (2). Then, applying the 3-SUM Self-Reduction (Theorem 13) with the new $T(n)$, $S(n)$ and some parameter h, we can then solve 3-SUM in $\mathsf{TISP}(h^2(n + T(n/h)), n/h + S(n/h)) =$

$$\mathsf{TISP}\left(h^2\left(n + \frac{n^2/h^2}{f(n/(hg(n/h)))}\right), n/h\right) \subseteq \mathsf{TISP}\left(\frac{n^2}{f(n/g)}, n/h\right),$$

by our hypothesis on h. ◀

4.3 Space-Efficient Fast 3-SUM

When we apply Theorem 14 directly to known algorithms, we obtain immediate space improvements with negligible loss in running time. Very recently, Gold and Sharir [16] have given a faster 3-SUM algorithm in the real-RAM model, building on the work of Gronlund and Pettie [17]:

▶ **Theorem 15** (Gold and Sharir [16]). *3-SUM can be solved in $O(n^2 \lg\lg(n)/\lg(n))$ time over the reals and integers.*

As discussed in the introduction, their novel approach uses quite a bit of space. Applying Theorem 14, we can reduce the space usage to only $O\left(\sqrt{n\lg(n)/\lg\lg(n)}\right)$, with the same asymptotic running time of Gold and Sharir.

▶ **Corollary 16** (Space-Efficient 3-SUM Algorithm). *3-SUM is in $\mathsf{TISP}\left(n^2\frac{\lg\lg(n)}{\lg(n)}, \sqrt{\frac{n\lg(n)}{\lg\lg(n)}}\right)$.*

Proof. We shall apply Theorem 14. First, set $f(n) := \lg(n)/\lg\lg(n)$, so that 3-SUM is solvable in $O(n^2/f(n))$ time by Theorem 15.

Set $g(n) := \sqrt{\frac{n\lg\lg(n)}{\lg(n)}}$ and $h(n) := \sqrt{\frac{n\lg\lg(n)}{\lg(n)}}$. By our choice of $f(n)$ and basic properties of logarithms, observe that

$$f(n/g) = f(\tilde{O}(\sqrt{n})) = \Theta(f(n)), \tag{3}$$

and furthermore

$$f(n/(hg(n/h))) = f\left(\tilde{O}(\sqrt{n})/\tilde{O}(n^{1/4})\right) = \Theta(f(n)). \tag{4}$$

By (3), we have

$$g = \sqrt{\frac{n\lg\lg(n)}{\lg(n)}} \geq \Omega\left(\sqrt{\frac{n}{f(n/g)}}\right), \text{ so the first constraint of Theorem 14 is satisfied.}$$

Moreover, by (4) we have

$$h^2 + \frac{n}{f(n/(hg(n/h)))} = \frac{n\lg\lg(n)}{\lg(n)} + \frac{n}{\Theta(f(n))}, \text{ which is } O\left(\frac{n}{f(n/g)}\right) \text{ by (3).}$$

Therefore the second constraint of Theorem 14 is also satisfied, and 3-SUM is solvable by an algorithm running in $O(n^2/f(n))$ time and $O\left(\sqrt{nf(n)}\right)$ space simultaneously. ◀

In general, Theorem 14 provides a generic *reduction* from faster 3-SUM algorithms to faster space-efficient 3-SUM algorithms. To illustrate:

▶ **Corollary 17.** *If 3-SUM is solvable in $O(n^2/\lg^a(n))$ time for some constant $a > 0$, then 3-SUM is in $\mathsf{TISP}(n^2/\lg^a(n), \sqrt{n}\lg^{a/2}(n))$.*

Proof. We apply Theorem 14. By assumption we have 3-SUM in $O(n^2/f(n))$ time, where $f(n) = \lg^a n$. Set $g(n) := \sqrt{n}/\lg^{a/2}(n)$, and $h(n) := \sqrt{n}/\lg^{a/2}(n)$. Note that $f(n/g(n)) = \Theta(\lg^a(n))$ and $f(n/(h(n)\cdot g(n/h(n)))) = \Theta(\lg^a(n))$, similar to Corollary 16. Therefore

$$g(n) = \sqrt{n}/\lg^{a/2}(n) \geq \Omega\left(\sqrt{\frac{n}{f(n/g(n))}}\right)$$

and

$$h^2 + \frac{n}{f(n/(hg(n/h)))} \leq O\left(\frac{n}{\lg^a n}\right) \leq O\left(\frac{n}{f(n/g(n))}\right).$$

Hence Theorem 14 applies to these settings of the parameters, and 3-SUM is in $O(n^2/f(n/g)) = O(n^2/\lg^a(n))$ time and $O(n/h) = O(\sqrt{n}\lg^{a/2}(n))$ space. ◀

4.4 The 3-SUM Conjecture and Small Space

Finally, we use the Space Reduction Theorem (Theorem 14) to show that the 3-SUM conjecture is false, then it is also false with respect to small-space algorithms.

▶ **Lemma 18.** *If 3-SUM is in $O(n^{2-\epsilon})$ time for some $\epsilon > 0$, then for every $\alpha > 0$, there is a $\delta > 0$ such that 3-SUM is solvable in $O(n^{2-\delta})$ time and $O(n^{1/2+\alpha})$ space, simultaneously.*

Proof. The proof of Theorem 14 applies the 3-SUM Self Reduction (Theorem 13) twice. We will basically perform the first part of the proof of Theorem 14, but instead of applying the second part of the proof, we have to choose a different setting of parameters, focused on minimizing the space usage instead of preserving running time.

Let $T(n) := n^2/f(n)$ with $f(n) = n^\epsilon$. We first reduce the space usage of the algorithm to linear. To this end, set $g(n) := n^{(1-\epsilon)/(2-\epsilon)}$. Then, applying the 3-SUM Self-Reduction (Theorem 13) with $T(n) = S(n)$ and $g(n)$, we can then solve 3-SUM in

$$\mathsf{TISP}(g^2(n + T(n/g)), n/g + T(n/g)) = \mathsf{TISP}\left(\frac{n^2}{f(n/g)}, n\right) = \mathsf{TISP}\left(\frac{n^2}{n^{\epsilon/(2-\epsilon)}}, n\right).$$

Now reset $f(n) := n^{\epsilon/(2-\epsilon)}$, and reset $g(n) := n^{1/2+\alpha}$ with $\alpha \in (0, 1/2)$. Applying the 3-SUM Self-Reduction (Theorem 13) with $T(n) = n^2/f(n)$, $S(n) = n$, and $g(n)$ as above, we find an algorithm for 3-SUM in

$$\mathsf{TISP}\left(n^{2-2\alpha} + n^{2-(1/2-\alpha)\epsilon/(2-\epsilon)}, n^{1/2+\alpha}\right).$$

Note that for all $\epsilon > 0$ and $\alpha \in (0, 1/2)$, the running time bound is truly subquadratic. Further note that for any $\alpha \geq 1/2$, we only have more space to work with, so we clearly obtain $O(n^{2-\delta})$ time and $O(n^{1/2+\alpha})$ space (for some $\delta > 0$) in that case as well. ◀

This lemma can be applied to show that the 3-SUM Conjecture is equivalent to seemingly much weaker statement:

The Small-Space 3-SUM Conjecture (Conjecture 5). *On a word-RAM with $O(\log n)$-bit words, there exists an $\epsilon > 0$ such that every algorithm that solves 3-SUM in $O(n^{1/2+\epsilon})$ space must take at least $n^{2-o(1)}$ time.*

▶ **Theorem 19.** *The Small-Space 3-SUM Conjecture is equivalent to the 3-SUM Conjecture.*

Proof. It suffices to show that the 3-SUM Conjecture if true implies the Small-Space 3-SUM Conjecture and that the refutation of the 3-SUM Conjecture implies the Small-Space 3-SUM Conjecture. First, we observe that the 3-SUM Conjecture trivially implies the Small-Space 3-SUM Conjecture.

Suppose the 3-SUM Conjecture is false. Then a $O(n^{2-\epsilon})$ time algorithm for 3-SUM exists, and Lemma 18 implies that for every $\alpha > 0$, there is a $\delta > 0$ such that 3-SUM is solvable in $O(n^{2-\delta})$ time and $O(n^{1/2+\alpha})$ space, simultaneously. But this means that for any choice of $\epsilon' > 0$ for the Small-Space 3-SUM Conjecture, we can find a truly-subquadratic 3-SUM algorithm that uses only $O(n^{1/2+\epsilon'/2})$ space. This would falsify the Small-Space 3-SUM Conjecture. ◀

We conclude that, in order to prove the 3-SUM conjecture, it is sufficient to prove that no algorithm can solve 3-SUM in $\mathsf{TISP}(n^{2-\epsilon}, n^{0.51})$ for some $\epsilon > 0$.

5 k-SUM

5.1 k-SUM Self-Reduction

We now generalize from 3-SUM to k-SUM. Again, we plan to split the lists into g groups of size $O(n/g)$. By Corollary 11, we will have only $O(g^{k-1})$ subproblems of size $O(n/g)$. Unlike 3-SUM, where we just used the naive algorithm to solve subproblems, in this section we use a general algorithm; we reduce from k-SUM to itself (albeit on smaller instances).

▶ **Theorem 20.** *Suppose real k-SUM can be solved in $\mathsf{TISP}(T(n), S(n))$. Then for any g, it can also be solved in $\mathsf{TISP}(g^{k-1}(n + T(n/g)), n/g + S(n/g))$.*

Proof. This follows from a generalized analysis of the proof of Theorem 13. We brute force over which groups the first $k-1$ elements are in. We then extract groups where the negative sum of elements from these first $k-1$ groups could land. By Corollary 11 and similar reasoning as before, there are only $O(g^{k-1})$ tuples of blocks. For each tuple, we make a call to NextGroup and to our input k-SUM algorithm on a subproblem of size $O(n/g)$. This gives the desired time and space, completing the proof. ◀

5.2 Applying our k-SUM Self-Reduction

We want to apply the self-reduction on efficient deterministic algorithms. One of the best starting points is the Schroeppel-Shamir 4-SUM algorithm, which we note is actually deterministic and works on reals because it simply uses priority queues and reduces to the classic 2-SUM algorithm, both of which only use comparisons.

▶ **Lemma 21** (From [23]). *Real 4-SUM is solvable in* $\mathit{TISP}(n^2, n)$.

Another useful fact observed by Wang is that an algorithm for k-SUM can be transformed into an algorithm for $(k+1)$-SUM by brute-forcing one element:

▶ **Lemma 22** (From [25]). *If Real k-SUM is solvable in* $\mathit{TISP}(T(n), S(n))$ *then real $k+1$-SUM is solvable in* $\mathit{TISP}(nT(n), S(n)+1)$.

Suppose we want to use our results to derive a linear-space algorithm for k-SUM. We will assume k is a multiple of 4, although Lemma 22 allows us to fill in for the other values of k. By writing down sums of $k/4$ elements, we can transform k-SUM to 4-SUM, yielding a $\mathsf{TISP}(n^{k/2}, n^{k/4})$ algorithm. We can then apply Theorem 20 with $g = n^{(k-4)/k}$ to get a $\mathsf{TISP}(n^{k-3+4/k}, n)$ algorithm. Notice that this algorithm runs significantly faster than $O(n^k)$ time; we get $O(n^{11/2})$ for 8-SUM and $O(n^{28/3})$ for 12-SUM. As a coarse upper bound, we can apply Lemma 22 and round down our savings (to make things cleaner), compensating for k which are not a multiple of 4, we get:

▶ **Corollary 23.** *For $k \geq 4$, k-SUM is solvable in* $\mathit{TISP}(n^{k-3+4/(k-3)}, n)$.

Suppose we wanted to use $O(\sqrt{n})$ space instead. We get smaller subproblems by making more groups; choosing $g = n^{(k-2)/k}$ instead yields a $\mathsf{TISP}(n^{k-2+2/k}, \sqrt{n})$. Similarly applying Lemma 22 and round down our savings to compensate for k which are not a multiple of 4, we get another coarse upper bound:

▶ **Corollary 24.** *For $k \geq 4$, k-SUM is solvable in* $\mathit{TISP}(n^{k-2+2/k}, \sqrt{n})$.

6 Future Work

We would like to extend these results to derandomize other known randomized algorithms for k-SUM. To do that, it seems we require a "deterministic simulation" of the hash functions used in those results. Baran, Demaine, and Patrascu use hashing to get subquadratic algorithms for 3-SUM [7]; Patrascu uses it to reduce 3-SUM to Convolution 3-SUM [20]; Wang uses it to produce a family of linear-space algorithms for k-SUM [25]. Which of these results, if any, can be derandomized?

The hash families involved have three crucial properties: load-balancing (the hash buckets are not "too large"), few subproblems (the number of k-tuples of hash buckets examined is "small"), and few false positives (there are few non-k-SUM solutions mapped to k-tuples of hash buckets examined). Our s-Select algorithm (Lemma 12) and Domination Lemma (Lemma 10) are used to achieve the first two properties, without using randomization. Can the last property also be simulated deterministically? (Note that it's not entirely clear what it would mean to simulate "few false positives" deterministically.) If so, it is likely that all these results can be derandomized efficiently.

References

1 Amir Abboud, Kevin Lewi, and Ryan Williams. Losing weight by gaining edges. In *Algorithms – ESA 2014 – 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 1–12, 2014.

2 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 39–51, 2014.

3 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 434–443, 2014.

4 Nir Ailon and Bernard Chazelle. Lower bounds for linear degeneracy testing. *J. ACM*, 52(2):157–171, 2005.

5 Amihood Amir, Timothy M. Chan, Moshe Lewenstein, and Noa Lewenstein. On hardness of jumbled indexing. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 114–125, 2014.

6 Per Austrin, Petteri Kaski, Mikko Koivisto, and Jussi Määttä. Space-time tradeoffs for subset sum: An improved worst case algorithm. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 45–56, 2013.

7 Ilya Baran, Erik D Demaine, and Mihai Patraşcu. Subquadratic algorithms for 3sum. In *Algorithms and Data Structures*, pages 409–421. Springer, 2005.

8 Paul Beame, Raphaël Clifford, and Widad Machmouchi. Element distinctness, frequency moments, and sliding windows. In *FOCS*, pages 290–299, 2013.

9 Paul Beame, Michael E. Saks, Xiaodong Sun, and Erik Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195, 2003.

10 Manuel Blum, Robert W Floyd, Vaughan Pratt, Ronald L Rivest, and Robert E Tarjan. Time bounds for selection. *Journal of computer and system sciences*, 7(4):448–461, 1973.

11 A. Czumaj and A. Lingas. Finding a heaviest triangle is not harder than matrix multiplication. In *Proc. SODA*, pages 986–994, 2007.

12 Scott Diehl, Dieter van Melkebeek, and Ryan Williams. An improved time-space lower bound for tautologies. *J. Comb. Optim.*, 22(3):325–338, 2011.

13 Jeff Erickson. Lower bounds for linear satisfiability problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1995. San Francisco, California.*, pages 388–395, 1995.

14 Lance Fortnow, Richard J. Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *J. ACM*, 52(6):835–865, 2005.

15 Anka Gajentaan and Mark H Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

16 Omer Gold and Micha Sharir. Improved bounds for 3sum, k-sum, and linear degeneracy. *arXiv preprint arXiv:1512.05279*, 2015.

17 Allan Gronlund and Seth Pettie. Threesomes, degenerates, and love triangles. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 621–630. IEEE, 2014.

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

19 Richard M Karp. *Reducibility among combinatorial problems*. Springer, 1972.

20 Mihai Patrascu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 603–610. ACM, 2010.

21 Seth Pettie and Vijaya Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM J. Comput.*, 34(6):1398–1431, 2005.

22 Mihai Pătraşcu and Ryan Williams. On the possibility of faster sat algorithms. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'10, pages 1065–1075, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

23 Richard Schroeppel and Adi Shamir. A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain np-complete problems. *SIAM journal on Computing*, 10(3):456–464, 1981.

24 Virginia Vassilevska and Ryan Williams. Finding, minimizing, and counting weighted subgraphs. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 455–464. ACM, 2009.

25 Joshua R Wang. Space-efficient randomized algorithms for k-sum. In *Algorithms-ESA 2014*, pages 810–829. Springer, 2014.

26 R. Ryan Williams. Time-space tradeoffs for counting NP solutions modulo integers. *Computational Complexity*, 17(2):179–219, 2008.

Semi-Streaming Algorithms for Annotated Graph Streams*

Justin Thaler†

Yahoo Labs, New York, USA

Abstract

Considerable effort has been devoted to the development of streaming algorithms for analyzing massive graphs. Unfortunately, many results have been negative, establishing that a wide variety of problems require $\Omega(n^2)$ space to solve. One of the few bright spots has been the development of *semi-streaming* algorithms for a handful of graph problems – these algorithms use space $O(n \cdot \text{polylog}(n))$.

In the annotated data streaming model of Chakrabarti et al. [7], a computationally limited client wants to compute some property of a massive input, but lacks the resources to store even a small fraction of the input, and hence cannot perform the desired computation locally. The client therefore accesses a powerful but untrusted service provider, who not only performs the requested computation, but also *proves* that the answer is correct.

We consider the notion of *semi-streaming algorithms for annotated graph streams* (semi-streaming annotation schemes for short). These are protocols in which both the client's space usage and the length of the proof are $O(n \cdot \text{polylog}(n))$. We give evidence that semi-streaming annotation schemes represent a more robust solution concept than does the standard semi-streaming model. On the positive side, we give semi-streaming annotation schemes for two dynamic graph problems that are intractable in the standard model: (exactly) counting triangles, and (exactly) computing maximum matchings. The former scheme answers a question of Cormode [22]. On the negative side, we identify for the first time two natural graph problems (connectivity and bipartiteness in a certain edge update model) that can be solved in the standard semi-streaming model, but cannot be solved by annotation schemes of "sub-semi-streaming" cost. That is, these problems are as hard in the annotations model as they are in the standard model.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases graph streams, stream verification, annotated data streams, probabilistic proof systems

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.59

1 Introduction

The rise of cloud computing has motivated substantial interest in protocols for *verifiable data stream computation*. These protocols allow a computationally weak client (or *verifier*), who lacks the resources to locally store a massive input, to outsource the storage and processing of that input to a powerful but untrusted service provider (or *prover*). Such protocols provide a guarantee that the answer returned by the prover is correct, while allowing the verifier to make only a single streaming pass over the input.

* The full version of this paper is available at `http://arxiv.org/abs/1407.3462`.

† The majority of this work was performed while the author was at the Simons Institute for the Theory of Computing, UC Berkeley. Supported by a Research Fellowship from the Simons Institute for the Theory of Computing.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 59; pp. 59:1–59:14

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

Several recent works have introduced closely related models capturing the above scenario [18, 7, 8, 19, 6, 20, 12, 13, 11]. Collectively, these works have begun to reveal a rich theory, leveraging algebraic techniques developed in the classical theory of interactive proofs [23, 29, 4, 16] to obtain efficient verification protocols for a variety of problems that require linear space in the standard streaming model (sans prover).

The primary point of difference among the various models of verifiable stream computation that have been proposed is the amount of interaction that is permitted between the verifier and prover. The *annotated data streaming* model of Chakrabarti et al. [7] (subsequently studied in [12, 19, 6, 11]) is non-interactive, requiring the correctness proof to consist of just a single message from the prover to the verifier, while other models, such as the *Arthur–Merlin streaming protocols* of Gur and Raz [18, 6] and the *streaming interactive proofs* of Cormode et al. [13, 8] permit the prover and verifier to exchange two or more messages. Our focus in this paper is on the annotated data streaming model of Chakrabarti et al. – owing to their non-interactive nature, protocols in this model possess a number of desirable properties not shared by their interactive counterparts, such as reusability (see Section 2 for details). We are concerned with protocols for problems on graph streams, described below.

Graph Streams. The ubiquity of massive relational data sets (derived, e.g., from the Internet and social networks) has led to detailed studies of data streaming algorithms for analyzing graphs. In this setting, the data stream consists of a sequence of edges, defining a graph G on n nodes, and the goal is to compute various properties of G (is G connected? How many triangles does G contain?). Unfortunately, many results on graph streaming have been negative: essentially any graph problem of the slightest practical interest requires $\Omega(n)$ space to solve in the standard streaming model, and many require $\Omega(n^2)$ space even to approximate. Due to their prohibitive cost in the standard streaming model, many basic graph problems are ripe for outsourcing.

One of the few success stories in the study of graph streams has been the identification of the *semi-streaming* model as something of a "sweet spot" for streaming algorithms[26, 24]. The semi-streaming model is characterized by an $O(n \cdot \operatorname{polylog} n)$ space restriction, i.e., space proportional to the number of nodes rather than the number of edges. For dense graphs this represents considerably less space than that required to store the entire graph. It has long been known that problems like connectivity and bipartiteness possess semi-streaming algorithms when the stream consists only of edge insertions, with no deletions. Recently, semi-streaming algorithms have been provided for these and other problems even for dynamic graph streams, which contain edge deletions as well as insertions [1, 2, 14]. We direct the interested reader toward the recent survey of McGregor [25] on graph stream algorithms.

In this work, we consider the notion of *semi-streaming annotation schemes* for graph problems. Here, the term "scheme" refers to a protocol in the annotated data streaming model. A scheme's total cost is defined to be the sum of the verifier's space usage (referred to as the *space cost* of the scheme) and the length of the proof (referred to as the scheme's *help cost*). A scheme is said to be semi-streaming if its total cost is $O(n \cdot \operatorname{polylog} n)$.

We give evidence that semi-streaming annotation schemes represent a substantially more robust solution concept (i.e., a "sweeter spot") for graph problems than does the standard semi-streaming model. First, we give novel semi-streaming annotation schemes for two challenging dynamic graph problems, counting triangles and maximum matching, that require $\Omega(n^2)$ space in the standard streaming model. The total cost of these schemes is provably optimal up to a logarithmic factor. Second, we show that two canonical problems that do possess semi-streaming algorithms in the standard streaming model (connectivity

Table 1 Comparison of our new scheme for TRIANGLES to prior work.

Reference	TRIANGLES Scheme Costs (help cost, space cost)	Total Cost Achieved
[7]	$(n^2 \log n, \log n)$	$O(n^2 \log n)$
[7]	$(x \log n, y \log n)$ for any $x \cdot y \geq n^3$	$O(n^{3/2} \log n)$
Theorem 1	$(n \log n, n \log n)$	$O(n \log n)$

and bipartiteness in a certain edge update model) are *just as hard* in the annotations model. Formally, we show that any scheme for these problems with space cost $O(n^{1-\delta})$ requires a proof of length $\Omega(n^{1+\delta})$ for any $\delta > 0$. Thus, for these problems, giving a streaming algorithm access to an untrusted prover does not allow for a significant reduction in cost relative to what is achievable without a prover. This gives further evidence for the robustness of semi-streaming annotation schemes as a solution concept: while several fundamental problems that cannot be solved by standard semi-streaming algorithms can be solved by semi-streaming annotation schemes, there are problems that do have semi-streaming solutions in the standard model that cannot be solved by schemes of sub-semi-streaming cost.

1.1 Summary of Contributions and Techniques

Throughout this informal overview, n will denote the number of nodes in the graph defined by the data stream, and m the number of edges. To avoid boundary cases in the statement of our lower bounds, we assume that the help cost of any scheme is always at least 1 bit.

1.1.1 New Semi-Streaming Annotation Schemes

Prior work has given semi-streaming annotation schemes for two graph problems that require $\Omega(n^2)$ space in the standard semi-streaming model: bipartite perfect matching [7], and shortest s-t path in graphs of polylogarithmic diameter [12]. As discussed above, we give semi-streaming schemes for two more challenging problems: maximum matching (MAXMATCHING) and counting triangles (TRIANGLES). Both schemes apply to dynamic graph streams. We begin by describing our result for TRIANGLES.

▶ **Theorem 1** (Informal Version of Theorem 4). *There is a scheme for* TRIANGLES *with total cost $O(n \log n)$. Every scheme requires the product of the space and help costs to be $\Omega(n^2)$, and hence has total cost $\Omega(n)$.*

Theorem 1 affirmatively answers a question of Cormode [22], resolves the Merlin-Arthur communication complexity of the problem up to a logarithmic factor, and improves over the best previous bound of $O(n^{3/2} \log n)$, due to [7] (see Table 1 for a comparison).

As is the case for essentially all non-trivial protocols for verifiable stream computation, the scheme of Theorem 1 uses algebraic techniques related to the famous *sum-check protocol* of Lund et al. [23] from the classical theory of interactive proofs. Yet, our scheme deviates in a significant way from all earlier annotated data stream and interactive proof protocols [15, 8, 7, 29]. Roughly speaking, in previous protocols, the verifier's updates to her memory state were commutative, in the sense that reordering the stream tokens would not change the final state reached by the verifier. However, our new verifier is inherently non-commutative: her update to her state at time i depends on her actual state at time i, and reordering the stream tokens can change the final state reached by the verifier. The full version of the paper contains further discussion of this point.

Table 2 Comparison of our new scheme for MaxMatching to prior work.

Reference	MaxMatching Scheme Costs (help cost, space cost)	Total Cost Achieved
[12]	$(m \log n, \log n)$	$O(m \log n)$
Theorem 5	$(n \log n, n \log n)$	$O(n \log n)$

▶ **Theorem 2** (Informal Version of Theorem 5). *There is a scheme for* MaxMatching *with total cost $O(n \log n)$. Every scheme for* MaxMatching *requires the product of the space and help costs to be $\Omega(n^2)$, and hence has total cost $\Omega(n)$.*

Our scheme combines the Tutte-Berge formula with algebraic techniques to allow the prover to establish matching upper and lower bounds on the size of a maximum matching in the input graph. Schemes for maximum matching had previously been studied by Cormode et al. [12], but this prior work only gave schemes with help cost proportional to the number of edges, which is $\Omega(n^2)$ in dense graphs (like us, Cormode et al. exploited the Tutte-Berge formula, but did not do so in a way that achieved help cost sublinear in the input size). Prior work had also given a scheme achieving optimal tradeoffs between help and space costs for *bipartite perfect matching* [7, Theorem 7.5] – our scheme for MaxMatching can be seen as a broad generalization of [7, Theorem 7.5].

1.1.2 New Lower Bounds

On the other hand, we identify, for the first time, natural graph problems that possess standard semi-streaming algorithms, but in a formal sense are just as hard in the annotations model as they are in the standard streaming model. The problems that we consider are connectivity and bipartiteness in a certain edge update model that we call the XOR update model. In this update model, the stream $\langle e_1, \ldots, e_m \rangle$ is a sequence of edges from $[n] \times [n]$, which define a graph $G = (V, E)$ via: $e \in E \iff |\{i : e_i = e\}| = 1 \mod 2$. Intuitively, each stream update e_i is interpreted as changing the status of edge e_i: if it is currently in the graph, then the update causes e_i to be deleted; otherwise e_i is inserted. Our lower bound holds for schemes for connectivity and bipartiteness in the XOR update model, even under the promise that $e_1, \ldots, e_{m-n}$ are all unique (hence, all but the last n stream updates correspond to edge insertions), and the last n updates are all incident to a single node.

▶ **Theorem 3** (Informal Version of Corollary 7). *Consider any scheme for* Connectivity *or* Bipartiteness *in the XOR update model with help cost c_a and and space cost c_v. Then $(c_a + n) \cdot c_v \geq n^2$, even under the promise that the first $m - n$ stream updates are all unique, and the last n stream updates are all incident to a single node. In particular, the total cost of any annotation scheme for these problems is $\Omega(n)$.*

Both connectivity and bipartiteness in the XOR update model possess standard semi-streaming algorithms [1].[1] Hence, Theorem 3 implies that the total cost of any annotation scheme is at most a polylogarithmic factor smaller than the problems' space complexity in the standard streaming model. Like all prior work establishing lower bounds on the cost of

[1] The algorithms of [1] are described in the turnstile update model, in which each stream update explicitly specifies whether to insert or delete a (copy of) an edge. However, these algorithms are easily modified to apply to the XOR update model as well. In brief, these algorithms have L_0-sampling algorithms at their core. Existing L_0-samplers are in turn built on top of sparse recovery algorithms (see, e.g., [10]), and many sparse recovery algorithms can be implemented in the XOR update model directly (see, e.g., [17]).

protocols for verifiable stream computation, our lower bounds are established using notions of *Merlin-Arthur communication complexity* [7, 12, 19, 8, 18].

Prior to this work, only one other problem was known to be as hard (up to logarithmic factors) in the annotations model as in the standard streaming model [6, Corollary 3.3]. The problem considered in [6, Corollary 3.3] was an "exponentially sparse" variant of the classic INDEX problem, in which the data stream consists of a vector $x \in \{0,1\}^n$ promised to have Hamming weight $O(\log n)$, followed by an index $i \in [n]$, and the goal is to output the value x_i. Connectivity and bipartiteness are arguably more natural problems, and are qualitatively different, as we now explain.

On an informal level, the reason the exponentially sparse INDEX problem is hard in the annotations model is that any "useful" annotation must at least specify a single index into the vector x, which requires $\log n$ bits of annotation. And since x is exponentially sparse, $\log n$ is actually equal (up to a constant factor) to the space complexity of a standard streaming algorithm for the problem. In our view, BIPARTITENESS and CONNECTIVITY are hard in the annotations for a different reason – roughly speaking, any useful annotation for these problems must at least specify, for each node u, the side of the bipartition or the connected component in which u resides.

Overview of the Proof. Our proof of Theorem 3 works by specifying a reduction from the INDEX problem on inputs of length n^2, for which a lower bound of $\Omega(n^2)$ on the product of the help and space costs of any annotation scheme was established in [7], to CONNECTIVITY and BIPARTITENESS on graphs with n nodes and $\Theta(n^2)$ edges.

Notice that in the standard (sans prover) streaming model, the INDEX problem on n^2 variables is strictly harder than connectivity and bipartiteness problems on graphs with n nodes, as the former requires $\Omega(n^2)$ space, while the latter two problems require only $O(n \cdot \text{polylog}(n))$ space. Yet Theorem 3 establishes that in the annotations model, all three problems are of essentially equivalent difficulty (in particular, schemes of total cost $\tilde{O}(n)$ are necessary and sufficient to solve all three problems). To establish such a result, it is necessary to use a reduction that is specifically tailored to the annotations model, in the sense that the reduction must not apply in the standard streaming model (since INDEX and CONNECTIVITY are not of equivalent difficulty in the standard setting). Namely, in our reduction from INDEX to connectivity, the prover *helps the verifier* transform an instance of the INDEX problem into a CONNECTIVITY instance. This "help" consists of $\Theta(n)$ bits, and this is why our lower bound is of the form $(c_a + n) \cdot c_v \geq n^2$. This is in contrast to prior lower bounds, which, with the exception of [6, Corollary 3.3], were of the form $c_a \cdot c_v = \Omega(C)$ for some quantity C.

1.2 Other Related Work

As discussed above, several recent papers [11, 8, 13, 19, 20, 18, 6, 7, 12] have all studied annotated data streams and closely related models for verifiable stream computation. Refinements and implementations [11, 31] have demonstrated genuine practicality for many of the protocols developed in this line of work. Protocols for verifiable stream computation have also been studied in the cryptography community [9, 27]. These works only require security against cheating provers that run in polynomial time, whereas we require security to hold even against computationally unbounded provers. In exchange for the weaker security guarantee, these protocols may achieve properties that are unattainable in our information-theoretic setting. For example, some of these protocols achieve a stronger form of reusability than we do (see Section 2 for our definition of reusability) – they remain secure for many uses even if the prover learns all of the verifier's accept/reject decisions. The work of Chung et al. [9]

uses fully homomorphic encryption (FHE), which remains far from practical at this time. Papamanthou et al. [27] avoid the use of FHE, but handle only much simpler queries (e.g., point queries and range search) than the graph problems we consider here.

2 Models of Streaming Computation

Our presentation of data streaming models closely follows Chakrabarti et al. [6]. Recall that a (standard) data stream algorithm computes a function f of an input sequence $\mathbf{x} \in \mathcal{U}^m$, where m is the number of stream updates, and $\mathcal{U}$ is some data universe. The algorithm has only sequential access to $\mathbf{x}$, uses a limited amount of space, and has access to a random string. The function f may or may not be Boolean. An annotated data stream algorithm, or a *scheme*, is a pair $\mathcal{A} = (\mathfrak{h}, V)$, consisting of a help function $\mathfrak{h} : \mathcal{U}^m \times \{0,1\}^* \to \{0,1\}^*$ used by a *prover* and a data stream algorithm run by a *verifier*, V. The prover provides $\mathfrak{h}(\mathbf{x})$ as annotation to be read by the verifier. We think of $\mathfrak{h}$ as being decomposed into $(\mathfrak{h}_1, \ldots, \mathfrak{h}_m)$, where the function $\mathfrak{h}_i : \mathcal{U}^m \to \{0,1\}^*$ specifies the annotation supplied after the arrival of the ith token x_i. That is, $\mathfrak{h}$ acts on $\mathbf{x}$ to create an *annotated stream* $\mathbf{x}^{\mathfrak{h}}$ defined as follows:

$$\mathbf{x}^{\mathfrak{h}} := (x_1, \mathfrak{h}_1(\mathbf{x}), x_2, \mathfrak{h}_2(\mathbf{x}), \ldots, x_m, \mathfrak{h}_m(\mathbf{x})) .$$

Note that this is a stream over $\mathcal{U} \cup \{0,1\}$, of length $m + \sum_i |\mathfrak{h}_i(\mathbf{x})|$. The streaming verifier, who has access to a (private) random string r, then processes this annotated stream, eventually giving an output $\text{out}^V(\mathbf{x}^{\mathfrak{h}}, r)$.

We say a scheme is *online* if each function $\mathfrak{h}_i$ depends only on $(x_1, \ldots, x_i)$. The scheme $\mathcal{A} = (\mathfrak{h}, V)$ is said to be δ_s-sound and δ_c-complete for the function F if the following hold:

1. For all $\mathbf{x} \in \mathcal{U}^m$, we have $\Pr_r[\text{out}^V(\mathbf{x}^{\mathfrak{h}}, r) \neq F(\mathbf{x})] \leq \delta_c$.
2. For all $\mathbf{x} \in \mathcal{U}^m$, $\mathfrak{h}' = (\mathfrak{h}'_1, \mathfrak{h}'_2, \ldots, \mathfrak{h}'_m) \in (\{0,1\}^*)^m$, we have $\Pr_r[\text{out}^V(\mathbf{x}^{\mathfrak{h}'}, r) \notin \{F(\mathbf{x})\} \cup \{\perp\}] \leq \delta_s$.

An output of "$\perp$" indicates that the verifier rejects the prover's claims in trying to convince the verifier to output a particular value for $F(\mathbf{x})$. We define $\text{err}(\mathcal{A})$ to be the minimum value of $\max\{\delta_s, \delta_c\}$ such that the above conditions are satisfied. We define the *annotation length* $\text{hc}(\mathcal{A}) = \max_{\mathbf{x}} \sum_i |\mathfrak{h}_i(\mathbf{x})|$, the total size of the prover's communications, and the *verification space cost* $\text{vc}(\mathcal{A})$ to be the space used by the verifier. We say that $\mathcal{A}$ is an online (c_a, c_v) scheme if $\text{hc}(\mathcal{A}) = O(c_a)$, $\text{vc}(\mathcal{A}) = O(c_v)$, and $\text{err}(\mathcal{A}) \leq \frac{1}{3}$ (the constant 1/3 is arbitrary and chosen by convention).

Chakrabarti et al. [7] also define the notion of a *prescient* scheme, which is the same as an online scheme, except the annotation at any time i is allowed to depend on data which the verifier has not seen yet. Prescient schemes have the undesirable property that the prover may need to "see into the future" to convince the verifier to produce the correct output. Note that even though our Triangles and MaxMatching protocols are online, they are optimal up to logarithmic factors *even among prescient schemes* (see Theorems 4 and 5).

While the annotated data streams model allows the prover to interleave the annotation with the stream, in all of the schemes we present in this paper, all of the annotation comes at the end of the stream. This property avoids any need for fine-grained coordination between the annotation and the stream, and permits the prover to send the annotation as a single email attachment, or post it to a website. We clarify that the *lower bounds* for Connectivity and Bipartiteness that we establish in Section 4 apply to any online scheme, even those which interleave the annotation with the stream.

The schemes we present in this work permit a natural form of *reusability*: if the verifier wants to compute a function f on a given dataset $\mathbf{x}$, the verifier can receive $f(\mathbf{x})$ (with

a correctness proof), and check the validity of the proof using a "secret state" that she computed while observing the stream $\mathbf{x}$. Further updates to the stream $\mathbf{x}$ can then occur, yielding a (longer) stream $\mathbf{x}'$, and the verifier can update her secret in a streaming fashion. The verifier may then receive the answer $f(\mathbf{x}')$ (with a correctness proof) on the updated dataset, and check its correctness using the updated secret state. The probability that the verifier gets fooled into outputting an incorrect answer on even a single query grows only linearly with the number of times the prover sends the verifier an answer. Such reusability is not possible with interactive solutions [13, 8, 20], which require the verifier to reveal information about r over the course of the protocol.

3 Upper Bounds

Graph Streams in the Strict Turnstile Model. The annotation schemes of this section apply to graph streams in the strict turnstile update model. In this model, a data stream σ consists of a sequence of undirected edges, accompanied by (signed) multiplicities: $\langle (e_1, \Delta_1), \ldots, (e_m, \Delta_m) \rangle$. Each edge $e_i \in [n] \times [n]$, and each $\Delta_i \in \mathbb{Z}$. An update (e_i, Δ_i) with $\Delta_i > 0$ is interpreted as an insertion of Δ_i copies of edge e_i into graph G. If $\Delta_i < 0$, the update is interpreted as a deletion of Δ_i copies of edge e_i. It is assumed that at the end of the stream, no edge has been deleted more times than it has been inserted (all of our protocols work even if this property does not hold at *intermediate* time steps, as long as the property holds after the final stream update has been processed).[2] When analyzing the time costs of our schemes, we assume that any addition or multiplication in a finite field of size poly(n) takes one unit of time.

3.1 A Semi-Streaming Scheme for Counting Triangles

In the TRIANGLESproblem, the goal is to determine the number of unordered triples of distinct vertices (u, v, z) such that edges (u, v), (v, z), and (z, u) all appear in G. More generally, if these edges appear with respective multiplicities M_1, M_2, and M_3, we view triple (u, v, z) as contributing $M_1 \cdot M_2 \cdot M_3$ triangles to the total count.[3] Computing the number of triangles is a well-studied problem [3] and there has been considerable interest in designing algorithms in a variety of models including the data stream model [5, 28], MapReduce [30], and the quantum query model [21]. One motivation is the study of social networks where important statistics such as the clustering coefficient and transitivity coefficient are based on the number of triangles. Understanding the complexity of counting triangles captures the ability of a model to perform a non-trivial correlation within large graphs. Chakrabarti et al. [7] gave two annotated data streaming protocols for this problem. The first protocol had help cost $O(n^2 \log n)$, and space cost $O(\log n)$. The second protocol achieved help cost $O(x \log n)$ and space cost $O(y \log n)$ for any x, y such that $x \cdot y \geq n^3$. In particular, by setting $x = y = n^{3/2}$, the second protocol of Chakrabarti et al. ensured that both help cost and space cost equaled $O\left(n^{3/2} \log n\right)$. Cormode [22] asked whether it is possible to achieve an annotated data streaming protocol in which both the help cost and space cost are $\tilde{O}(n)$. We

[2] We do not consider edges with negative weights at the end of the stream because it is unclear what is the most meaningful way to define problems like TRIANGLES and MAXMATCHING in this setting. See Footnotes 3 and 4.

[3] The TRIANGLESscheme of Theorem 3.1 gives meaningful results even if the M_i's may be negative: a triangle with an odd (even) number of edges of negative multiplicity contributes a negative (positive) number to the total triangle count.

answer this question in the affirmative.

▶ **Theorem 4** (Formal Statement of Theorem 1)**.** *Assume there is an a priori upper bound $B \leq \text{poly}(n)$ on the multiplicity of any edge in G. There is an online scheme for* Triangles *with space and help costs $O(n \log n)$. Every scheme (online or prescient) requires the product of the space and help costs to be $\Omega(n^2)$, and hence total cost $\Omega(n)$, even for $B = 1$, and even if G is promised to have exactly 0 or 1 triangles.*

Discussion. Before proving Theorem 4, it is instructive to consider a simple *interactive* streaming verification protocol for Triangles(described in detail in the full version of the paper). At a high level, the interactive protocol applies the sum-check protocol of Lund et al. [23] to a suitably defined multivariate polynomial h. The space and communication costs of this protocol are comparable to that of Theorem 4; the advantage of Theorem 4 over this simple interactive solution is that Theorem 4 gives a protocol that is *non-interactive*, and comes with the associated reusability benefits described in Section 2. Theorem 4 below effectively removes the interaction from the simple interactive protocol as follows. We identify a *univariate* polynomial $g(Z)$ of degree $O(n)$ such that the number of triangles in G equals $\sum_{z \in [n]} g(Z)$. Moreover, we show that the verifier can evaluate $g(r)$ for any point $r \in \mathbb{F}$ in space $O(n \log |\mathbb{F}|)$ with a single streaming pass over the input. It follows that applying the sum-check protocol to g yields a scheme with costs claimed in Theorem 4. The polynomial g that we identify is defined as a sum of m constituent polynomials, one per stream update.

Proof of Theorem 4. The lower bound was proved in [7, Theorem 7.1]. Details of the upper bound follow. Let G_i denote the graph defined by the first i stream updates $\langle (e_1, \Delta_1), \ldots, (e_i, \Delta_i) \rangle$, and let $E_i \colon [n] \times [n] \to \mathbb{Z}$ denote the function that outputs the multiplicity of the edge (u, v) in graph G_{i-1}. On edge update $e_i = (u_i, v_i)$, notice that the number of triangles that e_i *completes* in E_i is precisely $\Delta_i \cdot \sum_{z \in [n]} E_i(u_i, z) E_i(v_i, z)$. Thus, the total number of triangles in the graph $G = G_m$ is precisely $\sum_{i \leq m} \Delta_i \sum_{z \in [n]} E_i(u_i, z) E_i(v_i, z)$. Let $\mathbb{F}$ denote a field of prime order $6(B \cdot n)^3 \leq |\mathbb{F}| \leq 12(B \cdot n)^3$, and let $\tilde{E}_i(X, Y)$ denote the unique polynomial over $\mathbb{F}$ of degree at most n in each variable X, Y such that $\tilde{E}_i(u, v) = E_i(u, v)$ for all $(u, v) \in [n] \times [n]$. Then the number of triangles in G equals

$$\sum_{i \leq m} \Delta_i \sum_{z \in [n]} E_i(u_i, z) E_i(v_i, z) = \sum_{i \leq m} \Delta_i \sum_{z \in [n]} \tilde{E}_i(u_i, z) \tilde{E}_i(v_i, z) = \sum_{z \in [n]} \sum_{i \leq m} \Delta_i \tilde{E}_i(u_i, z) \tilde{E}_i(v_i, z). \tag{1}$$

In turn, the right hand side of Equation (1) can be written as $\sum_{z \in [n]} g(z)$, where g denotes the *univariate* polynomial defined via: $g(Z) = \sum_{i \leq m} \Delta_i \cdot \tilde{E}_i(u_i, Z) \tilde{E}_i(v_i, Z)$. Notice $g(Z)$ is a univariate polynomial of degree at most $2n$. Our scheme proceeds as follows.

Prover's computation. At the end of the stream, the prover sends a univariate polynomial $s(Z)$ of degree at most $2n$, where $s(Z)$ is claimed to equal $g(Z)$. Notice that since $s(Z)$ has degree at most $2n$, $s(Z)$ can be specified by sending its values on all inputs in $\{0, \ldots, 2n\}$ – this requires help cost $O(n \log |\mathbb{F}|) = O(n \log n)$.

Verifier's computation. At the start of the stream, the verifier picks a random field element $r \in \mathbb{F}$, and keeps the value of r secret from the prover. We will show below that the verifier can evaluate $g(r)$ with a single streaming pass over the input, using space $O(n \log n)$. The

verifier checks whether $s(r) = g(r)$. If this check fails, the verifier halts and rejects. If the check passes, the verifier outputs $\sum_{z \in [n]} s(z)$ as the correct answer.

We now explain how the verifier can evaluate $g(r)$ with a single streaming pass over the input. The high-level idea is as follows. $g(r)$ is defined as a sum of m terms, where the ith term equals $\Delta_i \cdot \tilde{E}_i(u_i, r)\tilde{E}_i(v_i, r)$. For each $u \in [n]$, we will show how the verifier can incrementally maintain the quantity $\tilde{E}_i(u, r)$ at all times i. The verifier will maintain all n of these quantities, resulting in a total space cost of $O(n \log |\mathbb{F}|) = O(n \log n)$. With these quantities in hand, it is straightforward for the verifier to incrementally maintain the sum $\sum_{j \leq i} \Delta_j \cdot \tilde{E}_j(u_j, r)\tilde{E}_j(v_j, r)$ at all times i: upon the ith stream update, the verifier simply adds $\Delta_i \cdot \tilde{E}_i(u_i, r) \cdot \tilde{E}_i(v_i, r)$ to the running sum.

To maintain the quantity $\tilde{E}_i(u, r)$, we begin by writing the bivariate polynomial $\tilde{E}_i(X, Y)$ in a convenient form. Given a pair $(u, v) \in [n] \times [n]$, let $\tilde{\delta}_{(u,v)}$ denote the following (Lagrange) polynomial: $\tilde{\delta}_{(u,v)}(X, Y) = \left(\frac{\prod_{1 \leq u' \leq n : u' \neq u}(X - u')}{\prod_{1 \leq u' \leq n : u' \neq u}(u - u')}\right)\left(\frac{\prod_{1 \leq v' \leq n : v' \neq v}(Y - v')}{\prod_{1 \leq v' \leq n : v' \neq v}(v - v')}\right)$. Notice that $\tilde{\delta}_{(u,v)}$ evaluates to 1 on input (u, v), and evaluates to 0 on all other inputs $(x, y) \in [n] \times [n]$. Thus, we may write $\tilde{E}_i(X, Y) = \sum_{j \leq i} \tilde{\delta}_{(u_j, v_j)}(X, Y)$. In particular, for each node $u \in [n]$,

$$\tilde{E}_i(u, r) = \tilde{E}_{i-1}(u, r) + \tilde{\delta}_{(u_i, v_i)}(u, r) + \tilde{\delta}_{(v_i, u_i)}(u, r).$$

Thus, the verifier can incrementally maintain the quantity $\tilde{E}_i(u, r)$ in a streaming manner using space $O(\log |\mathbb{F}|)$: while processing the ith stream update, the verifier simply adds $\tilde{\delta}_{(u_i, v_i)}(u, r) + \tilde{\delta}_{(v_i, u_i)}(u, r)$ to the running sum tracking $\tilde{E}_i(u, r)$.

Completeness. It is evident that if the prover sends the true polynomial $g(Z)$, then the verifier's check will pass, and the verifier will output the correct number of triangles.

Soundness. If the prover sends a polynomial $s(Z) \neq g(Z)$, then with probability at least $1 - 2n/|\mathbb{F}| \geq 1 - 1/(3n^2)$ over the verifier's random choice of $r \in \mathbb{F}$, it will hold that $s(r) \neq g(r)$. Hence, with probability at least $1 - 1/(3n^2) \geq 2/3$, the verifier's check will fail. ◀

Several remarks regarding Theorem 4 are in order.

- **Verifier Time.** The verifier in the protocol of Theorem 4 can process each stream update in constant time as follows. On stream update $e_i = (u_i, v_i)$, the verifier must add $\tilde{\delta}_{(u_i, v_i)}(u, r) + \tilde{\delta}_{(v_i, u_i)}(u, r)$ to each of the $\tilde{E}_i(u, r)$ values. However, it is straightforward to check that $\tilde{\delta}_{(u_i, v_i)}(u, r) = 0$ for all $u \neq u_i$, so the verifier need only update two quantities at time i: $\tilde{E}_i(u_i, r)$ and $\tilde{E}_i(v_i, r)$. We explain how both of these updates can be computed in constant time. It can be seen that $\tilde{\delta}_{(u_i, v_i)}(u_i, r) = \frac{\prod_{1 \leq v' \leq n : v' \neq v_i}(r - v')}{\prod_{1 \leq v' \leq n : v' \neq v_i}(v_i - v')}$. The right hand side of this equation can be computed in $O(1)$ time if the verifier maintains a pre-computed lookup table consisting of $O(n)$ field elements. Specifically, for each $v \in [n]$, it suffices for the verifier to maintain the quantities $q_1(v) := \prod_{1 \leq v' \leq n : v' \neq v}(r - v')$ and $q_2(v) = \left(\prod_{1 \leq v' \leq n : v' \neq v}(v - v')\right)^{-1}$. All $O(n)$ of these quantities can be computed in pre-processing in total time $O(n \log n)$, where the $\log n$ term is due to the time required to compute a multiplicative inverse in the field $\mathbb{F}$. Indeed, $q_1(1)$ and $q_2(1)$ can be computed naively in $O(n)$ time, and then for any $v > 1$, $q_1(v)$ and $q_2(v)$ can be computed in $O(\log n)$ time from $q_1(v-1)$ and $q_2(v-1)$ via the identities $q_1(v) = q_1(v-1) \cdot (r - v)^{-1} \cdot (r - v + 1)$ and $q_2(v) = q_2(v-1) \cdot (v-1)^{-1} \cdot (v - 1 - n)$.

Table 3 Statement of Time Costs For Our TrianglesScheme (Theorem 4).

Verifier Pre-Processing Time	Verifier Time Per Stream Update	Verifier Time to Process Proof	Prover Total Time
$O(n \log n)$	$O(1)$	$O(n)$	$O(m \cdot n)$

Finally, the verifier can process the proof in time $O(n)$. Recall that the proof consists of the values $s(x)$ for $x \in \{0, \ldots, 2n\}$, and the verifier simply needs to compute $\sum_{1 \leq x \leq n} s(x)$ as well as $s(r)$. The first quantity can trivially be computed in time $O(n)$, and the second can be computed in time $O(n)$ as well using standard techniques (see, e.g., [11]).

- **Prover Time.** The honest prover in the protocol of Theorem 4 can be implemented to run in time $O(m \cdot n)$. Indeed, the honest prover needs to evaluate $g(x)$ for $O(n)$ points $x \in \mathbb{F}$, and we have explained above how $g(x)$ can be computed in $O(m)$ time (in fact, in $O(1)$ time per stream update). This is comparable to the naive triangle counting algorithm that, for each edge and node combination, tests whether the two edges incident on the edge and node exist in the graph.
  The time costs for both the prover and verifier are summarized in Table 3.
- **MA communication.** Theorem 4 implies that the (online) MA communication complexity of counting triangles is $O(n \log n)$ (see Section 4.1 for the definition of the (online) MA communication model). This essentially matches an $\Omega(n)$ lower bound on the (even non-online) MA communication complexity of the problem, proved by Chakrabarti et al. [7] via a standard reduction to Set-Disjointness, and answers a question of Cormode [22].
- **Extensions: Counting Structures Other Than Triangles.** Let H be a graph on k vertices. It is possible to extend the protocol of Theorem 4 to count the number of occurrences of H as a subgraph of G. The protocol requires $k - 2$ rounds, and its help and space costs are $O(k^3 n \log n)$ and $O(kn \log n)$. Details are deferred to the full version.

3.2 A Semi-Streaming Scheme for Maximum Matching

We give a semi-streaming scheme for the MaxMatching problem in general graphs.[4] Due to space constraints, the proof of Theorem 5 is deferred to the full version.

▶ **Theorem 5** (Formal Version of Theorem 2). *Assume there is an a priori upper bound $B \leq \text{poly}(n)$ on the multiplicity of any edge in G. There is an online scheme for* MaxMatching *of total cost $O(B \cdot n \log n)$. Every scheme for* MaxMatching *(online or prescient) requires the product of the space and help costs to be $\Omega(n^2)$, and hence total cost $\Omega(n)$, even for $B = 1$.*

4 Lower Bounds for Connectivity and Bipartiteness

In this section, we establish our lower bounds on the cost of online schemes for Connectivity and Bipartiteness in the XOR update models. Like almost all previous lower bounds for data stream computations, our lower bounds use reductions from problems in communication complexity. To model the prover in a scheme, the appropriate communication setting is Merlin-Arthur communication, which we now introduce.

[4] It is possible to modify our scheme to give meaningful answers on graphs with edges of negative multiplicity. Specifically, the modified scheme can treat edges of negative multiplicity as having strictly positive multiplicity. We omit the details for brevity.

4.1 Merlin-Arthur Communication

Consider a communication game involving three parties, named Alice, Bob, and Merlin. Alice holds an input $x \in \mathcal{X}$, Bob and input $y \in \mathcal{Y}$, and Merlin is omniscient (he sees both x and y) but untrusted. Alice and Bob's goal is to compute $f(x,y)$ for some agreed upon function $f : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$. In an MA communication protocol $\mathcal{P}$, Merlin first broadcasts a message m_M to both Alice and Bob. Alice and Bob then engage in a randomized communication protocol, before outputting a single bit. To clarify, Merlin does not learn the randomness that Alice and Bob use until after sending the message $m_M(x,y)$. For each input (x,y), the protocol $\mathcal{P}$ defines a game between Merlin, Alice, and Bob, in which Merlin's goal is to make Alice and Bob output 1. We define $\mathbf{Val}^{\mathcal{P}}(x,y)$ to be Merlin's probability of winning with optimal play. Given a Boolean function f, we say that $\mathcal{P}$ computes f if, for all (x,y) we have (1) $f(x,y)=0 \Longrightarrow \mathbf{Val}^{\mathcal{P}}(x,y) \leq 1/3$, and (2) $f(x,y)=1 \Longrightarrow \mathbf{Val}^{\mathcal{P}}(x,y) \geq 2/3$. We refer to the Property (1) as *soundness* and Property (2) as *completeness*.

The *help cost*, or $\mathrm{hc}(\mathcal{P})$, of $\mathcal{P}$ is $\max_{(x,y)} |m_M(x,y)|$, i.e., the maximum length of Merlin's message in bits. The *verification cost*, or $\mathrm{vc}(\mathcal{P})$, of $\mathcal{P}$ is the maximum number of bits that Alice and Bob exchange, where the maximum is taken over all inputs (x,y), all possible Merlin messages m_M, and all choices of Alice and Bob's randomness. The *total cost* of $\mathcal{P}$ is the sum of the help and verification costs of $\mathcal{P}$.

In an *online* MA (OMA) communication protocol, neither Merlin nor Bob can talk to Alice. Given any online scheme for a function f, we naturally obtain an OMA protocol $\mathcal{P}$ for the communication problem in which Alice holds a prefix of a stream, Bob holds a suffix, and the goal is to evaluate f on the concatenated stream $x \circ y$. Hence, if we establish lower bounds on the help and verification costs of any OMA protocol for f, an equivalent lower bound on the help and space costs of any online scheme for f follows.

In this section, we establish lower bounds on the help and verification costs of any OMA protocol for the Disconnectivity and Bipartiteness problems in the XOR update model. More precisely, we consider the communication problems Disconnectivity$^{\mathrm{cc}}$ and Bipartiteness$^{\mathrm{cc}}$ in which Alice holds the first $m-n$ tuples in a graph stream in the XOR update model, Bob holds the length n tuples, and the output function evaluates to 1 if and only if the resulting graph is disconnected or bipartite, respectively.[5]

4.2 The Lower Bound

▶ **Theorem 6.** *Consider any OMA protocol $\mathcal{P}$ for* Disconnectivity$^{\mathrm{cc}}$ *or* Bipartiteness$^{\mathrm{cc}}$. *Then it holds that* $(\mathrm{hc}(\mathcal{P})+n)\cdot \mathrm{vc}(\mathcal{P}) = \Omega(n^2)$. *This holds even under the promise that the first $m-n$ stream updates (i.e., Alice's input) are all unique, and the last n stream updates (i.e., Bob's input) are all incident to a single node. In particular, the total cost of $\mathcal{P}$ is $\Omega(n)$.*

Proof. We prove the lower bound Disconnectivity$^{\mathrm{cc}}$problem. The proof for Bipartiteness$^{\mathrm{cc}}$is similar, and is deferred to the full version of the paper. Let $\mathcal{P}$ denote any OMA protocol for Disconnectivity$^{\mathrm{cc}}$that works on graphs with $n+1$ nodes, under the promise

[5] The reason that we consider Disconnectivity$^{\mathrm{cc}}$ rather than Connectivity$^{\mathrm{cc}}$ is the asymmetric way that inputs in $F^{-1}(0)$ and $F^{-1}(1)$ in the definition of OMA communication complexity. Recall that the OMA communication problem for a decision problem F requires only that if $F(x)=1$ then there is some prover that will cause the verifier to accept with high probability, and if $F(x)=0$ then there is no such prover. (By contrast, our definition of a scheme for a function F requires there to be a convincing proof of the value of $F(x)$ for all values $F(x)$.) Hence, the OMA communication complexities of Disconnectivity$^{\mathrm{cc}}$ and Connectivity$^{\mathrm{cc}}$ may not be equal, and indeed our lower bound argument applies only to Disconnectivity$^{\mathrm{cc}}$.

described in the theorem hypothesis. As discussed in the outline of Section 1.1.2, our proof will use a reduction from the INDEX problem on $\binom{n}{2}$ inputs. In this problem, Alice's input consists of a bitstring $\mathbf{x}$ of length $\binom{n}{2}$, Bob's input is an index $i^* \in [\binom{n}{2}]$, and the goal is to output $\mathbf{x}_{i^*}$. It was established in [7] that any OMA protocol $\mathcal{Q}$ for INDEX on $\binom{n}{2}$ inputs requires $\text{hc}(\mathcal{Q}) \cdot \text{vc}(\mathcal{Q}) = \Omega(n^2)$. We show how to use $\mathcal{P}$ to construct a protocol $\mathcal{Q}$ for INDEX on $\binom{n}{2}$ inputs with $\text{hc}(\mathcal{Q}) = n + \text{hc}(\mathcal{P})$ and $\text{vc}(\mathcal{Q}) = \text{vc}(\mathcal{P})$. It will then follow from the lower bound for INDEX that $(\text{hc}(\mathcal{P}) + n) \cdot \text{vc}(\mathcal{P}) \geq n^2$.

Description of $\mathcal{Q}$. In $\mathcal{Q}$, Alice interprets her input $\mathbf{x} \in \{0,1\}^{\binom{n}{2}}$ as an undirected graph G_1 with n nodes as follows. She associates each index $i \in \binom{n}{2}$ with a unique edge (u_i, v_i) out of the set of all $\binom{n}{2}$ possible edges that could appear in G_1. Alice also adds to G_1 a special node v^*, and connects v^* to every other node in G_1. Denote the resulting graph on $n+1$ nodes by G_2. Notice that G_2 is always connected, as every node is connected to v^* by design.

Likewise, Bob interprets his input $i^* \in [\binom{n}{2}]$ as an edge (u_{i^*}, v_{i^*}). Clearly, determining whether $\mathbf{x}_{i^*} = 1$ is equivalent to determining whether edge (u_{i^*}, v_{i^*}) appears in Alice's graph G_2. Merlin sends Bob a list L claimed to equal all edges incident to node u_{i^*} in G_2. This requires only n bits of "help", since there are only n nodes to which u_{i^*} might be adjacent. Bob treats L as his input to the DISCONNECTIVITY$^{\text{cc}}$problem.

Alice, Bob, and Merlin now run the DISCONNECTIVITY$^{\text{cc}}$ protocol $\mathcal{P}$ (with Alice's input equal to G_2 and Bob's input equal to L). Bob outputs 1 iff the protocol $\mathcal{P}$ outputs 1, and L contains the edge (u_{i^*}, v_{i^*}).

Costs of $\mathcal{Q}$. The help cost of $\mathcal{Q}$ is equal to $n + \text{hc}(\mathcal{P})$, since the honest Merlin sends Bob the list L, and then behaves as he would in the protocol $\mathcal{P}$. The verification cost of $\mathcal{Q}$ is just $\text{vc}(\mathcal{Q})$, since the only message Alice sends to Bob is the message she would send in $\mathcal{P}$.

Completeness and Soundness of $\mathcal{Q}$. Let G_3 denote the graph obtained from G_2 by XORing all the edges in the list L. Let $I(u_{i^*})$ denote the set of edges incident to u_{i^*} in G_3. We claim that G_3 is disconnected if and only if L is equal to $I(u_{i^*})$. For the first direction, suppose that L is equal to $I(u_{i^*})$. Then by XORing the edges in G_3 with the edges in L, every edge incident to node u_{i^*} is deleted from the graph. Hence, u_{i^*} is an isolated vertex in G_3, implying that G_3 is disconnected. For the second direction, suppose that L is not equal to $I(u_{i^*})$. Let (u_{i^*}, v) denote an edge in $L \setminus I(u_{i^*})$. Then (u_{i^*}, v) is in G_3. Moreover, v is adjacent to node v^*, as are all nodes in G_3 other than u_{i^*}. Hence G_3 is connected.

To finish the proof of completeness of $\mathcal{Q}$, note that if $\mathbf{x}_{i^*} = 1$, then the edge (u_{i^*}, v_{i^*}) is in G_3. If Merlin sends $L = I(u_{i^*})$, then G_3 will be disconnected, and by the the completeness of $\mathcal{P}$, Merlin can convince Bob that G_3 is disconnected with probability at least 2/3. In this event, Bob will output 1, because (u_{i^*}, v_{i^*}) will be in the list L.

To finish the proof of soundness of $\mathcal{Q}$, note that if $\mathbf{x}_{i^*} = 0$, then the edge (u_{i^*}, v_{i^*}) is not in G_3. Hence, if Merlin sends $L = I(u_{i^*})$, then Bob will reject automatically, because (u_{i^*}, v_{i^*}) will not be in the list L. On the other hand, if Merlin sends a list L that is *not* equal to $I(u_{i^*})$, then G_3 will be connected. By the the soundness of $\mathcal{P}$, Merlin can convince Bob that G_3 is disconnected with probability at most 1/3. Hence, Bob will output 1 in $\mathcal{Q}$ with probability at most 1/3, completing the proof for DISCONNECTIVITY$^{\text{cc}}$. ◀

Because any online scheme for CONNECTIVITY or BIPARTITENESS can be simulated by an OMA communication protocol, we obtain the following corollary.

▶ **Corollary 7** (Formal Version of Theorem 3). *Consider any online scheme for* Connectivity *or* Bipartiteness *in the XOR update model with help cost c_a and and space cost c_v. Then $(c_a + n) \cdot c_v \geq n^2$, even under the promise that the first $m - n$ stream updates are all unique, and the last n stream updates are all incident to a single node. In particular, the total cost of any annotation scheme is $\Omega(n)$.*

Acknowledgments. The author is grateful to Graham Cormode, Amit Chakrabarti, Andrew McGregor, and Suresh Venkatasubramanian for many valuable discussions regarding this work.

References

1. Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *SODA*, pages 459–467, 2012. URL: http://dl.acm.org/citation.cfm?id=2095156.
2. Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *PODS*, pages 5–14, 2012. doi:10.1145/2213556.2213560.
3. Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.
4. László Babai. Trading group theory for randomness. In *STOC*, pages 421–429, 1985. doi:10.1145/22145.22192.
5. Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*, pages 623–632, 2002. URL: http://dl.acm.org/citation.cfm?id=545381.545464.
6. Amit Chakrabarti, Graham Cormode, Navin Goyal, and Justin Thaler. Annotations for sparse data streams. In *SODA*, pages 687–706, 2014. doi:10.1137/1.9781611973402.52.
7. Amit Chakrabarti, Graham Cormode, Andrew McGregor, and Justin Thaler. Annotations in data streams. *ACM Trans. Algorithms*, 11(1):7:1–7:30, 2014. doi:10.1145/2636924.
8. Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and arthur-merlin communication. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 217–243, 2015. doi:10.4230/LIPIcs.CCC.2015.217.
9. Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In *CRYPTO*, pages 151–168, 2011. doi:10.1007/978-3-642-22792-9_9.
10. Graham Cormode and Donatella Firmani. A unifying framework for l_0-sampling algorithms. *Distributed and Parallel Databases*, 32(3):315–335, 2014. doi:10.1007/s10619-013-7131-9.
11. Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *ITCS*, pages 90–112, 2012. doi:10.1145/2090236.2090245.
12. Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Streaming graph computations with a helpful advisor. *Algorithmica*, 65(2):409–442, 2013. doi:10.1007/s00453-011-9598-y.
13. Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *PVLDB*, 5(1):25–36, 2011. URL: http://www.vldb.org/pvldb/vol5/p025_grahamcormode_vldb2012.pdf.
14. Ashish Goel, Michael Kapralov, and Ian Post. Single pass sparsification in the streaming model with edge deletions. *CoRR*, abs/1203.4900, 2012. URL: http://arxiv.org/abs/1203.4900.

15 Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC'08, pages 113–122, New York, NY, USA, 2008. ACM. doi:10.1145/1374376.1374396.

16 Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. doi:10.1137/0218012.

17 Michael T. Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *Allerton*, pages 792–799, 2011. doi:10.1109/Allerton.2011.6120248.

18 Tom Gur and Ran Raz. Arthur-Merlin streaming complexity. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming: Part I*, ICALP'13, Berlin, Heidelberg, 2013. Springer-Verlag.

19 Hartmut Klauck and Ved Prakash. Streaming computations with a loquacious prover. In *ITCS*, pages 305–320, 2013. doi:10.1145/2422436.2422471.

20 Hartmut Klauck and Ved Prakash. An improved interactive streaming algorithm for the distinct elements problem. In *ICALP (1)*, pages 919–930, 2014. doi:10.1007/978-3-662-43948-7_76.

21 Troy Lee, Frédéric Magniez, and Miklos Santha. Improved quantum query algorithms for triangle finding and associativity testing. In *SODA*, pages 1486–1502, 2013. doi:10.1137/1.9781611973105.107.

22 List of open problems in sublinear algorithms: Problem 47. http://sublinear.info/47.

23 Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39:859–868, October 1992. doi:10.1145/146585.146605.

24 Andrew McGregor. Graph mining on streams. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 1271–1275. Springer US, 2009. doi:10.1007/978-0-387-39940-9_184.

25 Andrew McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, May 2014. doi:10.1145/2627692.2627694.

26 S. Muthukrishnan. *Data Streams: Algorithms And Applications*. Foundations and Trends in Theoretical Computer Science. Now Publishers Incorporated, 2005.

27 Charalampos Papamanthou, Elaine Shi, Roberto Tamassia, and Ke Yi. Streaming authenticated data structures. In *EUROCRYPT*, pages 353–370, 2013. doi:10.1007/978-3-642-38348-9_22.

28 A. Pavan, Kanat Tangwongsan, Srikanta Tirthapura, and Kun-Lung Wu. Counting and sampling triangles from a graph stream. *Proc. VLDB Endow.*, 6(14):1870–1881, September 2013. doi:10.14778/2556549.2556569.

29 Adi Shamir. IP = PSPACE. *J. ACM*, 39:869–877, October 1992. doi:10.1145/146585.146609.

30 Siddharth Suri and Sergei Vassilvitskii. Counting triangles and the curse of the last reducer. In *WWW*, pages 607–614, 2011. doi:10.1145/1963405.1963491.

31 Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO (2)*, pages 71–89, 2013. doi:10.1007/978-3-642-40084-1_5.

Randomized Query Complexity of Sabotaged and Composed Functions*

Shalev Ben-David[1] **and Robin Kothari**[2]

1 **Massachusetts Institute of Technology, Cambridge, MA, USA**
shalev@mit.edu
2 **Massachusetts Institute of Technology, Cambridge, MA, USA**
rkothari@mit.edu

Abstract

We study the composition question for bounded-error randomized query complexity: Is $R(f \circ g) = \Omega(R(f)R(g))$? We show that inserting a simple function h, whose query complexity is only $\Theta(\log R(g))$, in between f and g allows us to prove $R(f \circ h \circ g) = \Omega(R(f)R(h)R(g))$.

We prove this using a new lower bound measure for randomized query complexity we call randomized sabotage complexity, $\mathrm{RS}(f)$. Randomized sabotage complexity has several desirable properties, such as a perfect composition theorem, $\mathrm{RS}(f \circ g) \geq \mathrm{RS}(f)\,\mathrm{RS}(g)$, and a composition theorem with randomized query complexity, $R(f \circ g) = \Omega(R(f)\,\mathrm{RS}(g))$. It is also a quadratically tight lower bound for total functions and can be quadratically superior to the partition bound, the best known general lower bound for randomized query complexity.

Using this technique we also show implications for lifting theorems in communication complexity. We show that a general lifting theorem from zero-error randomized query to communication complexity implies a similar result for bounded-error algorithms for all total functions.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases Randomized query complexity, decision tree complexity, composition theorem, partition bound, lifting theorem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.60

1 Introduction

1.1 Composition theorems

A basic structural question that can be asked in any model of computation is whether there can be savings in complexity when computing the same function on several independent inputs. We say a direct sum theorem holds in a model of computation if solving a problem on n independent inputs requires roughly n times the resources needed to solve one instance. A direct sum theorem is known to hold for deterministic and randomized query complexity [9], and two-player refereed games [17], is known to fail for circuit size [16], and remains open for deterministic communication complexity [11].

More generally, instead of merely outputting the n answers, we could compute another function of these n answers. If f is an n-bit Boolean function and g is an m-bit Boolean function, we define the composed function $f \circ g$ to be an nm-bit Boolean function such that $f \circ g(x_1, \ldots, x_n) = f(g(x_1), \ldots, g(x_n))$, where each x_i is an m-bit string. The composition question now asks if there can be significant savings in computing $f \circ g$ compared to simply

* This work was partially supported by ARO grant number W911NF-12-1-0486.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 60; pp. 60:1–60:14

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

running the best algorithm for f and using the best algorithm for g to evaluate the input bits needed to compute f. If we let f be the identity function on n bits that just outputs all its inputs, we recover the direct sum problem.

Composition theorems are harder to prove and are known for only a handful of models, such as deterministic and quantum query complexity. Proving this for randomized query complexity remains a major open problem. More precisely, if $D(f)$, $R(f)$, and $Q(f)$ denote the deterministic, randomized, and quantum query complexities of f, then we know for all partial Boolean functions f and g, $D(f \circ g) = D(f)D(g)$ [18, 15] and $Q(f \circ g) = \Theta(Q(f)Q(g))$ [14, 12]. (Such theorems often fail for functions with non-Boolean output, and hence we only consider functions with Boolean output in this paper.) In contrast, in the randomized setting we have only the upper bound $R(f \circ g) = O(R(f)R(g) \log R(f))$.

▶ **Open Problem 1.** Does it hold that $R(f \circ g) = \Omega(R(f)R(g))$ for all Boolean f and g?

In this paper we prove something close to a composition theorem for randomized query complexity. While we cannot rule out the possibility of synergistic savings in computing $f \circ g$, we show that a composition theorem does hold if we insert a small gadget in between f and g to obfuscate the output of g. Our gadget is "small" in the sense that its randomized (and even deterministic) query complexity is $\Theta(\log R(g))$. Specifically we choose the index function, which on an input of size $k+2^k$ interprets the first k bits as an address into the next 2^k bits and outputs the bit stored at that address. The index function's query complexity is $k+1$ and we choose $k = \Theta(\log R(g))$ in our construction.

▶ **Theorem 1.** *Let f and g be partial Boolean functions and let* IND *be the index function with* $R(\text{IND}) = \Theta(\log R(g))$. *Then* $R(f \circ \text{IND} \circ g) = \Omega(R(f)R(\text{IND})R(g)) = \Omega(R(f)R(g) \log R(g))$.

Theorem 1 can be used instead of a true composition theorem in many applications. For example, recently a composition theorem for randomized query complexity was needed in the special case when f is the AND function [2, 5] or when g is the AND function [1]. Our composition theorem would suffice for both applications.

We prove Theorem 1 by introducing a new lower bound technique for randomized query complexity. This is not surprising since the composition theorems for deterministic and quantum query complexities are also proved using powerful lower bound techniques for these models, namely the adversary argument and the general adversary bound [7] respectively.

1.2 Sabotage complexity

To describe the new lower bound technique, consider the problem of computing a Boolean function f on an input $x \in \{0,1\}^n$ in the query model. In this model we have access to an oracle, which when queried with an index $i \in [n]$ responds with $x_i \in \{0,1\}$. Now imagine a saboteur damages the oracle making some of the input bits unreadable; for these input bits the oracle simply responds with a $*$. We can now view the oracle as storing a string $p \in \{0,1,*\}^n$ as opposed to a string $x \in \{0,1\}^n$. Although it is not possible to determine the true input x from the oracle string p, it may still be possible to compute $f(x)$ if all input strings consistent with p evaluate to the same f value. On the other hand, it is not possible to compute $f(x)$ if p is consistent with a 0-input and a 1-input to f, and we call such a string $p \in \{0,1,*\}^n$ a *sabotaged input*. For example, let f be the OR function that computes the logical OR of its bits. Then $p = 00*0$ is a sabotaged input since it is consistent with the 0-input 0000 and the 1-input 0010. However, $p = 01*0$ is not a sabotaged input since it is only consistent with 1-inputs to f.

Now consider a new problem in which the input is promised to be sabotaged (with respect to a function f) and our job is to find the location of a $*$. Intuitively, any algorithm that solves the original problem f when run on a sabotaged input must discover at least one $*$, since otherwise it would answer the same on 0- and 1-inputs consistent with the sabotaged input. This can be formalized and leads to a lower bound measure for several models of computation, including deterministic, randomized, and quantum query complexity.

As it stands the problem of finding a $*$ in a sabotaged input has multiple valid outputs, as the location of any star in the input is a valid output. For convenience we define a decision version of this problem by imaging there are two saboteurs, one of whom has sabotaged our input. The first saboteur, Asterix, replaces input bits with an asterisk ($*$) and the second, Obelix, uses an obelisk (†). Promised that the input has been sabotaged exclusively by one of Asterix or Obelix, our job is to identify the saboteur. This is now a decision problem since there are only two valid outputs. We call this decision problem f_{sab}, the *sabotage problem* associated with f.

We now define lower bound measures for various models using f_{sab}. For example, we can define the *deterministic sabotage complexity of* f as $\text{DS}(f) := D(f_{\text{sab}})$ and in fact, $\text{DS}(f) = D(f)$ as we show in the full version of this paper. We could define the *randomized sabotage complexity of* f as $R(f_{\text{sab}})$, but instead we define it as $\text{RS}(f) := R_0(f_{\text{sab}})$, where R_0 denotes zero-error randomized query complexity, since $R(f_{\text{sab}})$ and $R_0(f_{\text{sab}})$ are equal up to constant factors. Besides lower bounding $R(f)$, $\text{RS}(f)$ has the following desirable properties:

1. (Perfect composition) For all f and g, $\text{RS}(f \circ g) \geq \text{RS}(f)\,\text{RS}(g)$ (Theorem 15)
2. (Composition with R) For all f and g, $R(f \circ g) = \Omega(R(f)\,\text{RS}(g))$ (Theorem 17)
3. (Quadratically tight) For all total f, $R(f) = O(\text{RS}(f)^2 \log \text{RS}(f))$ (Theorem 28)
4. (Superior to $\text{prt}(f)$) There exists a total f with $\text{RS}(f) \geq \text{prt}(f)^{2-o(1)}$ (Theorem 26)

Here $\text{prt}(f)$ denotes the partition bound [8, 10], which subsumes most other lower bound techniques such as approximate polynomial degree and randomized certificate complexity. In fact, we are unaware of any total function f for which $\text{RS}(f) = o(R(f))$, leaving open the intriguing possibility that this lower bound technique is tight.

1.3 Lifting theorems

Using randomized sabotage complexity we are also able to show a relationship between lifting theorems in communication complexity. A lifting theorem relates the query complexity of a function f with the communication complexity of a related function created from f. Recently, Göös, Pitassi, and Watson [6] showed that there is a communication problem G with communication complexity $\Theta(\log n)$ such that for any function f on n bits, $D^{\text{cc}}(f \circ G) = \Omega(D(f) \log n)$, where D^{cc} denotes deterministic communication complexity.

Analogous lifting theorems are known for some complexity measures, but no such theorem is known for either zero-error randomized or bounded-error randomized query complexity. Our second result shows that a lifting theorem for zero-error randomized query complexity implies one for bounded-error randomized query complexity for total functions. We use R_0^{cc} and R^{cc} to denote zero-error and bounded-error communication complexity respectively.

▶ **Theorem 2.** *Let G be the communication gadget from [6] with $D^{\text{cc}}(G) = \Theta(\log n)$. If it holds that for all n-bit (possibly partial) functions f, $R_0^{\text{cc}}(f \circ G) = \Omega(R_0(f)/\operatorname{polylog} n)$, then it holds that for all n-bit total Boolean functions f, $R^{\text{cc}}(f \circ G) = \Omega(R(f)/\operatorname{polylog} n)$.*

Proving a lifting theorem for bounded-error randomized query complexity remains an important open problem, and would imply super-quadratic separations between randomized and quantum communication complexity [1], and a nearly quadratic separation between

randomized communication complexity and partition number [2]. Our result shows that it is sufficient to prove a lifting theorem for zero-error randomized protocols instead.

2 Preliminaries

We now define some basic notions in query complexity. Note that all the functions in this paper have Boolean output. In the model of query complexity, we wish to compute an n-bit Boolean function f on an input x given query access to the bits of x. The function f may be total, i.e., $f : \{0,1\}^n \to \{0,1\}$, or partial, which means it is defined only on a subset of $\{0,1\}^n$, which we denote by $\mathrm{Dom}(f)$. The goal is to output $f(x)$ using as few queries to the bits of x as possible. The number of queries used by the best possible deterministic algorithm (over worst-case choice of x) is denoted $D(f)$.

A randomized algorithm is a probability distribution over deterministic algorithms. The worst-case cost of a randomized algorithm is the worst-case number of queries made by the algorithm on any input x. The expected cost of the algorithm is the expected number of queries made by the algorithm maximized over all inputs x. A randomized algorithm has error at most ϵ if it outputs $f(x)$ on every x with probability at least $1-\epsilon$.

We use $R_\epsilon(f)$ to denote the worst-case cost of the best randomized algorithm that computes f with error ϵ. Similarly, we use $\overline{R}_\epsilon$ to denote the expected cost of the best randomized algorithm that computes f with error ϵ. When ϵ is unspecified it is taken to be $\epsilon = 1/3$. Thus $R(f)$ denotes the bounded-error randomized query complexity of f. Finally, we also define zero-error randomized query complexity, which is $\overline{R}_0(f)$, which we also denote by $R_0(f)$ to be consistent with the literature. For precise definitions of these measures as well as the definition of quantum query complexity $Q(f)$, see [3]. We also need two simple properties of randomized algorithms, which we prove in the full version of this paper.

▶ **Lemma 3.** *If A is a randomized algorithm that uses T expected queries and finds a certificate with probability $1-\epsilon$, then repeating A when it fails turns it into a zero-error algorithm that uses at most $T/(1-\epsilon)$ expected queries.*

▶ **Lemma 4.** *Let f be a partial function and A be an ϵ-error randomized algorithm for f that uses at most T expected queries. For $x, y \in \mathrm{Dom}(f)$ if $f(x) \neq f(y)$ then when A is run on x, it must query an entry on which x differs from y with probability at least $1-2\epsilon$.*

3 Sabotage complexity

Given a (partial or total) n-bit Boolean function f, let $P_f \subseteq \{0,1,*\}^n$ be the set of all partial assignments of f that are consistent with both a 0-input and a 1-input; that is, for each $p \in P_f$, there exist $x, y \in \mathrm{Dom}(f)$ such that $f(x) \neq f(y)$ and $x_i = y_i = p_i$ whenever $p_i \neq *$. Let $P_f^\dagger \subseteq \{0,1,\dagger\}^n$ be the same as P_f, except using the symbol $\dagger$ instead of $*$. Observe that P_f and $P_f^\dagger$ are disjoint. Let $Q_f = P_f \cup P_f^\dagger \subseteq \{0,1,*,\dagger\}^n$. We then define f_{sab} as follows.

▶ **Definition 5.** Let f be an n-bit partial function. We define $f_{\mathrm{sab}} : Q_f \to \{0,1\}$ as $f_{\mathrm{sab}}(q) = 0$ if $q \in P_f$ and $f_{\mathrm{sab}}(q) = 1$ if $q \in P_f^\dagger$.

See Section 1.2 for more discussion and motivation for this definition. Now that we have defined f_{sab}, we can define sabotage complexity for various models.

▶ **Definition 6.** Let f be a partial function. Then $\mathrm{DS}(f) := D(f_{\mathrm{sab}})$ and $\mathrm{RS}(f) := R_0(f_{\mathrm{sab}})$.

We will primarily focus on $\mathrm{RS}(f)$ in this work. To justify defining $\mathrm{RS}(f)$ as $R_0(f_{\mathrm{sab}})$ instead of $R(f_{\mathrm{sab}})$, we now show these definitions are equivalent up to constant factors.

▶ **Theorem 7.** *Let f be a partial function. Then $R_0(f_{\mathrm{sab}}) \geq \overline{R}_\epsilon(f_{\mathrm{sab}}) \geq (1-2\epsilon)R_0(f_{\mathrm{sab}})$.*

Proof. The first inequality follows trivially. For the second, let $x \in Q_f$ be any valid input to f_{sab}. Let x' be the input x with asterisks replaced with obelisks and vice versa. Then since $f_{\mathrm{sab}}(x) \neq f_{\mathrm{sab}}(x')$, by Lemma 4 any ϵ-error randomized algorithm that solves f_{sab} must find a position on which x and x' differ with probability at least $1-2\epsilon$. The positions at which they differ are either asterisks or obelisks. Since x was an arbitrary input, the algorithm must always find an asterisk or obelisk with probability at least $1-2\epsilon$. Since finding an asterisk or obelisk is a certificate for f_{sab}, by Lemma 3, we get a zero-error algorithm for f_{sab} that uses $\overline{R}_\epsilon(f_{\mathrm{sab}})/(1-2\epsilon)$ expected queries. Thus $R_0(f_{\mathrm{sab}}) \leq \overline{R}_\epsilon(f_{\mathrm{sab}})/(1-2\epsilon)$, as desired. ◀

Finally, we prove that $\mathrm{RS}(f)$ is indeed a lower bound on $R(f)$, i.e., $R(f) = \Omega(\mathrm{RS}(f))$.

▶ **Theorem 8.** *Let f be an n-bit partial function. Then $R_\epsilon(f) \geq \overline{R}_\epsilon(f) \geq (1-2\epsilon)\,\mathrm{RS}(f)$.*

Proof. Let A be a randomized algorithm for f that uses $\overline{R}_\epsilon(f)$ randomized queries and outputs the correct answer on every input in $\mathrm{Dom}(f)$ with probability at least $1-\epsilon$. Now fix a sabotaged input x, and let p be the probability that A finds a $*$ or † when run on x. Let q be the probability that A outputs 0 if it doesn't find a $*$ or † when run on x. Let x_0 and x_1 be inputs consistent with x such that $f(x_0) = 0$ and $f(x_1) = 1$. Then A outputs 0 on x_1 with probability at least $q(1-p)$, and A outputs 1 on x_0 with probability at least $(1-q)(1-p)$. These are both errors, so we have $q(1-p) \leq \epsilon$ and $(1-q)(1-p) \leq \epsilon$. Summing them gives $1-p \leq 2\epsilon$, or $p \geq 1-2\epsilon$.

This means A finds a $*$ entry within $\overline{R}_\epsilon(f)$ expected queries with probability at least $1-2\epsilon$. By Lemma 3, we get $\frac{1}{1-2\epsilon}\overline{R}_\epsilon(f) \geq \mathrm{RS}(f)$, or $\overline{R}_\epsilon(f) \geq (1-2\epsilon)\,\mathrm{RS}(f)$. ◀

We also define a variant of RS where the number of asterisks (or obelisks) is exactly one. Specifically, let $U \subseteq \{0,1,*,\dagger\}^n$ be the set of all partial assignments with exactly one $*$ or †.

▶ **Definition 9.** Let f be a partial function. We define f_{usab} as the restriction of f_{sab} to U, the set of strings with only one asterisk or obelisk. I.e., f_{usab} has domain $Q_f \cap U$, but is equal to f_{sab} on its domain. We then define $\mathrm{RS}_1(f) := R_0(f_{\mathrm{usab}})$. If $Q_f \cap U$ is empty, we define $\mathrm{RS}_1(f) := 0$.

The measure RS_1 will play a key role in our lifting result in Section 6. Since f_{usab} is a restriction of f_{sab} to a promise, it is clear that its zero-error randomized query complexity is smaller, so $\mathrm{RS}_1(f) \leq \mathrm{RS}(f)$. Another interesting property is the following theorem, which says $\mathrm{RS}_1(f)$ equals $\mathrm{RS}(f)$ for total functions. In other words, when f is total, we may assume without loss of generality that its sabotaged version has only one asterisk or obelisk.

▶ **Theorem 10.** *If f is a total function, then $\mathrm{RS}_1(f) = \mathrm{RS}(f)$.*

Proof. We showed that $\mathrm{RS}(f) \geq \mathrm{RS}_1(f)$. To show $\mathrm{RS}_1(f) \geq \mathrm{RS}(f)$, we argue that any zero-error algorithm A for f_{usab} also solves f_{sab}. The main observation is that any input to f_{sab} can be completed to an input to f_{usab} by replacing some asterisks or obelisks with 0s and 1s. To see this, let x be an input to f_{sab}. Without loss of generality, $x \in P_f$. Then there are two strings $y, z \in \mathrm{Dom}(f)$ that are consistent with x, satisfying $f(y) = 0$ and $f(z) = 1$.

The strings y and z disagree on some set of bits B, and x has a $*$ or † on all of B. Consider starting with y and flipping the bits of B one by one, until we reach the string z. At the beginning, we have $f(y) = 0$, and at the end, we reach $f(z) = 1$. This means that at

some point in the middle, we must have flipped a bit that flipped the string from a 0-input to a 1-input. Let w_0 and w_1 be the inputs where this happens. They differ in only one bit. If we replace that bit with $*$ or †, we get a partial assignment w consistent with both, so $w \in P_f$. Moreover, w is consistent with x. This means we have completed an arbitrary input to f_{sab} to an input to f_{usab}, as claimed.

Now, the algorithm A must find an asterisk or obelisk in any input to f_{usab}. But since each input to f_{sab} can be viewed as an input to f_{usab} with added asterisks and obelisks, the algorithm A also finds an asterisk or obelisk in any input to f_{sab}. Thus $\text{RS}(f) = R_0(f_{\text{sab}}) \leq R_0(f_{\text{usab}}) = \text{RS}_1(f)$. ◀

4 Direct Sum and Composition Theorems

In this section, we establish some composition theorems for RS. To do so, we first need to establish direct sum theorems for the problem f_{sab}. In fact, our direct sum theorems hold more generally for zero-error randomized query complexity of partial functions (and even relations). We will require Yao's minimax theorem [19]:

▶ **Theorem 11** (Yao). *Let f be a partial function. There is a distribution μ over inputs in $\text{Dom}(f)$ such that all zero-error algorithms for f use at least $R_0(f)$ expected queries on μ.*

4.1 Direct Sum Theorems

We start by defining the m-fold direct sum of a function f, which is simply the function that accepts m inputs to f and outputs f evaluated on all of them.

▶ **Definition 12.** Let $f : \text{Dom}(f) \to \mathcal{Z}$, where $\text{Dom}(f) \subseteq \mathcal{X}^n$ be a partial function with input and output alphabets $\mathcal{X}$ and $\mathcal{Z}$. The m-fold direct sum of f is the partial function $f^{\oplus m} : \text{Dom}(f)^m \to \mathcal{Z}^m$ such that for all $x_i \in \text{Dom}(f)$,

$$f(x_1, x_2, \ldots, x_m) = (f(x_1), f(x_2), \ldots, f(x_m)). \tag{1}$$

We can now prove a direct sum theorem for zero-error randomized query complexity. We prove these results for partial functions, although they also hold for relations.

▶ **Theorem 13** (Direct sum). *For any n-bit partial function f and any positive integer m, we have $R_0(f^{\oplus m}) = mR_0(f)$. Moreover, if μ is the hard distribution for f given by Theorem 11, then $\mu^{\otimes m}$ is a hard distribution for $f^{\oplus m}$.*

Proof. The upper bound follows from running the $R_0(f)$ algorithm on each of the m inputs to f. By linearity of expectation, this solves all m inputs after $mR_0(f)$ expected queries.

We now prove the lower bound. Let A be a zero-error randomized algorithm for $f^{\oplus m}$ that uses T expected queries when run on inputs from $\mu^{\otimes m}$. We convert A into an algorithm B for f that uses T/m expected queries when run on inputs from μ.

Given an input $x \sim \mu$, the algorithm B generates $m-1$ additional "fake" inputs from μ. B then shuffles these together with x, and runs A on the result. The input to A is then distributed according to $\mu^{\otimes m}$, so A uses T queries (in expectation) to solve all m inputs. B then reads the solution to the true input x.

Note that most of the queries A makes are to fake inputs, so they don't count as real queries. The only real queries B has to make happen when A queries x. But since x is shuffled with the other (indistinguishable) inputs, the expected number of queries A makes to x is the same as the expected number of queries A makes to each fake input; this must equal T/m. Thus B makes T/m queries to x (in expectation) before solving it.

Since B is a zero-error randomized algorithm for f that uses T/m expected queries on inputs from μ, we must have $T/m \geq R_0(f)$ by Theorem 11. Thus $T \geq mR_0(f)$. ◀

For our applications, however, we will need a strengthened version of this theorem, which we call a threshold direct sum theorem for R_0.

▶ **Theorem 14** (Threshold direct sum). *Given an input to $f^{\oplus m}$ sampled from $\mu^{\otimes m}$, we consider solving only some of the m inputs to f. We say an input x to f is solved if a z-certificate was queried that proves $f(x) = z$. Then any randomized algorithm that takes an expected T queries and solves an expected k of the m inputs when run on inputs from $\mu^{\otimes m}$ must satisfy $T \geq kR_0(f)$.*

Proof. Let A be such an algorithm. We convert A into an algorithm B for solving f on inputs from μ. The algorithm B is very similar to the algorithm in the proof of Theorem 13: on input $x \sim \mu$, it generates $m-1$ additional inputs from μ, shuffles them, and feeds them into A. The algorithm A uses an expected T queries, but since x is shuffled with the fake inputs, it gets queried only T/m times in expectation. Moreover, the algorithm A solves an expected k of the m inputs, so the expected number of times it solves x is k/m. This means B solves x with probability k/m.

Moreover, when B solves x, it also finds a certificate. So by Lemma 3, we get a zero-error algorithm with expected query complexity $(T/m)/(k/m) = T/k$. We conclude that $T/k \geq R_0(f)$, so $T \geq kR_0(f)$, as desired. ◀

4.2 Composition Theorems

Using the direct sum and threshold direct sum theorems we have established, we can now prove composition theorems for randomized sabotage complexity. We start with the behavior of RS itself under composition.

▶ **Theorem 15.** *Let f and g be partial functions. Then* $\mathrm{RS}(f \circ g) \geq \mathrm{RS}(f)\,\mathrm{RS}(g)$.

Proof. Let A be any algorithm for $(f \circ g)_{\mathrm{sab}}$, and let T be the expected query complexity of A (maximized over all inputs). We turn A into an algorithm B for f_{sab}.

B takes a sabotaged input x for f. It then runs A on a sabotaged input to $f \circ g$ constructed as follows. Each 0 bit of x is replaced with a 0-input to g, each 1 bit of x is replaced with a 1-input to g, and each $*$ or † of x is replaced with a sabotaged input to g. The sabotaged inputs are generated from μ, the hard distribution for g_{sab} obtained from Theorem 11. The 0-inputs are generated by first generating a sabotaged input, and then selecting a 0-input consistent with that sabotaged input. The 1-inputs are generated analogously.

This is implemented in the following way. On input x, the algorithm B generates n sabotaged inputs from μ (the hard distribution for g_{sab}), where n is the length of the string x. Call these inputs $y_1, y_2, \ldots, y_n$. B then runs the algorithm A on this collection of n strings, pretending that it's an input to $f \circ g$, with the following caveat: whenever A tries to query a $*$ or † in an input y_i, B instead queries x_i. If x_i is 0, B selects an input from $f^{-1}(0)$ consistent with y_i, and replaces y_i with this input. It then returns to A an answer consistent with the new y_i. If x_i is 1, B selects a consistent input from $f^{-1}(1)$ instead. If x_i is a $*$ or †, B returns a $*$ or † respectively.

Now, by Theorem 14, if A makes T expected queries, the expected number of $*$ or † entries it finds among $y_1, y_2, \ldots, y_n$ is at most $T/\mathrm{RS}(g)$. It follows that the expected number of queries B makes to x is at most $T/\mathrm{RS}(g)$. Thus we have $\mathrm{RS}(f) \leq T/\mathrm{RS}(g)$, which gives $T \geq \mathrm{RS}(f)\,\mathrm{RS}(g)$. ◀

Using this we can lower bound the randomized query complexity of composed functions. We use f^n to denote the function f composed with itself n times, i.e., $f^1 = f$ and $f^{i+1} = f \circ f^i$.

▶ **Corollary 16.** *Let $f : \{0,1\}^n \to \{0,1\}$ be a partial function. Then $R(f^n) \geq \mathrm{RS}(f)^n/3$.*

This follows straightforwardly from observing that $R(f^n) = R_{1/3}(f^n) \geq (1 - 2/3)\,\mathrm{RS}(f^n)$ (using Theorem 8) and $\mathrm{RS}(f^n) \geq \mathrm{RS}(f)^n$ (using Theorem 15).

We can also prove a composition theorem for randomized query complexity in terms of randomized sabotage complexity. In particular this yields a composition theorem for $R(f \circ g)$ when $R(g) = \Theta(\mathrm{RS}(g))$.

▶ **Theorem 17.** *Let f and g be partial functions. Then $\overline{R}_\epsilon(f \circ g) \geq \overline{R}_\epsilon(f)\,\mathrm{RS}(g)$.*

Proof. The proof follows a similar argument to the proof of Theorem 15. Let A be a randomized algorithm for $f \circ g$ that uses T expected queries and makes error ϵ. We turn A into an algorithm B for f by having B generate inputs from μ, the hard distribution for g_{sab}, and feeding them to A, as before. The only difference is that this time, the input x to B is not a sabotaged input. This means it has no $*$ or † entries, so all the sabotaged inputs that B generates turn into 0- or 1-inputs if A tries to query a $*$ or † in them.

Since A uses T queries, by Theorem 14, it finds at most $T/\,\mathrm{RS}(g)$ asterisks or obelisks (in expectation). Therefore, B makes at most $T/\,\mathrm{RS}(g)$ expected queries to x. Since B is correct whenever A is correct, its error probability is at most ϵ. Thus $\overline{R}_\epsilon(f) \leq T/\,\mathrm{RS}(g)$, and thus $T \geq \overline{R}_\epsilon(f)\,\mathrm{RS}(g)$. ◀

Setting ϵ to 0 yields the following corollary.

▶ **Corollary 18.** *Let f and g be partial functions. Then $R_0(f \circ g) \geq R_0(f)\,\mathrm{RS}(g)$.*

For the more commonly used $R(f \circ g)$, we obtain the following composition result.

▶ **Corollary 19.** *Let f and g be partial functions. Then $R(f \circ g) \geq R(f)\,\mathrm{RS}(g)/10$.*

This follows from $R(f \circ g) \geq \overline{R}_{1/3}(f \circ g) \geq \overline{R}_{1/3}(f)\,\mathrm{RS}(g) \geq R(f)\,\mathrm{RS}(g)/10$, where we used $\overline{R}_{1/3}(f) \geq R(f)/10$, which can be shown by error reduction and Markov's inequality.

Finally, we can also show an upper bound composition result for randomized sabotage complexity. We defer the proof to the full version of this paper.

▶ **Theorem 20.** *Let f and g be partial functions. Then $\mathrm{RS}(f \circ g) \leq \mathrm{RS}(f)R_0(g)$. We also have $\mathrm{RS}(f \circ g) = O(\mathrm{RS}(f)R(g)\log \mathrm{RS}(f))$.*

5 Composition with the index function

To prove the composition result, we require the strong direct product theorem for randomized query complexity that was established by Drucker [4].

▶ **Theorem 21** (Drucker)**.** *Let f be a partial Boolean function, and let k be a positive integer. Then any randomized algorithm for $f^{\oplus k}$ that uses at most $\gamma^3 kR(f)/11$ queries has success probability at most $(1/2 + \gamma)^k$, for any $\gamma \in (0, 1/4)$.*

The first step to proving the main result that $R(f \circ \textsc{Ind} \circ g) = \Omega(R(f)R(\textsc{Ind})R(g))$ is to show that $R(\textsc{Ind} \circ g)$ equals $\mathrm{RS}(\textsc{Ind} \circ g)$ up to constants if the index gadget is large enough.

▶ **Theorem 22.** *Let f be a partial Boolean function, and let $m = \Omega(R(f)^{1.1})$. Then $\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega(R(f)\log m) = \Omega(R(\textsc{Ind}_m)R(f))$.*

Moreover, if $f^{\oplus c}_{\mathrm{ind}}$ is the defined as the index function on $c + 2^c$ bits composed with f in only the first c bits, we have $\mathrm{RS}_1(f^{\oplus c}_{\mathrm{ind}}) = \Omega(cR(f))$ when $c = 1.1\log R(f) + \Omega(1)$.

Proof. Consider what the inputs to $(\textsc{Ind}_m \circ f)_{\text{sab}}$ look like. We can split an input to $\textsc{Ind}_m$ into a small index section and a large array section. To sabotage an input to $\textsc{Ind}_m$, it suffices to sabotage the array element that the index points to (using only a single star). It follows that to sabotage an input to $\textsc{Ind}_m \circ f$, it suffices to sabotage the input to f at the array element that the index points to. In other words, the only stars in the input will be in one array cell, whose index is the output of the first $\log m$ copies of f.

We now convert an $\mathrm{RS}(\textsc{Ind}_m \circ f)$ algorithm into a randomized algorithm for $f^{\log m}$. First, using Markov's inequality, we get a $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ query randomized algorithm that finds a $*$ or † with probability $1/2$ if the input is sabotaged. Next, consider running this algorithm on a non-sabotaged input. It makes $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ queries. With probability $1/2$, one of these queries will be in the array cell whose index is the true answer to $f^{\log m}$ evaluated on the first $n \log m$ bits. We can then consider a new algorithm A that runs the above algorithm for $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ queries, then picks one of the $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ queries at random, and if that query is in an array cell, it outputs the index of that cell. Then A uses $2\,\mathrm{RS}(\textsc{Ind}_m \circ f)$ queries and evaluates $f^{\log m}$ with probability at least $\mathrm{RS}(\textsc{Ind}_m \circ f)^{-1}/4$.

Next, Theorem 21 implies that for any $\gamma \in (0, 1/4)$, either A's success probability is smaller than $(1/2+\gamma)^{\log m}$, or else A uses at least $\gamma^3(\log m)R(f)/11$ queries. This means either $\mathrm{RS}(\textsc{Ind}_m \circ f)^{-1}/4 \le (1/2+\gamma)^{\log m}$ or $2\,\mathrm{RS}(\textsc{Ind}_m \circ f) \ge \gamma^3(\log m)R(f)/11$, which means

$$\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega\left(\gamma^3 \min\left\{\left(\frac{2}{1+2\gamma}\right)^{\log m}, R(f)\log m\right\}\right). \tag{2}$$

Now, we have

$$\left(\frac{2}{1+2\gamma}\right)^{\log m} = m^{\log(2/(1+2\gamma))} = m^{1-\log(1+2\gamma)} \ge m^{1-2(\log e)\gamma} \ge m^{1-3\gamma}. \tag{3}$$

If $m \ge (R(f)\log R(f))^{(1-3\gamma)^{-1}}$, the above is at least $R(f)\log R(f) = \Omega(R(f)\log m)$, which means $\mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega(\gamma^3 R(f)\log m)$.

Note that $(1-3\gamma)^{-1} \le 1 + 12\gamma$ for all $\gamma \le 1/4$. Setting $r = 13\gamma$, we get

$$m = \Omega(R(f)^{1+r}) \Rightarrow \mathrm{RS}(\textsc{Ind}_m \circ f) = \Omega(r^3 R(f)\log m) \tag{4}$$

for all r satisfying $r = O(1)$ and $r = \Omega(\log\log R(f)/\log R(f))$. Setting $r = 0.1$ gives the desired result. The lower bound on $\mathrm{RS}_1(f^{\oplus}_{\text{ind}})$ follows similarly once we observe that sabotaging the array cell indexed by the outputs to the c copies of f introduces only one asterisk or obelisk, so the above argument lower bounds RS_1 and not only RS. ◀

Finally, we can prove Theorem 1, more precisely stated as follows.

▶ **Theorem 23.** *Let f and g be (possibly partial) functions, and let $m = \Omega(R(g)^{1.1})$. Then $R(f \circ \textsc{Ind}_m \circ g) = \Omega(R(f)R(g)\log m) = \Omega(R(f)R(\textsc{Ind}_m)R(g))$.*

Proof. By Corollary 19, we have $R(f \circ \textsc{Ind}_m \circ g) \ge R(f)\,\mathrm{RS}(\textsc{Ind}_m \circ g)/10$. Combining with Theorem 22 gives $R(f \circ \textsc{Ind}_m \circ g) = \Omega(R(f)R(g)\log m)$, as desired. ◀

6 Lifting theorem

To establish the connection between lifting theorems, we start with the following lemma, which gives a sabotage lower bound in the communication complexity setting.

▶ **Lemma 24.** *Let f be a (possibly partial) Boolean function on n bits, and let G_b be the index gadget on $\{0,1\}^b \times \{0,1\}^{2^b}$, with $b = O(\log n)$. Then*

$$R^{\mathrm{cc}}(f \circ G_b) = \Omega\left(\frac{R_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b)}{\log n \log\log n}\right), \tag{5}$$

where G'_b is the index gadget mapping $\{0,1\}^b \times \{0,1,*,\dagger\}^{2^b}$ to $\{0,1,*,\dagger\}$.

Proof. We'll use a randomized protocol A for $f \circ G_b$ to construct a zero-error protocol B for $f_{\mathrm{usab}} \circ G'_b$. Note the given input to $f_{\mathrm{usab}} \circ G'_b$ must have a unique copy of G'_b that evaluates to $*$ or $\dagger$, with all other copies evaluating to 0 or 1. The goal of B is to find this copy and determine if it evaluates to $*$ or $\dagger$. This will evaluate $f_{\mathrm{usab}} \circ G'_b$ with zero error.

Note that if we replace all $*$ and $\dagger$ symbols in Bob's input with 0 or 1, we'd get a valid input to to $f \circ G_b$, which we can evaluate using A. Moreover, there is a single special $*$ or $\dagger$ in Bob's input that governs the value of this input to $f \circ G_b$. Without loss of generality, we assume that if the special symbol is replaced by 0, the function $f \circ G_b$ evaluates to 0, and if it is replaced by 1, it evaluates to 1.

We can now binary search to find this special symbol. There are at most $n2^b$ asterisks and obelisks in Bob's input. We can set the left half to 0 and the right half to 1, and evaluate the resulting input using A. If the answer is 0, the special symbol is on the left half; otherwise, it is on the right half. We can proceed to binary search in this way, until we've zoomed in on one gadget that must contain the special symbol. This requires narrowing down the search space from n possible gadgets to 1, which requires $\log n$ rounds. Each round requires a call to A, times a $O(\log\log n)$ factor for amplification. We can therefore find the right gadget with bounded error, using $O(R^{\mathrm{cc}}(f \circ G_b) \log n \log\log n)$ bits of communication.

Once we've found the right gadget, we can certify its validity by having Alice send the right index to Bob, using b bits of communication. Since we found a certificate with constant probability, we can use Lemma 3 to turn this into a zero-error algorithm. Thus

$$R_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b) = O(b + R^{\mathrm{cc}}(f \circ G_b) \log n \log\log n). \tag{6}$$

Since $b = O(\log n)$, we get $R_0^{\mathrm{cc}}(f_{\mathrm{usab}} \circ G'_b) = O(R^{\mathrm{cc}}(f \circ G_b) \log n \log\log n)$. ◀

Equipped with this lemma we can prove the connection between lifting theorems (Theorem 2), stated more precisely as follows.

▶ **Theorem 25.** *Suppose that for all partial Boolean functions f on n bits, we have*

$$R_0^{\mathrm{cc}}(f \circ G_b) = \tilde{\Omega}(R_0(f)) \tag{7}$$

with $b = O(\log n)$. Then for all partial functions Boolean functions, we also have

$$R^{\mathrm{cc}}(f \circ G_{2b}) = \tilde{\Omega}(R(f)). \tag{8}$$

The loss in the $\tilde{\Omega}$ for the R result is only $\log n \log\log^2 n$ worse than the loss in the R_0 hypothesis.

Proof. First, we note that for any function f and positive integer c,

$$R^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{R^{\mathrm{cc}}(f_{\mathrm{ind}}^{\oplus c} \circ G_{2b})}{c \log c}\right). \tag{9}$$

To see this, note that we can solve $f_{\mathrm{ind}}^{\oplus c} \circ G_{2b}$ by solving the c copies of $f \circ G_{2b}$ and then examining the appropriate cell of the array. This uses $cR^{\mathrm{cc}}(f \circ G_{2b})$ bits of communication, times $O(\log c)$ since we must amplify the randomized protocol to an error of $O(1/c)$.

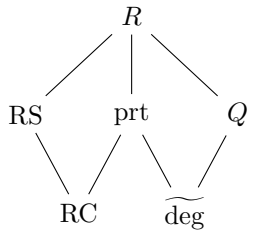


Figure 1 Lower bounds on $R(f)$.

Next, we apply Lemma 24 on $R^{\mathrm{cc}}(f_{\mathrm{ind}}^{\oplus c} \circ G_{2b})$ to get

$$R^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{R^{\mathrm{cc}}(f_{\mathrm{ind}}^{\oplus c} \circ G_{2b})}{c \log c}\right) = \Omega\left(\frac{R_0^{\mathrm{cc}}((f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}} \circ G'_{2b})}{c \log c \log n \log\log n}\right). \tag{10}$$

From here we want to use the assumed lifting theorem for R_0. However, there is a technicality: the gadget G'_{2b} is not the standard index gadget, and the function $(f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}$ does not have Boolean alphabet. To remedy this, we use two bits to represent each of the symbols $\{0, 1, *, \dagger\}$. Using this representation, we define a new function $(f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}^{\mathrm{bin}}$ on twice as many bits.

We now compare $(f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}^{\mathrm{bin}} \circ G_b$ to $(f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}} \circ G'_{2b}$. Note that the former uses two pointers of size b to index two bits, while the latter uses one pointer of size $2b$ to index one symbol in $\{0, 1, *, \dagger\}$ (which is equivalent to two bits). It's not hard to see that the former function is equivalent to the latter function restricted to a promise. This means the communication complexity of the former is smaller, so

$$R^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{R_0^{\mathrm{cc}}((f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}} \circ G'_{2b})}{c \log c \log n \log\log n}\right) = \Omega\left(\frac{R_0^{\mathrm{cc}}((f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}^{\mathrm{bin}} \circ G_b)}{c \log c \log n \log\log n}\right). \tag{11}$$

We're ready to use the assumed lifting theorem for R_0. To be more precise, let's suppose a lifting result that states $R_0^{\mathrm{cc}}(f \circ G_b) = \Omega(bR_0(f)/\log^k n)$ for some integer k. Applying this to the above gives

$$R^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{R_0^{\mathrm{cc}}((f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}^{\mathrm{bin}} \circ G_b)}{c \log c \log n \log\log n}\right) = \Omega\left(\frac{bR_0((f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}^{\mathrm{bin}})}{c \log c \log^{k+1} n \log\log n}\right). \tag{12}$$

We note that

$$R_0((f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}}^{\mathrm{bin}}) = \Omega(R_0((f_{\mathrm{ind}}^{\oplus c})_{\mathrm{usab}})) = \Omega(\mathrm{RS}_1(f_{\mathrm{ind}}^{\oplus c})). \tag{13}$$

Setting $c = 1.1 \log R(f) + \Omega(1)$, we have $\mathrm{RS}_1(f_{\mathrm{ind}}^{\oplus c}) = \Omega(cR(f))$ by Theorem 22. Thus

$$R^{\mathrm{cc}}(f \circ G_{2b}) = \Omega\left(\frac{bcR(f)}{c \log c \log^{k+1} n \log\log n}\right) = \Omega\left(\frac{bR(f)}{\log^{k+1} n \log\log^2 n}\right). \tag{14}$$

This gives the desired lifting theorem for R, with parameters at most $\log n \log\log^2 n$ worse than the assumed R_0 lifting theorem. ◀

7 Comparison with other lower bound methods

In this section we compare $\mathrm{RS}(f)$ with other lower bound techniques for bounded-error randomized query complexity. Figure 1 shows the two most powerful lower bound techniques for $R(f)$, the partition bound ($\mathrm{prt}(f)$) and quantum query complexity ($Q(f)$), which subsume

all other general lower bound techniques. The partition bound and quantum query complexity are incomparable, since there are functions for which the partition bound is larger, e.g., the OR function, and functions for which quantum query complexity is larger [2]. Another common lower bound measure, approximate polynomial degree ($\widetilde{\deg}$) is smaller than both.

Randomized sabotage complexity (RS) can be much larger than the partition bound and quantum query complexity as we show in this section. We also show that randomized sabotage complexity is always as large as randomized certificate complexity (RC), which itself is larger than block sensitivity, another common lower bound technique. Lastly, we also show that $R_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$, showing that RS is a quadratically tight lower bound, even for zero-error randomized query complexity.

7.1 Partition bound and quantum query complexity

We start by showing the superiority of randomized sabotage complexity against the two best lower bounds for $R(f)$. Informally, what we show is that any separation between $R(f)$ and a lower bound measure like $Q(f)$, $\mathrm{prt}(f)$, or $\widetilde{\deg}(f)$ readily gives a similar separation between $\mathrm{RS}(f)$ and the same measure.

▶ **Theorem 26.** *There exist total functions f and g such that $\mathrm{RS}(f) \geq \mathrm{prt}(f)^{2-o(1)}$ and $\mathrm{RS}(g) = \widetilde{\Omega}(Q(g)^{2.5})$. There also exists a total function h with $\mathrm{RS}(h) \geq \widetilde{\deg}(h)^{4-o(1)}$.*

Proof. These separations were shown with $R(f)$ in place of $\mathrm{RS}(f)$ in [1] and [2]. To get a lower bound on RS, we can simply compose IND with these functions and apply Theorem 22. This increases RS to be the same as R (up to logarithmic factors), but it does not increase prt, $\widetilde{\deg}$, or Q more than logarithmically, so the desired separations follow. ◀

As it turns out, we didn't even need to compose IND with these functions. It suffices to observe that they all use the cheat sheet construction, and that an argument similar to the proof of Theorem 22 implies that $\mathrm{RS}(f_{\mathrm{CS}}) = \widetilde{\Omega}(R(f))$ for all f (where f_{CS} denotes the cheat sheet version of f, as defined in [1]). In particular, cheat sheets can never be used to separate RS from R (by more than logarithmic factors).

7.2 Randomized certificate complexity

Randomized certificate complexity, $\mathrm{RC}(f)$, is a lower bound for $R(f)$ first studied in [**?**]. We can show that for any partial function f, randomized sabotage complexity upper bounds randomized certificate complexity.

▶ **Theorem 27.** *Let f be a partial function. Then $\mathrm{RS}(f) \geq \mathrm{RC}(f)/4$.*

We defer the definition of $\mathrm{RC}(f)$ and the proof of this theorem to the full version of the paper.

7.3 Zero-error randomized query complexity

▶ **Theorem 28.** *Let f be a total function. Then $R_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$ or alternately, $\mathrm{RS}(f) = \Omega(\sqrt{R_0(f)/\log R_0(f)})$.*

Proof. Let A be the $\mathrm{RS}(f)$ algorithm. The idea is to run A on an input to x for long enough that we can ensure it queries a bit in every sensitive block of x; this will mean A found a certificate for x. That will allow us to turn the algorithm into a zero-error algorithm for f.

Let x be any input and let b be a block of x. If we replace the bits of x specified by b with stars, then we can find a $*$ with probability $1/2$ by running A for $2\,\mathrm{RS}(f)$ queries by Markov's inequality. This means that if we run A on x for $2\,\mathrm{RS}(f)$ queries, it has at least $1/2$ probability of querying a bit in any given block of x. Repeating this k times, we get a $2k\,\mathrm{RS}(f)$ query algorithm that queries a bit in any given block of x with probability at least $1 - 2^{-k}$.

Now, by [13], the number of sensitive blocks in x is at most $\mathrm{RC}(f)^{\mathrm{bs}(f)}$ for a total function f. Our probability of querying a bit in all of these blocks is at least $1 - 2^{-k}\,\mathrm{RC}(f)^{\mathrm{bs}(f)}$ by the union bound. When $k \geq 1 + \mathrm{bs}(f)\log_2 \mathrm{RC}(f)$, this is at least $1/2$. Since a bit from every block is a certificate, by Lemma 3, we can turn this into a zero-error randomized algorithm with expected query complexity at most $4(1 + \mathrm{bs}(f)\log_2 \mathrm{RC}(f))\,\mathrm{RS}(f)$, which gives $R_0(f) = O(\mathrm{RS}(f)\,\mathrm{bs}(f)\log \mathrm{RC}(f))$. Since $\mathrm{bs}(f) \leq \mathrm{RC}(f) \leq \mathrm{RS}(f)$ by Theorem 27, we have $R_0(f) = O(\mathrm{RS}(f)^2 \log \mathrm{RS}(f))$, or $\mathrm{RS}(f) = \Omega(\sqrt{R_0(f)/\log R_0(f)})$. ◀

References

1. Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. *To appear in Proceedings of STOC 2016. arXiv preprint* arXiv:1511.01937, 2015.
2. Andris Ambainis, Martins Kokainis, and Robin Kothari. Nearly optimal separations between communication (or query) complexity and partitions. *To appear in Proceedings of CCC 2016. arXiv preprints* arXiv:1512.00661 *and* arXiv:1512.01210, 2015.
3. Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.
4. Andrew Drucker. Improved direct product theorems for randomized query complexity. *Computational Complexity*, 21(2):197–244, 2012. doi:10.1007/s00037-012-0043-7.
5. Mika Göös, T.S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication vs. partition number. *Electronic Colloquium on Computational Complexity (ECCC)* TR15-169, 2015.
6. Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 1077–1088, Oct 2015. doi:10.1109/FOCS.2015.70.
7. Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC 2007)*, pages 526–535, 2007. doi:10.1145/1250790.1250867.
8. Rahul Jain and Hartmut Klauck. The partition bound for classical communication complexity and query complexity. In *Proceedings of the 2010 IEEE 25th Annual Conference on Computational Complexity*, CCC'10, pages 247–258, 2010. doi:10.1109/CCC.2010.31.
9. Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. *Information Processing Letters*, 110(20):893–897, 2010. doi:10.1016/j.ipl.2010.07.020.
10. Rahul Jain, Troy Lee, and Nisheeth K. Vishnoi. A quadratically tight partition bound for classical communication complexity and query complexity. *arXiv preprint* arXiv:1401.4512, 2014.
11. Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3-4):191–204, 1995. doi:10.1007/BF01206317.

12 Shelby Kimmel. Quantum adversary (upper) bound. In *Automata, Languages, and Programming*, volume 7391 of *Lecture Notes in Computer Science*, pages 557–568, 2012. doi:10.1007/978-3-642-31594-7_47.

13 Raghav Kulkarni and Avishay Tal. On fractional block sensitivity. *Electronic Colloquium on Computational Complexity (ECCC)* TR13-168, 2013.

14 Troy Lee, Rajat Mittal, Ben W. Reichardt, Robert Špalek, and Mario Szegedy. Quantum query complexity of state conversion. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 344–353, 2011. arXiv:1011.3020, doi:10.1109/FOCS.2011.75.

15 Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago Journal of Theoretical Computer Science*, 2014(6), July 2014. doi:10.4086/cjtcs.2014.006.

16 Denis Pankratov. Direct sum questions in classical communication complexity. Master's thesis, University of Chicago, 2012.

17 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998. doi:10.1137/S0097539795280895.

18 Avishay Tal. Properties and applications of Boolean function composition. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS'13, pages 441–454, 2013. doi:10.1145/2422436.2422485.

19 A. Yao. Probabilistic computations: Toward a unified measure of complexity. *Proceedings of the 18th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 222–227, 1977. doi:10.1109/SFCS.1977.24.

Coding for Interactive Communication Correcting Insertions and Deletions*

Mark Braverman[†1], Ran Gelles[2], Jieming Mao[3], and Rafail Ostrovsky[‡4]

1 **Department of Computer Science, Princeton University, Princeton, USA**
mbraverm@cs.princeton.edu

2 **Department of Computer Science, Princeton University, Princeton, USA**
rgelles@cs.princeton.edu

3 **Department of Computer Science, Princeton University, Princeton, USA**
jiemingm@cs.princeton.edu

4 **Department of Computer Science and Department of Mathematics, UCLA, Los Angeles, USA**
rafail@cs.ucla.edu

Abstract

We consider the question of interactive communication, in which two remote parties perform a computation while their communication channel is (adversarially) noisy. We extend here the discussion into a more general and stronger class of noise, namely, we allow the channel to perform *insertions* and *deletions* of symbols. These types of errors may bring the parties "out of sync", so that there is no consensus regarding the current round of the protocol.

In this more general noise model, we obtain the first interactive coding scheme that has a constant rate and tolerates noise rates of up to $1/18-\varepsilon$. To this end we develop a novel primitive we name *edit distance tree code*. The edit distance tree code is designed to replace the Hamming distance constraints in Schulman's tree codes (STOC 93), with a stronger edit distance requirement. However, the straightforward generalization of tree codes to edit distance does not seem to yield a primitive that suffices for communication in the presence of synchronization problems. Giving the "right" definition of edit distance tree codes is a main conceptual contribution of this work.

1998 ACM Subject Classification E.4 Coding and Information Theory, F.1.2 Models of Computation

Keywords and phrases Interactive communication, coding, edit distance

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.61

* Full version of this paper can be found at http://arxiv.org/abs/1508.00514.

† Mark Braverman is supported in part by an NSF CAREER award (CCF-1149888), NSF CCF-1215990, a Turing Centenary Fellowship, a Packard Fellowship in Science and Engineering, and the Simons Collaboration on Algorithms and Geometry.

‡ Rafail Ostrovsky is supported in part by NSF grants 09165174, 1065276, 1118126 and 1136174, DARPA, US-Israel BSF grant 2008411, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award, Teradata Research Award, and Lockheed-Martin Corporation Research Award. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 61; pp. 61:1–61:14

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

1 Introduction

In the setting of interactive communication two remote parties, Alice and Bob, wish to run a distributed protocol utilizing a noisy communication channel. The study of this problem was initiated by the seminal work of Schulman [22, 23, 24] who showed a coding scheme for interactive protocols in which the communication complexity of the resilient protocol is larger than the communication of the input (noiseless) protocol by only a constant factor. Schulman's coding schemes tolerates *random noise* where each bit is flipped with a small probability, as well as some *adversarial noise* where the only restriction on the noise is the amount of bits being flipped by the adversary. Subsequently, many works considered the question of interactive communication, obtaining coding schemes reaching optimality in terms of their computational efficiency [13, 14, 2, 4, 3, 15], communication efficiency [19, 17, 12], and noise resilience, both in the standard setting [7, 8, 6, 15], and in various other noise models and settings [21, 10, 11, 16, 15, 1, 5, 9].

The recent successes in developing the theory of interactive error-correcting codes brought the study of two-way interactive coding to nearly match what we know about good codes against adversarial noise in the one-way setting.

So far, all works focused on either substitutions (where Eve can substitute a sent symbol with a different symbol from the alphabet) or erasures (where Eve can substitute a sent symbol with a $\perp$). In this work we extend the question of coding for interactive communication over noisy channels to a more general type of noise. Namely, we consider channels with *insertions and deletions* (indels). In the one-way setting, this corresponds to the insertion and deletion model, where Eve is allowed to completely remove transmitted symbols, or inject new symbols. Note that this model is stronger than the substitution model, since a substitution can always be implemented as a deletion followed by an insertion. Even in the one-way regime, this model is more difficult to analyze than the model with substitution errors. As an example, Schulman and Zuckerman [25] gave a polynomial-time encodable and decodable codes for insertion/deletion errors. Their code can tolerate around $\frac{1}{100}$ fraction of insertion/deletion errors. This should be contrasted with efficient codes in the standard noise setting, e.g., [18], tolerating about $\frac{1}{4}$ fraction of bit flips.

The major additional challenge in dealing with indels in the interactive setting compared to the non-interactive indel model and the interactive substitutions model, is that we can no longer assume that Alice and Bob are synchronized: at a given time they may be at different stages of their sides of the protocol! Indeed, if Eve deletes Alice's transmission to Bob and additionally injects a 'spoofed' reply from Bob back to Alice, then while Bob has received no message and assumes the protocol hasn't advanced yet, Alice has received a spoofed reply, and proceeds to the next step of her protocol. From this point and on, unless the insertion/deletion is detected, the parties are unsynchronized, as they run different steps of the protocol. The challenge in dealing with this model is to design a protocol that manages to succeed even without knowing whether the two parties are synchronized.

1.1 Modeling insertions and deletions

Some care is required when dealing with insertion and deletion noise patterns, as certain choices make the model too strong or too weak. For example, consider the case where Alice and Bob send each other symbols in an alternating way. Then, even if a single deletion of a symbol is allowed the noise can cause the protocol to "hang": Bob will be waiting for a symbol from Alice, while Alice will be waiting for Bob's response. Clearly, such a model is

too strong for our purpose, and we should restrict the allowed noise patterns to preserve the protocol's liveliness.

There are two main paradigms for distributed protocol in which parties are not fully-synchronized. The first is a *message-driven* paradigm, in which each party "sleeps" until the arrival of a new message that triggers it into performing some computation and sending a message to the other party. The second is *clock-driven*, where each party holds a clock: each clock tick, the party wakes up, checks the incoming messages queue, performs some computation, and sends a message to the other side. The issue here is that different parties may have mismatching or skewed clocks. Then, instead of acting in an alternating manner, one party may wake up several times while the other party is still asleep.

We emphasize that if the parties have matching clocks, then no insertions and deletions are possible – channel corruption in this case has either the effect of changing one symbol to another (as in a standard noisy channel), or causing a detectible corruption, i.e., an erasure. Both these types of noise are substantially weaker than insertions and deletions, and were already analyzed in previous work (e.g., [24, 8, 11, 9]).

Our noise model, which we describe shortly, makes sense for both the above paradigms: it guarantees liveliness in a message-driven setting; for the clock-driven model, we can show that any such settings reduces to our model, that is, any resilient protocol in our model can be used to obtain a resilient protocol in the clock-driven setting. The skewness of the clocks in that case, is related to the noise-resilience of the protocol in our model. See the full version for the complete details.

In this work we assume a message-driven setting, where the parties normally speak in alternating manner. Any corruption in our model must be a deletion which is followed by an insertion. We name each such tampering as an *edit corruption*.

▶ **Definition 1** (Edit Corruption)**.** An edit-corruption is a single deletion followed by a single insertion (whether the inserted symbol is aimed at the same or the opposite party as the deleted message).

This gives rise to two types of attacks Eve can perform: (i) delete a symbol and replace it with a different symbol (insert a symbol at the same direction as the deleted symbol; a substitution attack); (ii) delete a symbol, and insert a spoofed reply from the other side (insert a symbol at the opposite direction of the deleted symbol). The second type of corruption has an effect of making the parties 'out-of-sync': one party advances one step in the protocol, while the other does not; see Figure 1. Note that a substitution has a cost a single corruption, i.e., it is counted as a single deletion followed by a single insertion. Also note that although an outside viewer can split Eve's attack into pairs of deletion-and-insertion, the string that a certain party receives, from that party's own view, suffers an *arbitrary* pattern of insertions/deletions.

1.2 Our Results and Techniques

Tree codes with edit distance. When only a single message is to be transmitted (i.e., in the one-way setting), codes that withstand insertions and deletions were first considered by Levenshtein [20]. In such codes, each two codewords must be far away in their *edit distance*, a notion of distance that captures the amount of insertions/deletions it takes to convert one codeword to another. The edit distance replaces the Hamming distance, which essentially counts the amount of bit flips required to turn one codeword to another.

A key ingredient in interactive-communication schemes is the *tree code* [24], a labeled tree such that the labels on each descending path in the tree can be seen as a codeword. The

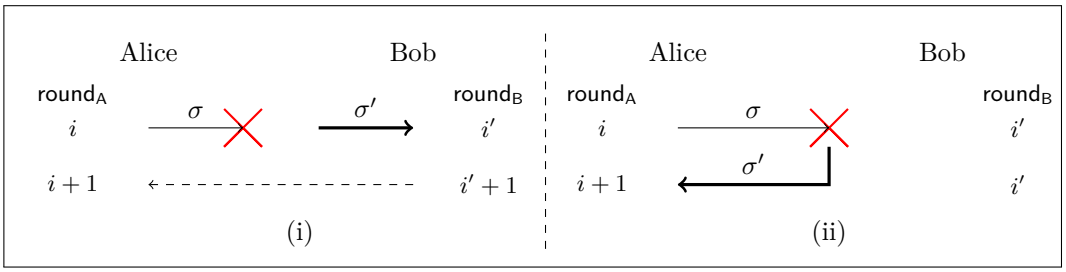


Figure 1 An illustration of the two insertion/deletion attacks: (i) a deletion followed by an insertion in the same direction (a substitution); (ii) a deletion followed by an insertion to the opposite direction (an out-of-sync attack). The deleted transmission is marked with a cross, and the inserted transmission is marked with a bold arrow. The dashed arrow denotes a possible (non-interrupted) reply.

tree code is parametrized by a distance parameter α, and it holds that any two codewords whose paths diverge from the same node, are at least α-apart in their Hamming distance. Encoding a message via a tree code allows a party to *eventually* obtain the message sent by the other side, as long as not too many errors have happened. This in turn allows the parties to correct errors that previously occurred in the simulation, and revert the simulation back into a correct state [24, 8]. In order to keep the communication overhead a constant factor when tree code encoding is used in interactive schemes, it is required that each label of the tree comes from an alphabet of constant size (that is, the size of the alphabet is independent of the tree's depth and thus independent of the length of protocol to simulate).

It is only natural to believe that we could obtain interactive-communication schemes that withstand insertions/deletions by replacing tree codes with a stronger notion of codes, namely, *edit distance tree codes*. In edit distance tree codes, each two codewords (possibly of different lengths) which diverge at a certain point, are required to be far apart in their edit distance rather than their Hamming distance. Yet, since the parties are not synchronized, new difficulties arise. To give a simple example, assume Alice sends one of the two following encodings $s_1 = ABCAAABBB$ and $s_2 = ABCABCAAA$, and assume Bob has receives the string $ABCBBB$. If Bob knew that Alice thinks she is in round 6 of the protocol, he would decode to s_2; if he knew that Alice thinks she is in round 9, he would decode to s_1. Alas, he does not know which is the case!

To mitigate situations in which not being synchronized may hurt us, we require an even stronger property, namely, we wish that the suffixes (of arbitrary lengths) of any two overlapping codewords will have appropriately large edit distance (see Definitions 14 and 15 for the precise condition). This stronger property guarantees that two "branches" in the tree are far apart in their edit distance, even when they are shifted with respect to each other due to lack of synchronization possibly caused by previous indels. We can then show that as long as not too many indels occurred in the *suffix* of the received codeword, the tree-code succeeds to recover the entire sent message. Crucial in this approach is a notion of distance we call *suffix distance* (Definition 21), that measures the amount of noise in a codewords' suffix. This generalizes a distance measure by Franklin et al. [10, 11] (see also Braverman and Efremenko [6]) to the case where the received word may be misaligned with respect to the sent word, due to indels.

Alas, while (Hamming distance) tree codes over a constant alphabet were shown to exist by Schulman [24], it is not clear if such trees exist for edit distance, and if so, for which distance parameter α, as Schulman's proof doesn't carry over to the edit distance case.

Our first result (Section 3) shows the existence of edit distance tree codes, for any distance parameter α,

▶ **Theorem 2** (Informal). *For any $\alpha < 1$ and any $d, n \in \mathbb{N}$ there exists a d-ary edit distance tree code of depth n over a constant-size alphabet.*

As in the case of standard tree codes, finding an *efficient* construction for such trees remains an important open question. Building on the techniques of Gelles, Moitra and Sahai [13, 14] we give in the full version an efficient randomized construction of a relaxed notion for edit-distance tree codes, we call a *potent edit distance* tree codes. These trees satisfy the edit-distance guarantee *almost* everywhere, and are good enough to replace the tree-code notion of Theorem 2 in most applications.

▶ **Theorem 3** (Informal). *For any $\alpha < 1$ and any $d, n \in \mathbb{N}$ there exists a randomized construction of a d-ary potent edit distance tree code of length n over a constant-size alphabet. The construction is efficient and succeeds with overwhelming probability (in n).*

While in the rest of the paper (namely, for our coding scheme) we assume the edit-distance notion of Theorem 2, all our schemes work the same when the tree is replaced with a potent one; see the full version for further details.

Interactive-communication schemes tolerating insertions/deletions. Equipped with edit distance tree codes, we show a protocol that solves the *pointer jumping problem* over a noisy channel with insertions and deletions and exhibits linear communication complexity in the noiseless communication. Since the pointer jumping problem is complete for two-party interactive communication, this implies a coding scheme that can simulate any protocol over a channel that may introduce insertion/deletions. Specifically, in Sections 4 and 5 we show that for any $\varepsilon > 0$ and any noiseless protocol π and inputs x, y, there is a scheme that correctly simulates π (that is, produces the transcript $\pi(x, y)$ at both parties), withstands $1/18 - \varepsilon$ fraction of edit-corruptions, and has a linear communication complexity with respect to the communication of the protocol π.

▶ **Theorem 4.** *For any $\varepsilon > 0$, and for any binary (noiseless) protocol π with communication $CC(\pi)$, there exists a noise-resilient coding scheme with communication $O_\varepsilon(CC(\pi))$ that succeeds in simulating π as long as the adversarial edit-corruption rate is at most $1/18 - \varepsilon$.*

Our coding scheme and analysis follows ideas by Braverman and Rao [8] and by Braverman and Efremenko [6] – first focusing on channels with polynomial-size alphabet and then generalizing to channels with constant-size alphabet – however, the analysis in the light of insertions and deletions is more complicated and subtle. In particular, similar to [6], our analysis uses the notion of suffix distance for relating the effect of the noise to the progress of the simulation.

We note again that due to out-of-sync attacks, it is possible that the parties' belief of the "current" round of the protocol is different. In the worst case, while Alice reaches the end of the coding protocol (say, round N), it is possible that Bob has only reached round $(1-2\rho)N$, e.g., due to $2\rho N$ deletions in his received communication (ρ is the fraction of edit-corruptions in that instance). Therefore, if we wish to tolerate a ρ-fraction of edit-corruptions, it is imperative that the parties output the correct answer already at round $(1-2\rho)N$. Our coding scheme (Theorem 4) satisfies even this more strict requirement.

Finally, we show that for a family of rigid protocols, in which we require *both* parties to output the correct value at round $(1-2\rho)N$, then $\rho = 1/6$ is an upper bound on the admissible edit-corruption rate. Details can be found in the full version.

▶ **Theorem 5** (Informal). *If both parties are required to give output at round* $(1-2\rho)N$*, then no coding scheme of length* N *can tolerate an edit-corruption rate of* $\rho = 1/6$*.*

Closing the gap between the upper bound of 1/6, and the resilience 1/18 achieved by the scheme of Theorem 4 is left for future work.

2 Preliminaries

For any finite set S, we denote by $x \underset{U}{\leftarrow} S$ the case that x is uniformly distributed over S. All logarithms are taken to base 2. We denote the set $\{1, 2, \ldots, n\}$ by $[n]$. For a set Σ we denote $\Sigma^{\leq n} = \cup_{0 \leq i \leq n} \Sigma^i$, and $\Sigma^* = \cup_{i \geq 0} \Sigma^i$. Let $s \in \Sigma^l$ be a string of length $|s| = l$. For $1 \leq i \leq j \leq l$, we use $s[i]$ to denote the i-th symbol of s and $s[i..j]$ to denote the string $s[i] \circ s[i+1] \circ \cdots \circ s[j]$.

▶ **Definition 6** (Pointer Jumping Problem). Any communication protocol of T rounds where the parties alternately exchange bits can be reduced to the following *pointer jumping problem* $PJP(T)$: Let $\mathcal{T}$ be a binary tree of depth T. Alice's input X is a set of *consistent* edges leaving vertices at even depths. Bob's input Y is a set of consistent edges leaving odd-level vertices. A set of edges is *consistent* on a specified set of vertices, if it contains exactly one edge leaving every vertex in that specified set. Due to being consistent on all the nodes, $X \cup Y$ defines a unique root-to-leaf path. The parties' goal is to output this unique path. Note that T alternating rounds of communication suffice to compute this path, assuming noiseless channels.

▶ **Definition 7** (Protocols). An interactive protocol π for a function $f(x, y)$ is a distributed algorithm that dictates for each party, at every round, the next message (symbol) to send given the party's input and the messages received so far. Each transmitted symbol is assumed to be out of a fixed alphabet Σ. The protocol runs for N rounds (also called the length of the protocol), after which the parties give output. An instance of the protocol, on inputs x, y is said to be *correct* if both parties output $f(x, y)$ at the end of the protocol.

In this work we focus on *alternating* protocols, where as long as there is no noise, the protocol runs for $2N$ rounds in which Alice and Bob send symbols alternately. In the presence of ρ-fraction of edit corruptions (i.e., at most $2\rho N$ insertion/deletion errors), it is possible that some party receives only $N(1-2\rho)$ symbols throughout the protocol. We say that the protocol is correct in presence of ρ-fraction of edit corruption if both parties output $f(x, y)$ by round $N(1-2\rho)$ (according to their own round counting, which may differ from the count of the other party).

▶ **Definition 8** (String Matching). We say that $\tau = (\tau_1, \tau_2)$ is a string matching between a sent message sm and a received message rm (denoted $\tau : sm \to rm$), if $|\tau_1| = |\tau_2|$, $del(\tau_1) = sm$, $del(\tau_2) = rm$, and $\tau_1[i] \approx \tau_2[i]$ for all $i = 1, \ldots, |\tau_1|$. Here del is a function that deletes all the $*$'s in the string and two characters a and b satisfy $a \approx b$ if $a = b$ or one of a and b is $*$. We assume that $*$ is a special symbol that does not appear in sm and rm.

▶ **Definition 9** (Edit Distance). The edit distance of between $sm \in \Sigma^*$ and $rm \in \Sigma^*$ is defined as $ED(sm, rm) = \min_{\tau : sm \to rm} sc(\tau_1) + sc(\tau_2)$. Here $sc(\tau_1)$ is the number of $*$'s in τ_1.

▶ **Fact 10** (Triangle Inequality). *For any three strings* x, y, z*,* $ED(x, y) \leq ED(x, z) + ED(y, z)$*.*

▶ **Fact 11.** *For any two strings* x, y*,* $ED(x, y) = |x| + |y| - 2 \cdot LCS(x, y)$*. Here* $LCS(x, y)$ *is the longest common substring of* x *and* y*.*

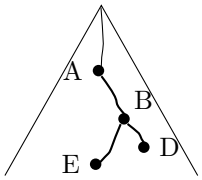


Figure 2 An illustration of lambda structure.

▶ **Lemma 12.** *Let $x \in \Sigma^m$ be some given string, $m \geq n$ and $|\Sigma| \geq 4$. For any constant $\alpha \in [0,1]$, $\Pr_{y \underset{U}{\leftarrow} \Sigma^n}[ED(x,y) \leq \alpha \cdot m] \leq |\Sigma|^{-\frac{(1-\alpha)m}{2}}$.*

The proof of the above Lemma, along with other missing proofs, are deferred to the full version of this work.

3 Edit-distance tree code

In this section we recall the notion of tree-codes [24] and provide a novel primitive, namely, the edit-distance tree-code.

▶ **Definition 13** (Prefix Code)**.** A prefix code $C : \Sigma_{in}^n \to \Sigma_{out}^n$ is a code such that $C(x)[i]$ only depends on $x[1..i]$. C can be also considered as a $|\Sigma_{in}|$-ary tree of depth n with symbols written on edges of the tree using an alphabet size $|\Sigma_{out}|$. On this tree, each tree path from the root of length l corresponds to a string $x \in \Sigma_{in}^l$ and the symbol written on the deepest edge of this path corresponds to $C(x)[l]$.

▶ **Definition 14** (α-bad Lambda)**.** We say that a prefix code C contains an α-bad lambda if when you consider this prefix code as a tree, there exist four tree nodes A,B,D,E such that $B \neq D$, $B \neq E$, B is D and E's common ancestor, A is B's ancestor or B itself, and $ED(AD, BE) \leq \alpha \cdot \max(|AD|, |BE|)$. Here AD and BE are strings of symbols along the tree path from A to D and the tree path from B to E. See Figure 2.

▶ **Definition 15** (Edit-distance Tree Code)**.** We say that a prefix code C: $\Sigma_{in}^n \to \Sigma_{out}^n$ is a α-edit-distance tree code if C does not contain an α-bad lambda.

Our main theorem in this section is the existence of edit-distance tree codes,

▶ **Theorem 16.** *For any $d \geq 2$, $n > 0$ and $0 < \alpha < 1$, there exists an α-edit-distance tree code of depth n with alphabet size $|\Sigma_{in}| = d$ and $|\Sigma_{out}| = (176 \cdot d)^{4/(1-\alpha)}$.*

Proof. We prove this theorem by induction on n. To this end we define a slightly stronger notion than α-bad lambda free, which we call "excellent". Intuitively, if a tree-code C is excellent, then with a good probability, C will not cause a bad lambda in a tree-code that contains C as a subtree. This would allow us to construct lambda-free trees of length n building on lambda-free trees with a smaller depth as subtrees.

▶ **Definition 17** (Potential Probability)**.** For any prefix code $C : \Sigma_{in}^n \to \Sigma_{out}^n$, and any $i \geq 0$, consider C as a tree and define a new (non-regular) tree C' by connecting a simple path of length i to the root of C, making the other end of this path the root of C'. Label each edge along the new path with a symbol from Σ_{out} chosen uniformly and independently.

The potential probability $P_i(C)$ is defined as the probability that the new tree C' has a α-bad lambda with $A =$ the root of C' and $B =$ the root of C.

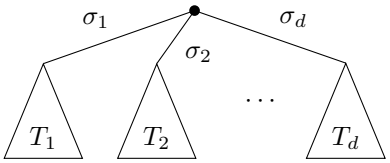
▶ **Definition 18** (Excellent)**.** For a constant $c_1 > 1$, which we will fix shortly, we say that a prefix code C is excellent if $\forall i \geq 0$, $P_i(C) \cdot (d \cdot c_1)^i < 1$ and C is α-bad-lambda-free.

In the proof, we are going to construct a set called S_n which would contain only excellent d-ary prefix codes of depth n with alphabet size $d^{O_\alpha(1)}$. Since excellent is a stronger notion than α-bad lambda-free, if we can construct S_n and show that for any $n > 0$ it is non-empty, then we are done. In each S_n, all codes use the same Σ_{in} and Σ_{out}. We have $|\Sigma_{in}| = d$ and $|\Sigma_{out}| = s$ where the conditions we put on s thorough the following proof are given by:

$$s = \max\left\{ \frac{\alpha d}{(1-\alpha)} \cdot (d \cdot c_1)^{\frac{\alpha}{1-\alpha}} + 1, d^2, (\frac{4d}{c_2})^{\frac{4}{1-\alpha}}, (c_1 \cdot d \cdot 4)^{\frac{4}{1-\alpha}} \right\}.$$

Here $c_1 = 44$, and $c_2 = \frac{1}{11}$. Therefore, taking $s \geq (176d)^{4/(1-\alpha)}$ satisfies all the above conditions.

Let us now inductively construct S_n. For notation convenience, let $S_0 = \{\text{a single node}\}$. For $n > 0$, let

$$\tilde{S}_n = \left\{ T = \right.$$


$$\left| \begin{array}{l} T_1, T_2, ..., T_d \in S_{n-1} \text{ and} \\ \sigma_1, \sigma_2, ..., \sigma_d \in \Sigma_{out} \end{array} \right\}.$$

and define $S_n = \{C \in \tilde{S}_n |\ C \text{ is excellet}\}$. From this definition, we directly have that every $C \in S_n$ is excellent, and we are only left to show that S_n is non-empty. Actually, we are going to prove the following claim by induction.

▶ **Claim 19.** *For all n, $\frac{|S_n|}{|\tilde{S}_n|} \geq c_2 = \frac{1}{11}$.*

Base case ($n = 1$): Consider the following set.

$$S_1' = \{C \mid C \text{ is a } d\text{-ary prefix code of depth 1 and } d \text{ different codewords}\}$$

We want to show that $S_1' \subseteq S_1$. For any $C \in S_1'$, it is clear that C does not have any α-bad lambda. Because in this depth 1 tree, one can only pick $A = B$ to be the root, and D, E to be some different leaves. Then $ED(AD, BE) = 2$ and $|AD| = |BE| = 1$. So $ED(AD, BE) > \alpha \cdot \max(|AD|, |BE|)$. Now let's consider the potential probability $P_i(C)$. Suppose there exists an α-bad lambda in the tree after adding a path of length i. Then in this α-bad lambda, B is the original root, AB is the path added to the root and D and E are two different leaves of the tree. There are two cases to consider:

1. $i = |AB| > \alpha/(1-\alpha)$: In this case, since $|BE| = 1$, $(|AD| - |BE|)/|AD| = 1 - \frac{1}{1+|AB|} > \alpha$. So it is not possible that $ED(AD, BE) < \alpha \cdot |AD|$. Thus $P_i(C) = 0$.
2. $i = |AB| \leq \alpha/(1-\alpha)$: In this case, let W be the event that one of the labels along the path AB equals to one of the d codewords of C. Clearly, when W does not happen, $ED(AD, BE) = |AD| + |BE| > \alpha \cdot |AD|$. It is also immediate that $\Pr[W] \leq i \cdot \frac{d}{s}$. We require $s > \frac{\alpha d}{(1-\alpha)} \cdot (d \cdot c_1)^{\frac{\alpha}{1-\alpha}}$, and get that $P_i(C) \cdot (d \cdot c_1)^i \leq \Pr[W] \cdot (d \cdot c_1)^i \leq \frac{\alpha d}{(1-\alpha)s} \cdot (d \cdot c_1)^{\frac{\alpha}{1-\alpha}} < 1$.

Therefore, we have proved that all the prefix codes in S_1' are excellent and therefore in S_1. Then because $s \geq d^2$, we get $\frac{|S_1|}{|\tilde{S}_1|} \geq \frac{|S_1'|}{|\tilde{S}_1|} = \frac{s(s-1)\times\cdots\times(s-d+1)}{s^d} \geq (1 - 1/d)^d > 1/11 = c_2$.

Inductive step: Suppose we already know that for $1 \le i < n$, $\frac{|S_i|}{|\tilde{S}_i|} \ge c_2$, and let us prove that $\frac{|S_n|}{|\tilde{S}_n|} \ge c_2$. We will use the following lemma, whose proof is quite straightforward.

▶ **Lemma 20.** *If for all* $1 \le i < n$, $\frac{|S_i|}{|\tilde{S}_i|} \ge c_2$. *Then for any* $x \in \Sigma_{out}^j$ *where* $0 < j \le n$ *and* $y \in \Sigma_{in}^n$, *it holds that* $\Pr_{C \xleftarrow[U]{} \tilde{S}_n}[C(y)[1..j] = x] \le c_2^{1-j} s^{-j}$.

In order to show that $\frac{|S_n|}{|\tilde{S}_n|} \ge c_2$, we can choose C randomly from $\tilde{S}_n$, and show that $\Pr[C \text{ is excellent}] \ge c_2$. To this end, consider the conditions of being excellent in turn:

1. Let's first consider $\Pr[P_i(C) \cdot (d \cdot c_1)^i \ge 1]$. By Lemma 20 and Lemma 12, we get

$$\begin{aligned}
&\mathbb{E}_{C \xleftarrow[U]{} \tilde{S}_n}[P_i(C)] \\
\le &\sum_{1 \le n_1, n_2 \le n} \sum_{\substack{x \in \Sigma_{in}^{n_1}, y \in \Sigma_{in}^{n_2}, \\ x[1] \ne y[1]}} \Pr_{\substack{z \xleftarrow[U]{} \Sigma_{out}^i \\ C \xleftarrow[U]{} \tilde{S}_n}} [ED(z \circ C(x), C(y)) \le \alpha \cdot \max(i + n_1, n_2)] \\
\le &\sum_{1 \le n_1, n_2 \le n} d^{n_1+n_2} \cdot c_2^{2-n_1-n_2} \Pr_{x \xleftarrow[U]{} \Sigma_{out}^{n_1}, y \xleftarrow[U]{} \Sigma_{out}^{n_2}, z \xleftarrow[U]{} \Sigma_{out}^i} [ED(z \circ x, y) \le \alpha \cdot \max(i + n_1, n_2)] \\
\le &\sum_{1 \le n_1, n_2 \le n} \left(\frac{d}{c_2}\right)^{n_1+n_2} s^{-\frac{1-\alpha}{2} \cdot \frac{n_1+n_2+i}{2}} \le \sum_{1 \le n_1, n_2 \le n} \left(\frac{d}{c_2} \cdot s^{-\frac{1-\alpha}{4}}\right)^{n_1+n_2} s^{-\frac{(1-\alpha)i}{4}} \\
\le &\sum_{1 \le n_1, n_2 \le n} \left(\frac{1}{4}\right)^{n_1+n_2} (c_1 \cdot d \cdot 4)^{-i} \le \sum_{j=2}^{\infty} \frac{j}{4^j} \cdot (c_1 \cdot d \cdot 4)^{-i} = \frac{7}{36} (c_1 \cdot d \cdot 4)^{-i}
\end{aligned}$$

   By Markov's inequality, $\Pr[P_i(C) \cdot (d \cdot c_1)^i \ge 1] \le \frac{7}{36} \cdot \frac{1}{4^i}$. Then $\sum_{i=0}^{\infty} \Pr[P_i(C) \cdot (d \cdot c_1)^i \ge 1] \le \frac{7}{36} \sum_{i=0}^{\infty} \frac{1}{4^i} = \frac{7}{27} < \frac{5}{11} = \frac{1-c_2}{2}$.
2. Now let's consider the event that C has an α-bad lambda with $A = B = root$. It is easy to see that this event is equivalent to $P_0(C) \ge 1$. Therefore it is covered by the previous case.
3. Now let's consider the event that C has an α-bad lambda with $A = root$ and $B \ne root$. By Lemma 20 and Definition 18, we have

$$\begin{aligned}
&\Pr[C \text{ has an } \alpha\text{-bad lambda with } A = root, B \ne root] \\
&\le \sum_{n_1 \le n} d^{n_1} \cdot c_2^{1-n_1} \cdot (d \cdot c_1)^{-n_1} \le \sum_{n_1=1}^{\infty} (\frac{1}{c_1 c_2})^{n_1} = \sum_{n_1=1}^{\infty} (\frac{1}{4})^{n_1} = \frac{1}{3} < \frac{5}{11} = \frac{1-c_2}{2}.
\end{aligned}$$

4. Finally let's consider the case that C has an α-bad lambda with $A \ne root$. It is easy to see that this event never happens because C's subtrees rooted at depth 1 in C are all excellent.

Therefore by union bound, $\Pr[C \text{ is excellent}]$ is at least

$$\begin{aligned}
&\ge 1 - \sum_{i=0}^{\infty} \Pr[P_i(C) \cdot (d \cdot c_1)^i \ge 1] - \Pr[C \text{ has an } \alpha\text{-bad lambda with } A = root, B \ne root] \\
&\ge 1 - \frac{1-c_2}{2} - \frac{1-c_2}{2} = c_2.
\end{aligned}$$

◀

3.1 Decoding of edit-distance tree codes

The decoding of a codeword $rm \in \Sigma_{out}^j$ via an edit-distance tree-code C amounts to finding a message whose encoding minimizes the suffix distance to rm, i.e., $\mathsf{DEC}(rm) = \arg\min_{m \in \Sigma_{in}^{\le n}} SD(rm, C(m))$, where the suffix distance $SD(\cdot, \cdot)$ is defined as follows.

▶ **Definition 21** (Suffix Distance). Given any two strings $sm, rm \in \Sigma^*$, the suffix distance between sm and rm is $SD(sm, rm) = \min_{\tau: sm \to rm} \max_{i=1}^{|\tau_1|} \frac{sc(\tau_1[i..|\tau_1|]) + sc(\tau_2[i..|\tau_2|])}{|\tau_1| - i + 1 - sc(\tau_1[i..|\tau_1|])}$.

The following lemma, which plays an important role in the analysis of the simulation protocols that we present in the next sections, shows that if a message sm is encoded by some α-edit-distance tree code and the received message rm satisfies $SD(sm, rm) \leq \frac{\alpha}{2}$, then the receiver can recover the entire sent message correctly.

▶ **Lemma 22.** *Let $C\colon \Sigma_{in}^n \to \Sigma_{out}^n$ be an α-edit-distance tree code, and let $rm \in \Sigma_{out}^m$ (m can be different from n). There exists at most one $sm \in \cup_{i=1}^n C(\Sigma_{in}^n)[1..i]$ such that $SD(sm, rm) \leq \frac{\alpha}{2}$.*

4 A coding scheme with a polynomial alphabet size

In this section, we show a protocol π that solves $PJP(T)$ in $O(T)$ rounds over channels with alphabet size $poly(T)$, and is resilient to (a constant fraction of) insertion/deletion errors. Since the $PJP(T)$ is complete for interactive communication, this implies that any binary protocol with T rounds can be simulated in $O(T)$ rounds over a channel with polynomially-large alphabet that corrupts at most a fraction $1/18 - \varepsilon$ of the transmissions. While this protocol does not exhibit a constant rate, it contains all the main ideas for the constant-rate protocol of Theorem 4, and thus we focus on this simple variant first. Then, in Section 5 we discuss how to reduce the alphabet size and achieve a protocol with $O(T)$ communication complexity with the same resilience guarantees.

Assume π has $2N$ alternating rounds, that is, Alice and Bob send N symbols each, assuming there are no errors. We would like the protocol to resist a fraction of ρ edit-corruptions, that is, the protocol should succeed as long as there are at most $2\rho N$ insertion/deletion errors. Due our assumption that the adversary never causes the protocol to "get stuck", this amounts to at most $2\rho N$ deletions, where each deletion is followed by an insertion.

We assume that Alice and Bob share some fixed α-edit-distance tree code $C : \Sigma_{in}^N \to \Sigma_{out}^N$ given by Theorem 16. We will set the values of N, α, and Σ_{in} later. Currently we only need to know $N = poly(T)$.

Let us begin with a high-level outline of the protocol π. The protocol basically progresses by sending edges in the tree $\mathcal{T}$ of the underlying $PJP(T)$, interactively constructing the joint path (similar to [8]). To communicate an edge e, the parties encode it as a pair of numbers (n, s), where $0 \leq n \leq N$ and $s \in \{1, 2, 3, 4\}$. The value n indicates the number of some previous round in which some edge e' was sent, and the value s determines e as the s-grandchild of e'. That is, we always send an edge e by linking it to an edge e' that was previously sent, such that e' is located two levels above in unique path leading from the root to e. If e does not have a grandparent (e.g., it is the at the first or second level in $\mathcal{T}$), we will set $n = 0$. Sometimes the parties have no edge to send, in which case they set $n = N$ and say that e is an empty edge. We take Σ_{in} to be all the possible encodings (n, s). As $0 \leq n \leq N$ and $1 \leq s \leq 4$, we have $|\Sigma_{in}| = poly(N) = poly(T)$. As $|\Sigma_{out}|$ is polynomial in $|\Sigma_{in}|$ (Theorem 16), we also have $|\Sigma_{out}| = poly(N) = poly(T)$.

The protocol π is described in Protocol 1. The description is for Alice side; Bob's part is symmetric. Here we explain more than the pseudocode on how to get $E(d_A)$. Basically d_A is a string of symbols in Σ_{in} and those symbols are in the form (n, s). $E(d_A)$ will be the set of edges these symbols in d_A represent. To get the edge each symbol (n, s) represents, we will first find the edge sent in n-th round and get its proper grandchild according to s. If n is not in the correct range then we consider d_A as not valid.

Protocol 1 The protocol π

Let $\mathcal{T}$ be given by $PJP(T)$. Recall that Alice's input is X. Assume the parties share some fixed α-edit-distance tree code $C : \Sigma_{in}^N \to \Sigma_{out}^N$.

Initially we set the counter $i = 0$. For any leaf node v in $\mathcal{T}$ we initialize a counter $s(v) = 0$. Run the following for N times.

1. $i \leftarrow i + 1$.
2. Receive a symbol $r_A[i]$ from the other party. (For Alice, if $i = 1$, skip this step)
3. Find $d_A \in \Sigma_{in}^*$ which minimizes $SD(d_A, r_A[1 \dots i])$.
4. If $E(d_A) \cup X$ has a unique path from the root in $\mathcal{T}$, do the following. Here $E(d_A)$ is the set of edges indicated by d_A, if d_A is not a valid string of symbols, $E(d_A) = \emptyset$.
   a. If this path reaches a leaf node v, then $s(v) \leftarrow s(v) + 1$.
   b. Let e be the deepest edge on the the unique path from root. If $e \in X$, and e is either an edge in the first or second level of $\mathcal{T}$ or e's grandparent has been sent, set $s_A[i]$ to be encoding of e, otherwise set $s_A[i]$ to be encoding of an empty edge.
5. If $E(d_A) \cup X$ does not have a unique path from the root in $\mathcal{T}$, set $s_A[i]$ to be encoding of an empty edge.
6. If $i = N(1 - 2\rho)$, output the leaf node v with the largest $s(v)$.
7. Send $C(s_A[1 \dots i])[i]$ to Bob.

We now analyze Protocol 1 and prove it resists up to $(1/18-\varepsilon)$-fraction of edit corruptions. Let N_A and N_B be the counter i of Alice and Bob respectively, when one of them reaches the end of the protocol π. Let $\tau_A = (\tau_1, \tau_2)$ be the string matching between $s_B[1..N_B]$ and $r_A[1..N_A]$ that is consistent with the protocol. Let $\tau_B = (\tau_3, \tau_4)$ be the string matching between $s_A[1..N_A]$ and $r_B[1..N_B]$. Recall that we use $sc(\tau)$ to denote the number of $*$'s in the string. By definition, $sc(\tau_1) + sc(\tau_3) \le 2\rho N$ and $sc(\tau_2) + sc(\tau_4) \le 2\rho N$.

In the analysis we count the number of rounds in which Alice correctly decodes the entire (current) set of edges sent by Bob. We call each such round a *good decoding*.

▶ **Definition 23** (Good Decoding)**.** When a party decodes a message, we say it is a *good decoding* if the decoded messages is exactly the one sent by the other side (i.e., $d_A = s_B[1..i]$ or $d_B = s_A[1..i]$, assuming the other side is at round i), and the symbol just received is not an adversarial insertion. If a decoding is not good, we call it a bad decoding.

In the following lemma show that as long as noise is small enough, there will be many rounds with good decodings. The proof can be found in the full version.

▶ **Lemma 24.** *Alice has at least* $N_A + (1 - \frac{2}{\alpha})sc(\tau_2) - (1 + \frac{2}{\alpha})sc(\tau_1)$ *good decodings. Bob has at least* $N_B + (1 - \frac{2}{\alpha})sc(\tau_4) - (1 + \frac{2}{\alpha})sc(\tau_3)$ *good decodings.*

After we have established that Alice and Bob will have many good decodings, we show that this implies they will have good progress in constructing their joint path in the underlying $PJP(T)$. Consider the $N_A + N_B$ decodings that happen during the protocol, and sort them in a natural order; we say that these decodings occur at "times" $t = 1, 2, \dots, N_A + N_B$. Note that the decodings need not be alternating – Eve's insertions and deletions may cause one party to perform several consecutive decodings while the other party does not receive any symbol, and performs no decoding. We also assume that for each decoding, the sending of the next symbol happens at the same "time" as the decoding. Let $e_A(t)$ be the set of edges Alice has sent at time t and $e_B(t)$ be the set of edges Bob has sent at time t. Let P be the correct path of length T of $PJP(T)$. Define $l(t)$ to be the length of the longest path from

the root using edges in $P \cap (e_A(t) \cup e_B(t))$. Basically $l(t)$ measures how much progress Alice and Bob have made. Let us also define $m(i)$ to be the first time t such that $l(t) \geq i$. For notation convenience, let $m(0) = 0$.

The following lemma shows that if the parties do not make progress, many bad decodings (and thus, many errors) must have occurred. The proof can be found in the full version.

▶ **Lemma 25.** *For $i = 0, \ldots, T-1$, if $m(i+1) \neq m(i)+1$, then during times $m(i)+1, \ldots, m(i+1)-1$, the following is true.*

1. *If i is odd, then there are no good decodings of Bob. The number of good decodings of Alice is at most the number of bad decodings of Bob.*
2. *If i is even, then there are no good decodings of Alice. The number of good decodings of Bob is at most the number of bad decodings of Alice.*

Combining the above lemmas, we get the main theorem for protocols with polynomial size alphabet.

▶ **Theorem 26.** *For any $\varepsilon > 0$, the protocol π of Protocol 1 with $N = \lceil \frac{T}{16\varepsilon} \rceil$, and a $(1-\varepsilon)$-edit tree code, solves $PJP(T)$ and is resilient to a $(1/18 - \varepsilon)$-fraction of edit corruptions.*

Proof. Set $\rho = \frac{1}{18} - \varepsilon$ and $\alpha = 1 - \varepsilon$. Let g_A be the number of good decodings of Alice, $b_A = N_A - g_A$ be the number of bad decodings of Alice. Similarly, let g_B be the number of good decodings of Bob, and let $b_B = N_B - g_b$. Recall that $sc(\tau_1) + sc(\tau_3) = sc(\tau_2) + sc(\tau_4) \leq 2\rho N$, then by Lemma 24, we have

$$b_A + b_B \leq \frac{2}{\alpha}(sc(\tau_1) + sc(\tau_2)) + sc(\tau_1) - sc(\tau_2) + \frac{2}{\alpha}(sc(\tau_3) + sc(\tau_4)) + sc(\tau_3) - sc(\tau_4),$$

thus $b_A + b_B \leq 8\rho N/\alpha$. Then we have,

$$\begin{aligned} g_A = N_A - b_A &\geq N_A - \frac{8\rho N}{\alpha} \geq (N_A - N(1-2\rho)) + N(1-2\rho) - \frac{8\rho N}{\alpha} \\ &\geq b_A + b_B - \frac{8\rho N}{\alpha} - (N_A - N(1-2\rho)) + N(1-2\rho) + \frac{8\rho N}{\alpha} \\ &> b_A + b_B + (N_A - N(1-2\rho)) + N(1-(1-18\varepsilon)(1+2\varepsilon)) \\ &\geq b_A + b_B + (N_A - N(1-2\rho)) + 16\varepsilon N \geq b_A + b_B + (N_A - N(1-2\rho)) + T. \end{aligned}$$

Similarly we have $g_B > b_A + b_B + (N_B - N(1-2\rho)) + T$.

Using Lemma 25 we deduce that by time $m(T)$ the number of good decodings Alice may have is bounded by $T + b_B$. From this point and on, every good decoding at Alice's side adds one vote for the correct leaf, making at least $g_A - (T + b_B) > b_A + (N_A - N(1-2\rho))$ votes for that node by the end of the protocol, and at least $b_A + 1$ votes until Alice reaches round $N(1-2\rho)$ when she gives her output. On the other hand, any wrong output can get at most b_A votes, thus Alice outputs the correct leaf node at round $N(1-2\rho)$. By a similar reasoning, Bob also outputs the correct leaf node when he reaches round $N(1-2\rho)$. ◀

5 A coding scheme with a constant alphabet size

Based on the protocol in Section 4, with some modifications, we obtain a protocol that has a constant size alphabet and a constant rate. To this end, we show how to encode each edge using varying-length encoding over a constant size alphabet. Although substantially more technically involved, this protocol is quite a straightforward extension of the protocol presented above. We thus defer the detailed analysis of this protocol to the full version.

▶ **Theorem 27.** *For any $\varepsilon > 0$, there exists a simulation protocol π' with $N = \lceil \frac{T}{\varepsilon^2} \rceil$, and a $(1-\varepsilon)$-edit distance tree code, solves $PJP(T)$ and is resilient to a $(1/18-\varepsilon)$-fraction of edit corruptions.*

Since $PJP(T)$ is complete for interactive communication, the above theorem proves Theorem 4.

References

1. Shweta Agrawal, Ran Gelles, and Amit Sahai. Adaptive protocols for interactive communication. *arXiv preprint arXiv:1312.4182*, 2013.
2. Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. In *Foundations of Computer Science, IEEE Annual Symposium on*, pages 160–166. IEEE Computer Society, 2012. doi:10.1109/FOCS.2012.22.
3. Zvika Brakerski, Yael Tauman Kalai, and Moni Naor. Fast interactive coding against adversarial noise. *J. ACM*, 61(6):35:1–35:30, December 2014. doi:10.1145/2661628.
4. Zvika Brakerski and Moni Naor. Fast algorithms for interactive coding. In *SODA'13: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 443–456, 2013.
5. Gilles Brassard, Ashwin Nayak, Alain Tapp, Dave Touchette, and Falk Unger. Noisy interactive quantum communication. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 296–305, 2014. doi:10.1109/FOCS.2014.39.
6. Mark Braverman and Klim Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. In *Foundations of Computer Science (FOCS), IEEE 55th Annual Symposium on*, pages 236–245, 2014.
7. Mark Braverman and Anup Rao. Towards coding for maximum errors in interactive communication. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC'11, pages 159–166, New York, NY, USA, 2011. ACM. doi:10.1145/1993636.1993659.
8. Mark Braverman and Anup Rao. Toward coding for maximum errors in interactive communication. *Information Theory, IEEE Transactions on*, 60(11):7248–7255, Nov 2014. doi:10.1109/TIT.2014.2353994.
9. Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Maximal noise in interactive communication over erasure channels and channels with feedback. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS'15, pages 11–20, New York, NY, USA, 2015. ACM. doi:10.1145/2688073.2688077.
10. Matthew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J. Schulman. Optimal coding for streaming authentication and interactive communication. In Ran Canetti and Juan A. Garay, editors, *CRYPTO'13*, volume 8043 of *LNCS*, pages 258–276. Springer Berlin, 2013. doi:10.1007/978-3-642-40084-1_15.
11. Matthew Franklin, Ran Gelles, Rafail Ostrovsky, and Leonard J. Schulman. Optimal coding for streaming authentication and interactive communication. *Information Theory, IEEE Transactions on*, 61(1):133–145, Jan 2015. doi:10.1109/TIT.2014.2367094.
12. Ran Gelles and Bernhard Haeupler. Capacity of interactive communication over erasure channels and channels with feedback. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'15, pages 1296–1311, 2015.
13. Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *Foundations of Computer Science, 2011 IEEE 52nd Annual Symposium on*, pages 768–777. IEEE Computer Society, 2011. doi:10.1109/FOCS.2011.51.

14 Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *Information Theory, IEEE Transactions on*, 60(3):1899–1913, March 2014. doi:10.1109/TIT.2013.2294186.

15 Mohsen Ghaffari and Bernhard Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. In *Foundations of Computer Science (FOCS), IEEE 55th Annual Symposium on*, pages 394–403, 2014.

16 Mohsen Ghaffari, Bernhard Haeupler, and Madhu Sudan. Optimal error rates for interactive coding I: Adaptivity and other settings. In *STOC'14: Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 794–803, 2014. doi:10.1145/2591796.2591872.

17 Bernhard Haeupler. Interactive channel capacity revisited. In *Foundations of Computer Science (FOCS), IEEE 55th Annual Symposium on*, pages 226–235, 2014.

18 Jørn Justesen. Class of constructive asymptotically good algebraic codes. *Information Theory, IEEE Transactions on*, 18(5):652–656, Sep 1972. doi:10.1109/TIT.1972.1054893.

19 Gillat Kol and Ran Raz. Interactive channel capacity. In *STOC'13: Proceedings of the 45th annual ACM Symposium on theory of computing*, pages 715–724, 2013. doi:10.1145/2488608.2488699.

20 Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

21 Rafail Ostrovsky, Yuval Rabani, and Leonard J. Schulman. Error-correcting codes for automatic control. *Information Theory, IEEE Transactions on*, 55(7):2931–2941, July 2009. doi:10.1109/TIT.2009.2021303.

22 Leonard J. Schulman. Communication on noisy channels: a coding theorem for computation. *Foundations of Computer Science, Annual IEEE Symposium on*, pages 724–733, 1992. doi:10.1109/SFCS.1992.267778.

23 Leonard J. Schulman. Deterministic coding for interactive communication. In *STOC'93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 747–756, New York, NY, USA, 1993. ACM. doi:10.1145/167088.167279.

24 Leonard J. Schulman. Coding for interactive communication. *IEEE Trans. Inf. Theory*, 42(6):1745–1756, 1996.

25 Leonard J. Schulman and David Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *Information Theory, IEEE Transactions on*, 45(7):2552–2557, Nov 1999. doi:10.1109/18.796406.

Amplifiers for the Moran Process*

Andreas Galanis[1], Andreas Göbel[2], Leslie Ann Goldberg[3], John Lapinskas[4], and David Richerby[5]

1 **Department of Computer Science, University of Oxford, Oxford, UK**
2 **Department of Computer Science, University of Oxford, Oxford, UK**
3 **Department of Computer Science, University of Oxford, Oxford, UK**
4 **Department of Computer Science, University of Oxford, Oxford, UK**
5 **Department of Computer Science, University of Oxford, Oxford, UK**

Abstract

The Moran process, as studied by Lieberman, Hauert and Nowak, is a randomised algorithm modelling the spread of genetic mutations in populations. The algorithm runs on an underlying graph where individuals correspond to vertices. Initially, one vertex (chosen uniformly at random) possesses a mutation, with fitness $r > 1$. All other individuals have fitness 1. During each step of the algorithm, an individual is chosen with probability proportional to its fitness, and its state (mutant or non-mutant) is passed on to an out-neighbour which is chosen uniformly at random. If the underlying graph is strongly connected then the algorithm will eventually reach *fixation*, in which all individuals are mutants, or *extinction*, in which no individuals are mutants. An infinite family of directed graphs is said to be *strongly amplifying* if, for every $r > 1$, the extinction probability tends to 0 as the number of vertices increases. Strong amplification is a rather surprising property – it means that in such graphs, the fixation probability of a uniformly-placed initial mutant tends to 1 even though the initial mutant only has a fixed selective advantage of $r > 1$ (independently of n). The name "strongly amplifying" comes from the fact that this selective advantage is "amplified". Strong amplifiers have received quite a bit of attention, and Lieberman et al. proposed two potentially strongly-amplifying families – superstars and metafunnels. Heuristic arguments have been published, arguing that there are infinite families of superstars that are strongly amplifying. The same has been claimed for metafunnels. We give the first rigorous proof that there is an infinite family of directed graphs that is strongly amplifying. We call the graphs in the family "megastars". When the algorithm is run on an n-vertex graph in this family, starting with a uniformly-chosen mutant, the extinction probability is roughly $n^{-1/2}$ (up to logarithmic factors). We prove that all infinite families of superstars and metafunnels have larger extinction probabilities (as a function of n). Finally, we prove that our analysis of megastars is fairly tight – there is no infinite family of megastars such that the Moran algorithm gives a smaller extinction probability (up to logarithmic factors). Also, we provide a counter-example which clarifies the literature concerning the isothermal theorem of Lieberman et al. A full version [11] containing detailed proofs is available at `http://arxiv.org/abs/1512.05632`. Theorem-numbering here matches the full version.

1998 ACM Subject Classification F.2 Analysis of Algorithms, G.3 Mathematics of Computing: Probability and Statistics: Markov processes, Probabilistic algorithms, Stochastic processes

Keywords and phrases Moran process, randomised algorithm on graphs, evolutionary dynamics

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.62

* The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007–2013) ERC grant agreement no. 334828. The paper reflects only the authors' views and not the views of the ERC or the European Commission. The European Union is not liable for any use that may be made of the information contained therein.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 62; pp. 62:1–62:13

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

1 Introduction

This paper is about a randomised algorithm called the Moran process. This algorithm was introduced in biology [20, 16] to model the spread of genetic mutations in populations. Similar algorithms have been used to model the spread of epidemic diseases, the behaviour of voters, the spread of ideas in social networks, strategic interaction in evolutionary game theory, the emergence of monopolies, and cascading failures in power grids and transport networks [2, 3, 12, 15, 17].

There has been past work about analysing the expected convergence time of the algorithm [7, 8]. In fact, the fast-convergence result of [7] implies that when the algorithm is run on an undirected graph, and the "fitness" of the initial mutation is some constant $r > 1$, there is an FPRAS for the "fixation probability", which is the probability that a randomly-introduced initial mutation spreads throughout the whole graph.

This paper answers an even more basic question, originally raised in [16], about the long-term behaviour of the algorithm when it is run on directed graphs. In particular, the question is whether there even exists an infinite family of (directed) graphs such that, when the algorithm is run on an n-vertex graph in this family, the fixation probability is $1 - o(1)$, as a function of n. A heuristic argument that this is the case was given in [16], but a counter-example to the argument (and to the hypothesized bound on the fixation probability) was given in [6]. A further heuristic argument (with a revised bound) was given in [14]. Here we give the first rigorous proof that there is indeed a family of "amplifiers" with fixation probability $1 - o(1)$. Before describing this, and the other results of this paper, we describe the model.

The Moran algorithm has a parameter r which is the fitness of "mutants". All non-mutants have fitness 1. The algorithm runs on a directed graph. In the initial state, one vertex is chosen uniformly at random to become a mutant. After this, the algorithm runs in discrete steps as follows. At each step, a vertex is selected at random, with probability proportional to its fitness. Suppose that this is vertex v. Next, an out-neighbour w of v is selected uniformly at random. Finally, the state of vertex v (mutant or non-mutant) is copied to vertex w.

If the graph is finite and strongly connected then with probability 1, the process will either reach the state where there are only mutants (known as *fixation*) or it will reach the state where there are only non-mutants (*extinction*). In this paper, we are interested in the probability that fixation is reached, as a function of the mutant fitness r, given the topology of the underlying graph. If $r < 1$ then the single initial mutant has lower fitness than the non-mutants that occupy every other vertex in the initial configuration, so the mutation is overwhelmingly likely to go extinct. If $r = 1$, an easy symmetry argument shows that the fixation probability is $\frac{1}{n}$ in any strongly connected graph on n vertices [7, Lemma 1].[1] Because of this, we restrict attention to the case $r > 1$. Perhaps surprisingly, a single advantageous mutant can have a very high probability of reaching fixation, despite being heavily outnumbered in the initial configuration.

A directed graph is said to be *regular* if there is some positive integer d so that the in-degree and out-degree of every vertex is d. In a strongly connected regular graph on n vertices, the fixation probability of a mutant with fitness $r > 1$ when the Moran algorithm is run is given by

$$\rho_{\mathrm{reg}}(r, n) = (1 - \tfrac{1}{r})/(1 - \tfrac{1}{r^n}), \tag{1}$$

[1] The result is stated in [7] for undirected graphs but the proof goes through unaltered for strongly connected directed graphs.

so the extinction probability of such a mutant is given by

$$\zeta_{\mathrm{reg}}(r,n) = (\tfrac{1}{r} - \tfrac{1}{r^n})/(1 - \tfrac{1}{r^n}). \tag{2}$$

Thus, in the limit, as n tends to ∞, the extinction probability tends to $1/r$. To see why (1) and (2) hold, note that, for every configuration of mutants, the number of edges from mutants to non-mutants is the same as the number of edges from non-mutants to mutants. Suppose that the sum of the individuals' fitnesses is W and consider an edge (u, v). If u is a mutant in the current state, it is selected to reproduce with probability r/W, and, if this happens, the offspring is placed at v with probability $1/d$. Similarly, if u is not a mutant, reproduction happens along (u, v) with probability $1/(dW)$. So, in any state, the number of mutants is r times as likely to increase at the next step of the process as it is to decrease. If we observe the number of mutants every time it changes, the resulting stochastic process is a random walk on the integers, that starts at 1, absorbs at 0 and n, increases with probability $\frac{r}{r+1}$ and decreases with probability $\frac{1}{r+1}$. It is well known that this walk absorbs at n with probability (1) and at 0 with probability (2). In particular, the undirected n-vertex complete graph is regular. Thus, by (2), its extinction probability tends to $1/r$.

When the Moran process is run on non-regular graphs the extinction probability may be quite a bit lower than $1/r$. Consider the undirected $(n + 1)$-vertex "star" graph, which consists of single centre vertex that is connected by edges to each of n leaves. In the limit as $n \to \infty$, the n-leaf star has extinction probability $\frac{1}{r^2}$ [16, 5]. Informally, the reason that the extinction probability is so small is that the initial mutant is likely to be placed in a leaf, and, at each step, a mutation at a leaf is relatively unlikely to be overwritten.

Lieberman et al. [16] refer to graphs which have smaller extinction probability than (2) (and therefore have larger fixation probability than (1)) as *amplifiers*. The terminology comes from the fact that the selective advantage of the mutant is being "amplified" in such graphs.

The purpose of this paper is to explore the long-term behaviour of the Moran process by quantifying how good amplifiers can be. For this, it helps to have some more formal definitions.

▶ **Definition 1.** Consider a function $\zeta(r,n)\colon \mathbb{R}_{>1} \times \mathbb{Z}_{\geq 1} \to \mathbb{R}_{\geq 0}$. An infinite family Υ of directed graphs is said to be *up-to-ζ fixating* if, for every $r > 1$, there is an n_0 (depending on r) so that, for every graph $G \in \Upsilon$ with $n \geq n_0$ vertices, the following is true: When the Moran process is run on G, starting from a uniformly-random initial mutant, the extinction probability is at most $\zeta(r,n)$.

Equation (2) demonstrates that the infinite family of strongly-connected regular graphs is up-to-ζ_{reg} fixating and since $\zeta_{\mathrm{reg}} \leq 1/r$, this family is also up-to-$1/r$ fixating. Informally, an infinite family of graphs is said to be *amplifying* if it is up-to-ζ fixating for a function $\zeta(r,n)$ which is "smaller" than $\zeta_{\mathrm{reg}}(r,n)$. Here is the formal definition.

▶ **Definition 2.** An infinite family of directed graphs is *amplifying* if it is up-to-ζ fixating for a function $\zeta(r,n)$ which, for every $r > 1$, satisfies $\lim_{n\to\infty} \zeta(r,n) < 1/r$.

The infinite family of graphs containing all undirected stars (which can be viewed as directed graphs with edges in both directions) is up-to-$\zeta(r,n)$ fixating for a function $\zeta(r,n)$ satisfying $\lim_{n\to\infty} \zeta(r,n) = 1/r^2$, so this family of graphs is amplifying.

Lieberman et al. [16] were interested in infinite families of digraphs for which the extinction probability tends to 0, prompting the following definition.

▶ **Definition 3.** An infinite family of directed graphs is *strongly amplifying* if it is up-to-ζ fixating for a function $\zeta(r,n)$ which, for every $r > 1$, satisfies $\lim_{n\to\infty} \zeta(r,n) = 0$.

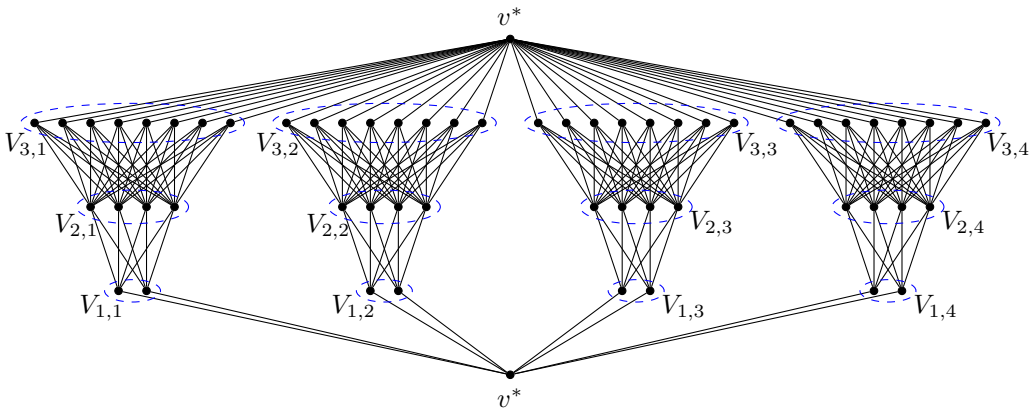


Figure 1 The metafunnel $\mathcal{G}_{3,4,2}$. All edges are directed downwards in the diagram and the centre vertex v^* is shown twice, once at the top and once at the bottom of the diagram. There are $\ell = 4$ copies of the basic unit, each of which consists of $k = 3$ levels $V_{1,j}$, $V_{2,j}$ and $V_{3,j}$, with $|V_{i,j}| = m^i = 2^i$.

Note that the infinite family of undirected stars is not strongly amplifying since the extinction probability of stars tends to $1/r^2$ rather than to 0.

Prior to this paper, there was no (rigorous) proof that a strongly amplifying family of digraphs exists (though there were heuristic arguments, as we explain later). Proving rigorously that there is an infinite family of directed graphs that is strongly amplifying for the Moran algorithm is one of our main contributions.

Lieberman et al. [16] produced good intuition about strong amplification and defined two infinite families of graphs – superstars and metafunnels – from which it turns out that strongly amplifying families can be constructed. It is extremely difficult to analyse the Moran process on these families, due mostly to the complexity of the graphs, and the difficulty of dealing with issues of dependence and concentration. Thus, all previous arguments have been heuristic. For completeness, we discuss these heuristic arguments in Section 6.

In this paper, we define a new family of digraphs called megastars. The definition of megastars is heavily influenced by the superstars of Lieberman et al. Our main theorem is the following.

▶ **Theorem 4.** *There exists an infinite family of megastars that is strongly amplifying.*

Megastars are not easier to analyse than superstars or metafunnels. The reason for our focus on this class of graphs is that it turns out to be provably better amplifying than any of the previously-proposed families. We will present several theorems along these lines. Before doing so, we define the classes of graphs.

2 Metafunnels, superstars and megastars

2.1 Metafunnels

We start by defining the metafunnels of [16]. Let k, ℓ and m be positive integers. The (k, ℓ, m)-*metafunnel* is the directed graph $\mathcal{G}_{k,\ell,m}$ defined as follows. (See Figure 1.)

The vertex set $V(\mathcal{G}_{k,\ell,m})$ is the union of $k+1$ disjoint sets $V_0, \ldots, V_k$. The set V_0 contains the single vertex v^* which is called the *centre vertex*. For $i \in [k]$, V_i is the union of ℓ disjoint

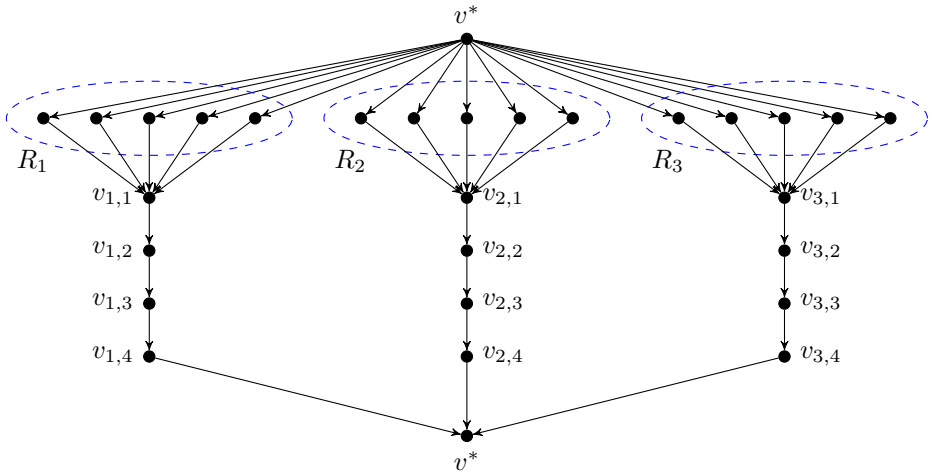


Figure 2 The superstar $\mathcal{S}_{4,3,5}$, with $\ell = 3$ reservoirs R_1, R_2 and R_3, each of size $m = 5$, connected by a path with $k = 4$ vertices to v^*. The centre vertex v^* is shown twice, at both the top and bottom of the diagram.

sets $V_{i,1}, \ldots, V_{i,\ell}$, each of which has size m^i. The edge set of $\mathcal{G}_{k,\ell,m}$ is

$$(V_0 \times V_k) \cup (V_1 \times V_0) \cup \bigcup_{i \in [k-1]} \bigcup_{j \in [\ell]} (V_{i+1,j} \times V_{i,j}) .$$

Lieberman et al. refer to metafunnels with $\ell = 1$ as "funnels".

2.2 Superstars

We next define the superstars of [16]. Let k, ℓ and m be positive integers. The (k, ℓ, m)-*superstar* is the directed graph $\mathcal{S}_{k,\ell,m}$ defined as follows. (See Figure 2.) The vertex set $V(\mathcal{S}_{k,\ell,m})$ of $\mathcal{S}_{k,\ell,m}$ is the disjoint union of ℓ size-m sets $R_1, \ldots, R_\ell$ (called *reservoirs*) together with $k\ell$ vertices $v_{1,1}, v_{1,2}, \ldots, v_{\ell,k}$ and a single centre vertex v^*. The edge set of $\mathcal{S}_{k,\ell,m}$ is given by

$$E(\mathcal{S}_{k,\ell,m}) = \bigcup_{i=1}^{\ell} \Big((\{v^*\} \times R_i) \cup (R_i \times \{v_{i,1}\}) \cup \{(v_{i,j}, v_{i,j+1}) \mid j \in [k-1]\} \cup \{(v_{i,k}, v^*)\} \Big).$$

2.3 Megastars

Finally, we define the new class of megastars, which turn out to be provably-better amplifiers than either metafunnels or superstars. The intuition behind the design of this class of graphs is that the path $v_{i,1}v_{i,2}\ldots v_{i,k}$ linking the i'th reservoir R_i of a superstar to the centre vertex v^* is good for amplifying but that a clique is even better.

Let k, ℓ and m be positive integers. The (k, ℓ, m)-megastar is the directed graph $\mathcal{M}_{k,\ell,m}$ defined as follows. (See Figure 3.) The vertex set $V(\mathcal{M}_{k,\ell,m})$ of $\mathcal{M}_{k,\ell,m}$ is the disjoint union of ℓ sets $R_1, \ldots, R_\ell$ of size m, called reservoirs, ℓ sets $K_1, \ldots, K_\ell$ of size k, called cliques, ℓ "feeder vertices" $a_1, \ldots, a_\ell$ and a single centre vertex v^*. The edge set of $\mathcal{M}_{k,\ell,m}$ consists of the following edges:

- an edge from v^* to every vertex in $R_1 \cup \cdots \cup R_\ell$,
- for each $i \in [\ell]$, an edge from each vertex in R_i to a_i,

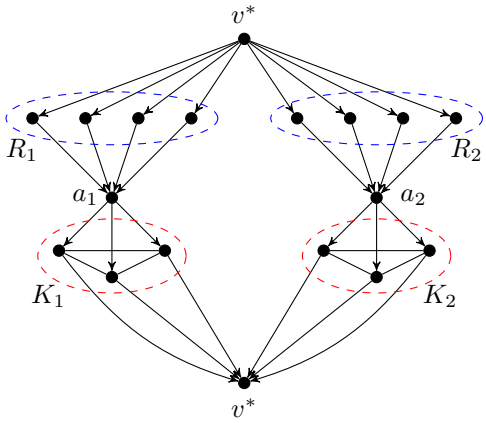


Figure 3 The megastar $\mathcal{M}_{3,2,4}$, with $\ell = 2$ reservoirs R_1 and R_2, each of size $m = 4$. Each reservoir R_i is attached, via the feeder vertex a_i to a clique of size $k = 3$. The centre vertex v^* is shown twice, once at the top and once at the bottom of the diagram. The edges within the cliques K_1 and K_2 are bidirectional.

- for each $i \in [\ell]$, an edge from a_i to each vertex in K_i,
- for each $i \in [\ell]$, edges in both directions between every pair of distinct vertices in K_i,
- an edge from every vertex in $K_1 \cup \cdots \cup K_\ell$ to v^*.

3 Our results

Our main result is that there is an infinite family of megastars that is strongly amplifying, so we start by defining this family. Although megastars are parameterised by three parameters, k, ℓ and m, the megastars in the family that we consider have a single parameter ℓ, so we define k and m to be functions of ℓ.

▶ **Definition 5.** Let $m(\ell) = \ell$ and $k(\ell) = \lceil (\log \ell)^{23} \rceil$. Let $\Upsilon_{\mathcal{M}} = \{\mathcal{M}_{k(\ell),\ell,m(\ell)} \mid \ell \in \mathbb{Z}, \ell \geq 2\}$.

Our main result can then be stated as follows.

▶ **Theorem 6.** *Let $\zeta_{\mathcal{M}}(r,n) = (\log n)^{23} n^{-1/2}$. The family $\Upsilon_{\mathcal{M}}$ is up-to-$\zeta_{\mathcal{M}}$ fixating.*

▶ **Corollary 7.** *The family $\Upsilon_{\mathcal{M}}$ is strongly amplifying.*

The proof of Theorem 6 requires a complicated analysis, accounting for dependencies and concentration. The theorem, as stated here, follows directly from Theorem 75 of the full version.

The reason that we studied megastars rather than the previously-introduced superstars and metafunnels is that megastars turn out to be provably better amplifying than any of the previously-proposed families. To demonstrate this, we prove the following theorem about superstars.

▶ **Theorem 8.** *Suppose that $\zeta(r,n)$ is any function such that, for any $r > 1$, we have $\lim_{n\to\infty} \zeta(r,n)(n \log n)^{1/3} = 0$. Then there is no infinite family of superstars that is up-to-ζ fixating.*

The function $\zeta_{\mathcal{M}}(r,n)$ from Theorem 6 certainly satisfies $\lim_{n\to\infty} \zeta_{\mathcal{M}}(r,n)(n \log n)^{1/3} = 0$, so Theorem 8 shows that there is no infinite family of superstars that is up-to-$\zeta_{\mathcal{M}}$ fixating.

More mundanely, it shows, for example, that if $\zeta(r,n) = n^{-1/3}(\log n)^{-1}$, then no infinite family of superstars is up-to-ζ fixating. Theorem 8 is a direct consequence of Theorem 29 of the full version.

Taken together, Theorems 6 and 8 show that superstars are worse amplifiers than megastars. We next show that metafunnels are *substantially* worse. We start with the following simple-to-state theorem.

▶ **Theorem 9.** *Fix any $\delta > 0$ and let $\zeta(r,n) = n^{-\delta}$. Then there is no infinite family of metafunnels that is up-to-ζ fixating.*

In fact, Theorem 9 can be strengthened by an exponential amount.

▶ **Theorem 10.** *Fix any $\epsilon < 1/2$ and let $\zeta(r,n) = n^{-1/(\log n)^{\epsilon}}$. Then there is no infinite family of metafunnels that is up-to-ζ fixating.*

Theorems 9 and 10 are a direct consequence of Theorem 47 in the full version. In fact, Theorem 47 provides even tighter bounds, as we will see in Section 5, though these are more difficult to state.

The theorems that we have already described (Theorem 6, Theorem 8 and Theorem 10) are the main contributions of the paper. Together, they show that there is a family of megastars that is strongly amplifying, and that there are no families of superstars or metafunnels that amplify as well. For completeness, we present a theorem showing that the analysis of Theorem 6 is fairly tight, in the sense that there are no infinite families of megastars that amplify substantially better than $\Upsilon_{\mathcal{M}}$ – in particular, our bound on extinction probability can only be improved by factors of $\log(n)$. It cannot be improved more substantially.

▶ **Theorem 11.** *Let $\zeta(r,n) = n^{-1/2}/(52r^2)$. There is no infinite family of megastars that is up-to-ζ fixating.*

Theorem 11 follows from Theorem 119 in the full version, which is straightforward. We conclude the paper with a digression which perhaps clarifies the literature. It is stated, and seems to be commonly believed, that an evolutionary graph (a weighted version of the Moran process – see Section 8 of the full version for details) is "isothermal" if and only if the fixation probability of a mutant placed uniformly at random is $\rho_{\text{reg}}(r,n)$. This belief seems to have come from an informal statement of the "isothermal theorem" in the main body of [16] (the formal statement in the supplementary material of [16] is correct, however) and it has spread, for example, as Theorem 1 of [22]. We clear this up by proving the following proposition, which says that there is a counter-example.

▶ **Proposition 12.** *There is an evolutionary graph that is not isothermal, but has fixation probability $\rho_{\text{reg}}(r,n)$.*

4 Proof techniques

As we have seen, it is easy to study the Moran process on a d-regular graph by considering the transition matrix of the corresponding Markov chain (which looks like a one-dimensional random walk). Highly symmetric graphs such as undirected stars can also be handled in a straightforward matter, by directly analysing the transition matrix. Superstars, metafunnels and megastars are more complicated, and the number of mutant-configurations is exponential, so instead we resort to dividing the process into phases, as is typical in the study of randomised algorithms and stochastic processes.

An essential and common trick in the area of stochastic processes (for example, in work on the voter model) is moving to continuous time. Instead of directly studying the discrete-time Moran process, one could consider the following natural continuous-time model which was studied in [8]: Given a set of mutants at time t, each vertex waits an amount of time before reproducing. For each vertex, the period of time before the next reproduction is chosen according to the exponential distribution with parameter equal to the vertex's fitness, independently of the other vertices. If the first vertex to reproduce is v at time $t + \tau$ then, as in the standard, discrete-time version of the process, one of its out-neighbours w is chosen uniformly at random, the individual at w is replaced by a copy of the one at v, and the time at which w will next reproduce is exponentially distributed with parameter given by its new fitness. The discrete-time process is recovered by taking the sequence of configurations each time a vertex reproduces. Thus, the fixation probability of the discrete-time process is *exactly the same* as the fixation probability of the continuous-time process. So moving to the continuous-time model causes no harm. As [8] explains, analysis can be easier in the continuous-time model because certain natural stochastic domination techniques apply in the continuous-time setting but not in the discrete-time setting.

It turns out that moving to the model of [8] does not suffice for our purposes. A major problem in our proofs is dealing with dependencies. In order to make this feasible, we instead study a continuous-time model (see "the clock process" in Section 3.1 of the full version) in which every edge of the underlying graph G is equipped with two Poisson processes, one of which is called a *mutant clock* and the other of which is called a *non-mutant clock*. The clock process is a stochastic process in which all of these clocks run independently. The continuous-time Moran process as defined in [8] can be recovered as a function of the times at which these clocks trigger.

Having all of these clocks available still does not give us the flexibility that we need. We say that a vertex u "spawns a mutant" in the Moran process if, at some point in time, u is a mutant, and it is selected for reproduction. We wish to be able to discuss events such as the event that the vertex u does not spawn a mutant until it has already been a mutant for some particular amount of time. In order to express such events in a clean way, making all conditioning explicit, we define additional stochastic processes called "star-clocks" (see Section 3.3 of the full version). All of the star-clocks run independently in the star-clock process.

In Section 3.4 of the full version we provide a coupling of the star-clock process with the Moran process. The coupling is valid in the sense that the two projections are correct – the projection onto the Moran process runs according to the correct distribution and so does the projection onto the star-clock process. The point of the coupling is that the different star-clocks can be viewed as having their own "local" times. In particular, there is a star-clock $M^*_{(u,v)}$ which controls reproductions from vertex u onto vertex v *during the time that u is a mutant*. Similarly, there is a star-clock $N^*_{(u,v)}$ which controls reproductions from vertex u onto vertex v during the time that u is a non-mutant. The coupling enables us to focus on relevant parts of the stochastic process, making all conditioning explicit.

The processes that we have described so far are all that we need to derive our upper bound on the fixation probability of superstars (Section 4 of the full version). This is the easiest of our main results.

Analysing the Moran process on metafunnels is more difficult. By design, the initial mutant x_0 is likely to be placed in the "top of a funnel" (in the set V_k). In the analysis, it is useful to be able to create independence by considering a "strain" of mutants which contains all of the descendants of a particular mutant spawned by x_0. Like the Moran process itself,

a strain can be viewed as a stochastic process depending on the triggering of the clocks. In order to facilitate the proof, we define a general notion of "mutant process" (Section 3.2 of the full version) – so the Moran process is one example of a mutant process, and a strain is another. The analysis of the Moran process on metafunnels involves both of these and also a third mutant process which is essentially the bottom level of a strain (called its head). Strains and heads-of-strains share some common properties, and they are analysed together as "colonies" in Section 5.4.1 of the full version. The analysis of the metafunnel is the technically most difficult of our results so we discuss it further in the next section.

Fortunately, the analysis of the megastar in Section 6 of the full version does not require three different types of mutant processes – it only requires one. The process that is considered is not the Moran process itself. Instead, it is a modification of the Moran process called the megastar process. The megastar process is similar to the Moran process except that the feeder vertices are forced to be non-mutants, except when their corresponding cliques are completely full or completely empty. It is easy to show (see the proof of Theorem 75 of the full version) that the fixation probability of the Moran process is at least as high as the fixation probability of the megastar process. However, the megastar process is somewhat easier to analyse because the cliques evolve somewhat independently. The proof of the key lemma (Lemma 77 of the full version) is fairly long but it is not conceptually difficult. The point is to prove that, with high probability, the cliques fill up and cause fixation.

5 Sketch of the analysis of metafunnels

The proofs of this paper are fairly technical, so the full version is 98 pages. In order to give a flavour, we give a brief sketch of the proof of Theorem 10 which is our most difficult result. We use $n = 1 + \ell \sum_{i=1}^{k} m^i$ to denote the number of vertices of a (k, ℓ, m)-metafunnel. We use X_t to denote the set of mutants at time t and x_0 to denote the initial mutant so $X_0 = \{x_0\}$. We prove the following theorem, which is slightly stronger than Theorem 10, and implies it.

▶ **Theorem 47.** *Let $r > 1$. Then there is a constant $c_r > 0$, depending on r, such that the following holds for all $k, \ell, m \in \mathbb{Z}_{\geq 1}$ such that the (k, ℓ, m)-metafunnel $\mathcal{G}_{k,\ell,m}$ has $n \geq 3$ vertices. Suppose that the initial state X_0 of the Moran process with fitness r is chosen uniformly at random from all singleton subsets of $V(\mathcal{G}_{k,\ell,m})$. The probability that the Moran process goes extinct is at least $e^{-\sqrt{\log r \cdot \log n}} (\log n)^{-c_r}$.*

Here is the sketch of the proof of Theorem 47. If $k = 1$ then $\mathcal{G}_{k,\ell,m}$ is a star and has extinction probability roughly $1/r^2$ so Theorem 47 follows easily. So for most of the proof (and the rest of this sketch) we assume $k \geq 2$. To prove the theorem, we divide the parameter space into two regimes.

In the (straightforward) first regime, $m \leq r^{\sqrt{\log_r n}}$. Since m is small, V_k is not too large compared to $V_0 \cup \cdots \cup V_{k-1}$. Thus, it is fairly likely that x_0 is born outside V_k, and becomes a non-mutant ("dies") before it can spawn a single mutant.

Most of the proof focusses on the second regime, where $m \geq r^{\sqrt{\log_r n}}$ which, since $n \geq \ell m^k$, implies $k \leq \sqrt{\log_r n}$. In this regime it is likely that a uniformly-chosen initial mutant x_0 is born in V_k (Lemma 49) so we assume that this is the case in most of the proof (and the rest of this sketch). The key lemma is Lemma 74 which shows that, in this case, it is (sufficiently) likely that x_0 dies before v^* spawns a mutant.

In more detail, we define a stopping time T_{pa} which is the first time t that one of the following occurs.

(A1) $X_t = \emptyset$, or
(A2) $|X_t|$ exceeds a given threshold m^* which is a polynomial in $\log n$, or
(A3) By time t, v^* has already become a mutant more than b^* times, where b^* is about half as large as its number ℓm of in-neighbours, or
(A4) t exceeds some threshold t_{max} which is (very) exponentially large in n.

The subscript "pa" is for "pseudo-absorption time" because (A1) implies that the Moran process absorbs by going extinct and (A2) is a prerequisite for absorbing by fixating. The proof of Lemma 74 shows that, with sufficiently high probability, (A2)–(A4) do not hold, and so the Moran process X must go extinct by time T_{pa}.

Conditioning makes it difficult to prove that (A2)–(A4) fail. To alleviate this, we divide the mutants into groups called "strains" which are easier to analyse. In particular, a strain contains all of the descendants of a particular mutant spawned by x_0. Informally, S^i is "born" at the i'th time at which x_0 spawns a mutant. It "dies" when all of the descendants of this spawn have died. It is "dangerous" if one (or more) of these descendants spawns a mutant onto v^* before T_{pa}.

Lemma 53 defines eight events $\mathcal{P}_1$–$\mathcal{P}_8$. These are defined in such a way that we can show (in the proof of Lemma 74) that if $\mathcal{P}_1$–$\mathcal{P}_8$ simultaneously occur, then (A2)–(A4) do not hold. The definitions are engineered in such a way that we can also show that it is fairly likely that they do hold simultaneously – this takes up most of the proof. Informally, the events are defined as follows.

$\mathcal{P}_1$: No star-clock $M^*_{(v^*,v)}$ triggers in the time interval $[0, 1]$.
$\mathcal{P}_2$: For some threshold $t_{x_0} < n$, the star-clock $N^*_{(v^*,x_0)}$ triggers in $[0, t_{x_0} - 2]$.
$\mathcal{P}_3$: v^* is a mutant for at most one unit of time up to time T_{pa}.
$\mathcal{P}_4$: The Moran process absorbs (either fixates or goes extinct) by time $t_{\mathsf{max}}/2$.
$\mathcal{P}_5$: Break $[0, t_{x_0}]$ into intervals of length $(\log n)^2$. During each interval, x_0 spawns at most $2r(\log n)^2$ mutants.
$\mathcal{P}_6$: Define s to be around $3rt_{x_0}$. Each of the strains $S^1, \ldots, S^s$ spawns at most $\log n$ mutants before T_{pa}.
$\mathcal{P}_7$: Each of the strains $S^1, \ldots, S^s$ dies within $(\log n)^2$ steps.
$\mathcal{P}_8$: At most $b^*/\log n$ of $S^1, \ldots, S^s$ are dangerous.

The rough sketch of Lemma 74 is as follows. $\mathcal{P}_1$ and $\mathcal{P}_3$ guarantee that v^* does not spawn a mutant until after T_{pa}. This together with $\mathcal{P}_2$ and $\mathcal{P}_3$ guarantees that the only mutants in the process before time T_{pa} are part of strains that are born before t_{x_0}. By $\mathcal{P}_5$, there are at most s such strains. By $\mathcal{P}_6$, each of these strains only has about $\log n$ mutants. Together with $\mathcal{P}_7$, this implies that (A2) does not hold at $t = T_{\mathsf{pa}}$. $\mathcal{P}_8$ and $\mathcal{P}_6$ imply that (A3) does not hold at $t = T_{\mathsf{pa}}$. Finally, $\mathcal{P}_4$ implies that (A4) does not hold at $t = T_{\mathsf{pa}}$.

The bulk of the proof involves showing (Lemma 53) that $\mathcal{P}_1$–$\mathcal{P}_8$ are sufficiently likely to simultaneously occur. Of these, $\mathcal{P}_3$–$\mathcal{P}_7$ are all so likely to occur that the probability that they do not occur can be subtracted off using a union bound (so conditioning on the other $\mathcal{P}_i$'s is not an issue). The majority of the failure probability comes from the probability that $\mathcal{P}_2$ does not occur. This is handled in the straightforward Lemma 54 which gives a lower bound on the probability that $\mathcal{P}_1$ and $\mathcal{P}_2$ both occur. The remaining event, $\mathcal{P}_8$, is sufficiently unlikely to occur that careful conditioning is required. This is (eventually) handled in Lemma 73, which shows that it is fairly likely to occur, conditioned on the fact that both $\mathcal{P}_1$ and $\mathcal{P}_2$ occur.

In order to get a good estimate on the probability that a strain is dangerous (in $\mathcal{P}_8$), we need to consider the number of mutants spawned from the "layer" of the strain closest to the centre vertex v^*. In order to do this, we define a new mutant process called the "head" of a

strain. Strains and heads of strains share some common properties, and they are analysed together as "colonies". Informally a "colony" is a mutant process Z whose mutants are in $V_1 \cup \cdots \cup V_{k-1}$ (and not in V_0 or V_k). Once a colony becomes empty, it stays empty. Since a colony is a mutant process but not necessarily a Moran process, vertices may enter and/or leave whenever a clock triggers but we say that the colony is *hit* when a vertex leaves a colony specifically because a non-mutant is spawned onto it in the underlying Moran process. We define the "spawning chain" Y^Z of a colony and show that it increases whenever the colony spawns a mutant and that it only decreases when the colony is hit. By analysing the jump chain of a spawning chain we are able to obtain the desired bounds on the probability that $\mathcal{P}_6$, $\mathcal{P}_7$ and $\mathcal{P}_8$ fail to occur.

6 Comparison with previous work

The Moran process is similar to a discrete version of directed percolation known as the contact process. There is a vast literature (e.g., [17, 10, 21, 9]) on the contact process and other related infection processes such as the voter model and susceptible-infected-susceptible (SIS) epidemic models. Often, the questions that are studied in these models are different from the question that we study here. For example, in voter systems [9] the two states (mutant/non-mutant) are often symmetric (similar to our $r = 1$ case) and the models are often studied on infinite graphs where the question is whether the process absorbs or not (both kinds of absorption, fixation and extinction, are therefore called "fixation" in some of this work). The particular details of the Moran process are very important for us because the details of the algorithm determine the long-term behaviour. For example, unlike the Moran process, in the contact process [4], the rate at which a node becomes a non-mutant is typically taken to be 1, whereas the rate at which a node becomes a mutant is proportional to the number of mutant neighbours. In the discrete-time versions of many commonly-studied models, a node is chosen randomly at each step for replacement, rather than (as in the Moran process) for reproduction. In any case, the important point for us is that the details of the algorithm are important – results do not carry over from one algorithm to the other. Therefore, we concentrate in this section on previous work about calculating the fixation probability of the Moran process itself.

Lieberman et al. [16] studied the fixation probability of the Moran process and introduced superstars and metafunnels. Intuitively, a superstar is a good amplifier because (as long as m is sufficiently large) the initial mutation is likely to be placed in a reservoir and (as long as ℓ is sufficiently large) this is unlikely to be killed quickly by the centre vertex. Moreover, the paths of a superstar are good for amplifying the selective advantage of mutants because, after the infection spreads from a reservoir vertex to the beginning of a path, it is likely to "pick up momentum" as it travels down the path, arriving at the centre vertex as a chain of $\Theta(k)$ mutants (which, taken together, are more likely to cause the centre to spread the infection than a single mutant arriving at the centre would be). As we have seen (Theorems 6 and 8) megastars are provably better for amplification than superstars. The reason for this is that a clique is substantially better than a path at doing this "amplification". Nevertheless, the amplifying properties of superstars strongly influenced our decision to study megastars.

Lieberman et al. [16, Equation (2)] claimed[2] that for sufficiently large n, the fixation probability of a superstar with parameter k tends to $1 - r^{-(k+2)}$, and that "similar results

[2] The reader who consults [16] might wonder why "k" as written in Equation (2) of [16] has become $k+2$ here. The reason is just that we use a slightly different parameterisation from that of [16]. To allow appropriate comparison, we describe all previous work using our parameterisation.

hold for the funnel and metafunnel". They provided a heuristic sketch-proof for the superstar, but not for the funnel or metafunnel. Hauert [13, Equation (5)] claims specifically that the fixation probability of funnels tends to $1 - r^{-(k+1)}$. As far as we know, no heuristic arguments have been given for funnels or metafunnels.

In any event, Díaz, Goldberg, Mertzios, Richerby, Serna and Spirakis [6] showed that the $1 - r^{-(k+2)}$ claim for superstars is incorrect for the case $k = 3$. In particular, for this case they showed that the fixation probability is at most $1 - \frac{r+1}{2r^5+r+1}$, which is less than the originally claimed value of $1 - r^{-5}$ for all $r \geq 1.42$.

Subsequently, Jamieson-Lane and Hauert [14, Equation (5)] made a more detailed but still heuristic[3] analysis of the fixation probability of superstars. They claim that for superstars with parameter k and with $\ell = m$, the fixation probability ρ_k has the following bounds for fixed $r > 1$,

$$1 - 1/(r^4(k-1)(1-\tfrac{1}{r})^2) - o(1) \leq \rho_k \leq 1 - 1/(r^4(k-1)) + o(1), \tag{3}$$

where the $o(1)$ terms tend to 0 as $\ell \to \infty$. They claim that their bounds are a good approximation as long as $k \ll \ell = m \sim \sqrt{n}$. It is not clear exactly what "$\ll$" means in this context. Certainly there are parameter regimes where $k = o(\ell)$ and $\ell = m \sim \sqrt{n}$ but nevertheless the extinction probability is much larger than the proposed upper bound $1/(r^4(k-1)(1-1/r)^2)$ from (3). For example, suppose that $\ell = m = k^{3/2}$. In this case (see Lemma 30 of the full version), the extinction probability is at least $k/(2r(m+k)) = 1/(2r(k^{1/2}+1))$ which is larger than $1/(r^4(k-1)(1-1/r)^2)$ for all sufficiently large k. Nevertheless, the bounds proposed by Jamieson-Lane and Hauert (3) seem to be close to the truth when k is very small compared to ℓ and m.

Our Corollary 34 of the full version identifies a wide class of parameters for which the extinction probability is provably at least $1/(1470r^4k)$. This is weaker than the suggested bound of Jamieson-Lane and Hauert by a factor of 1470. This constant factor is explained by the fact that our rigorous proof needs to show concentration of all random variables. We use lots of Chernoff bounds and other bounds on probabilities. In writing the proof, we optimised readability rather than optimising our constants, so our constants can presumably be improved.

There is recent work on other related aspects of the Moran process. For example, [18, 19] give fixation probability bounds on connected *undirected* graphs. [1] studies amplification with respect to adversarial or "temperature-based" placement of the initial mutation, in which the "temperature" of a vertex is proportional to the sum of all incoming edge weights. Also, [19] considers the extent to which the number of "good starts" for fixation can be bounded.

References

1 B. Adlam, K. Chatterjee, and M. A. Nowak. Amplifiers of selection. *Proceedings of the Royal Society A*, 471(2181), 2015.

2 Chalee Asavathiratham, Sandip Roy, Bernard Lesieutre, and George Verghese. The influence model. *IEEE Control Systems*, 21(6):52–64, 2001.

3 Eli Berge. Dynamic monopolies of constant size. *Journal of Combinatorial Theory, Series B*, 83(2):191–200, 2001.

[3] A full discussion of the argument of Jamieson-Lane and Hauert (and of the obstacles to making it a rigorous proof) are discussed in Section 9 of the full version.

4 Carol Bezuidenhout and Geoffrey Grimmett. The critical contact process dies out. *Ann. Probab.*, 18(4):1462–1482, 10 1990. doi:10.1214/aop/1176990627.

5 M. Broom and J. Rychtár. An analysis of the fixation probability of a mutant on special classes of non-directed graphs. *Proceedings of the Royal Society A*, 464:2609–2627, 2008.

6 J. Díaz, L. A. Goldberg, G. B. Mertzios, D. Richerby, M. J. Serna, and P. G. Spirakis. On the fixation probability of superstars. *Proceedings of the Royal Society A*, 469(2156):20130193, 2013.

7 J. Díaz, L. A. Goldberg, G. B. Mertzios, D. Richerby, M. J. Serna, and P. G. Spirakis. Approximating fixation probabilities in the generalised Moran process. *Algorithmica*, 69(1):78–91, 2014.

8 Josep Díaz, Leslie Ann Goldberg, David Richerby, and Maria J. Serna. Absorption time of the Moran process. *Random Structures and Algorithms*, To Appear.

9 Richard Durrett and Jeffrey E. Steif. Fixation results for threshold voter systems. *Ann. Probab.*, 21(1):232–247, 01 1993. doi:10.1214/aop/1176989403.

10 Rick Durrett. Some features of the spread of epidemics and information on random graphs. *Proceedings of the National Academy of Science*, 107(10):4491–4498, 2010.

11 Andreas Galanis, Andreas Göbel, Leslie Ann Goldberg, John Lapinskas, and David Richerby. Amplifiers for the Moran process. Preprint.

12 Herbert Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Interaction*. Princeton University Press, 2000.

13 C. Hauert. Evolutionary dynamics. In A. T. Skjeltorp and A. V. Belushkin, editors, *Proceedings of the NATO Advanced Study Institute on Evolution from Cellular to Social Scales*, pages 11–44. Springer, 2008.

14 A. Jamieson-Lane and C. Hauert. Fixation probabilities on superstars, revisited and revised. *Journal of Theoretical Biology*, 382:44–56, 2015.

15 David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proc. 9th ACM International Conference on Knowledge Discovery and Data Mining*, pages 137–146. ACM, 2003.

16 E. Lieberman, C. Hauert, and M. A. Nowak. Evolutionary dynamics on graphs. *Nature*, 433(7023):312–316, 2005. Supplementary material available at http://www.nature.com/nature/journal/v433/n7023/full/nature03204.html.

17 Thomas M. Liggett. *Stochastic Interacting Systems: Contact, Voter and Exclusion Processes*. Springer, 1999.

18 George B. Mertzios, Sotiris E. Nikoletseas, Christoforos Raptopoulos, and Paul G. Spirakis. Natural models for evolution on networks. *Theor. Comput. Sci.*, 477:76–95, 2013. doi:10.1016/j.tcs.2012.11.032.

19 George B. Mertzios and Paul G. Spirakis. Strong bounds for evolution in networks. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 669–680, 2013. doi:10.1007/978-3-642-39212-2_58.

20 P. A. P. Moran. Random processes in genetics. *Proceedings of the Cambridge Philosophical Society*, 54(1):60–71, 1958.

21 Devavrat Shah. Gossip algorithms. *Found. Trends Netw.*, 3(1):1–125, January 2009. doi:10.1561/1300000014.

22 P. Shakarian, P. Roos, and A. Johnson. A review of evolutionary graph theory with applications to game theory. *BioSystems*, 107(2):66–80, 2012.

Mixing Time of Markov Chains, Dynamical Systems and Evolution

Ioannis Panageas[*1] **and Nisheeth K. Vishnoi**[2]

1 **Georgia Institute of Technology, Atlanta, USA**
ioannis@gatech.edu

2 **École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland**
nisheeth.vishnoi@epfl.ch

Abstract

In this paper we study the mixing time of evolutionary Markov chains over populations of a fixed size (N) in which each individual can be one of m types. These Markov chains have the property that they are guided by a dynamical system from the m-dimensional probability simplex to itself. Roughly, given the current state of the Markov chain, which can be viewed as a probability distribution over the m types, the next state is generated by applying this dynamical system to this distribution, and then sampling from it N times. Many processes in nature, from biology to sociology, are evolutionary and such chains can be used to model them. In this study, the mixing time is of particular interest as it determines the speed of evolution and whether the statistics of the steady state can be efficiently computed. In a recent result [Panageas, Srivastava, Vishnoi, Soda, 2016], it was suggested that the mixing time of such Markov chains is connected to the geometry of this guiding dynamical system. In particular, when the dynamical system has a fixed point which is a global attractor, then the mixing is fast. The limit sets of dynamical systems, however, can exhibit more complex behavior: they could have multiple fixed points that are not necessarily stable, periodic orbits, or even chaos. Such behavior arises in important evolutionary settings such as the dynamics of sexual evolution and that of grammar acquisition. In this paper we prove that the geometry of the dynamical system can also give tight mixing time bounds when the dynamical system has multiple fixed points and periodic orbits. We show that the mixing time continues to remain small in the presence of several unstable fixed points and is exponential in N when there are two or more stable fixed points. As a consequence of our results, we obtain a phase transition result for the mixing time of the sexual/grammar model mentioned above. We arrive at the conclusion that in the interesting parameter regime for these models, i.e., when there are multiple stable fixed points, the mixing is slow. Our techniques strengthen the connections between Markov chains and dynamical systems and we expect that the tools developed in this paper should have a wider applicability.

1998 ACM Subject Classification G.3 Probability and Statistics, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Markov chains, Mixing time, Dynamical Systems, Evolutionary dynamics, Language evolution

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.63

* Ioannis Panageas was supported by NSF EAGER award grants CCF-1415496 and CCF-1415498. Part of this work was done while he was visiting Simons Institute for the Theory of Computing at Berkeley.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 63; pp. 63:1–63:14

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

1 Introduction

Evolutionary Markov chains and mixing time

In this paper we study Markov chains that arise in the context of evolution and which have also been used to model a wide variety of social, economical and cultural phenomena, see [18]. Typically, in such Markov chains, each state consists of a population of size N where each individual is of one of m *types*. Thus, the state space Ω has size $\binom{N+m-1}{m-1}$. At a very high level, in each iteration, the different types in the current generation reproduce according to their *fitnesses*, the reproduction could be *asexual* or *sexual* and have *mutations* that transform one type into another. This gives rise to an intermediate population that is subjected to the force of *selection*; a sample of size N is selected giving us the new generation. The specific way in which each of the *reproduction*, *mutation* and *selection* steps happen determine the transition matrix of the corresponding Markov chain. The size of the population (N), the number of types (m), the fitness of each type ($\{a_i \geq 0 : i \in [m]\}$), and the probabilities of mutation of one type to another ($\{Q_{ij} \geq 0 : i, j \in [m]\}$) are the parameters of the model. If we make the natural assumption that all the fitnesses are strictly positive and there is a non-zero probability of mutating from any type to the other, $Q_{ij} > 0$ for all $i, j \in [m]$, then the underlying chain is ergodic and has a unique steady state.

Most questions in evolution reduce to understanding the statistical properties of the steady state of an evolutionary Markov chain and how it changes with its parameters. However, in general, there seems to be no way to compute the desired statistical properties other than to sample from (close to) the steady state distribution by running the Markov chain for sufficiently long [5]. In the chains of interest, while there is an efficient way to sample the next state given the current state, typically, the state space is huge[1] This is captured by the notion of its *mixing time*. The mixing time of a Markov chain, t_{mix}, is defined to be the smallest time t such that for all $x \in \Omega$, the distribution of the Markov chain starting at x after t-time steps is within an ℓ_1-distance of $1/4$ of the steady state.[2] Apart from dictating the computational feasibility of sampling procedures, the mixing time also gives us the number of generations required to reach a steady state; an important consideration for validating evolutionary models [24, 5]. However, despite the importance of understanding when an evolutionary Markov chain mixes fast (i.e., is significantly smaller than the size of the state space), until recently, there has been a lack of rigorous mixing time bounds for the full range of evolutionary parameters in even in the simplest of stochastic evolutionary models; see [7, 9, 8] for results under restricted assumptions and [5, 25] for an extended discussion on mixing time bounds in evolutionary Markov chains.

The expected motion of a Markov chain

In a recent result [20], a new approach for bounding the mixing time of such Markov chains was suggested. Towards this, it is convenient to think of each state of an evolutionary Markov chain as a vector which captures the fraction of each type in the current population. Thus, each state is a point the m-dimensional probability simplex Δ_m,[3] and we can think of $\Omega \subseteq \Delta_m$. If $\mathbf{X}^{(t)}$ is the current state, then we define the *expected motion* of the chain at $\mathbf{X}^{(t)}$

[1] For example, even when $m = 40$ and the population is of size $10,000$, the number of states is more than 2^{300}, i.e., more than the number of atoms in the universe!

[2] It is well-known that if one is willing to pay an additional factor of $\log 1/\varepsilon$, one can bring down the error from $1/4$ to ε for any $\varepsilon > 0$; see [14].

[3] The probability simplex Δ_m is defined to be $\{p \in \mathbb{R}^m : p_i \geq 0 \ \forall i, \ \sum_i p_i = 1\}$.

to be the function

$$f(\mathbf{X}^{(t)}) \coloneqq \mathbb{E}\left[\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}\right]$$

where the expectation is over one step of the chain. Notice that while the domain of f is Ω, its range could be a larger subset of Δ_m. *What can the expected motion of a Markov chain tell us about the mixing time of a Markov chain?* Of course, without imposing additional structure on the Markov chain, we do not expect a very interesting answer. However, [20] suggested that, the expected motion can be helpful in establishing mixing time bounds, at least in the context of evolutionary dynamics. The first observation is that, while in the case of general Markov chains, the expected motion function is only defined at a subset of Δ_m, in the case of evolutionary Markov chains, the expected motion turns out to be a *dynamical system*; defined on *all* points of Δ_m. Further, the Markov chain can be recovered from the dynamical system: it can be shown that given a state $\mathbf{X}^{(t)}$ of the Markov chain, one can generate $\mathbf{X}^{(t+1)}$ *equivalently* by computing the probability distribution $f(\mathbf{X}^{(t)})$ and taking N i.i.d. samples from it. Subsequently, their main result is to prove that if this dynamical system has a *unique stable fixed point* and also all the trajectories converge to this point, then the evolutionary Markov chain mixes rapidly. Roughly, this is achieved by using the geometry of the dynamical system around this unique fixed point to construct a *contractive coupling*. As an application, this enabled them to establish rapid mixing for evolutionary Markov chains in which the reproduction is *asexual*.

What if the limit sets of the expected motion are complex: multiple fixed points – some stable and some unstable, or even periodic orbits? Not only are these natural mathematical questions given the previous work, such behavior arises in several important evolutionary settings; e.g., in the case when the reproduction is sexual (see [3, 17] and Chapter 20 in [11]) and an equivalent model for how children acquire grammar [19, 12]. While we describe these models later, we note that, as one changes the parameters of the model, the limit sets of the expected motion can exhibit the kind of complex behavior mentioned above and a finer understanding of how they influence the mixing time is desired.

Our contribution

In this paper we introduce prove that the geometry of the dynamical system can also give tight mixing time bounds when the dynamical system has multiple fixed points and periodic orbits. This completes the picture left open by the previous work. Recall that [20] proved that when there is a unique stable fixed point, then the mixing time is about $O(\log N)$ when N is large compared to the parameters of the model. We complement their result by proving the following mixing time bounds which depend on the structure of the limit sets of the expected motion:

- One stable fixed point and multiple unstable fixed points – the mixing time is $O(\log N)$, see Theorem 6.
- Multiple stable fixed points – the mixing time is $e^{\Omega(N)}$, see Theorem 7.
- Periodic orbits – the mixing time is $e^{\Omega(N)}$, see Theorem 8.

Thus, we can prove that despite the presence of unstable fixed points the mixing time continues to remain small. On the other hand, if there are two or more stable fixed points, the mixing time can undergo a phase transition and become exponential in N.

As an application, we characterize the mixing time of the dynamics of *grammar acquisition* (or, as explained later, sexual evolution). This Markov chain attempts to model a fascinating and important problem in linguistics; to understand the mechanism by which a child acquires

the capacity to comprehend a language and effectively communicate [16, 10]. Here, a parameter of interest is the *mutation rate* τ which is to be thought of as quantifying the *error of learning*; see Section 2.1. Corresponding to this, the probabilities of mutation $Q_{ij} = \tau$ for all $i \neq j$ and $Q_{ii} = 1 - (m-1)\tau$. We first prove that there is a critical value where the expected motion dynamical system goes through a bifurcation from multiple stable fixed points to one stable fixed point. Our main results then imply that for $\tau < \tau_c$ the mixing time is exponential in N and for $\tau > \tau_c$ it is $O(\log N)$, see Theorem 9. Thus, we arrive at the conclusion that, in the interesting parameter regime for an important and natural dynamics, i.e., when there is a stable fixed point other than the uniform one, the mixing is very slow.

Technically, there have been several influential works in the probability literature that use dynamical systems to analyze stochastic processes, see for example [22, 2, 26, 15]. While the techniques used in these results bear some similarity to ours, to the best of our knowledge, ours is the first paper which studies the question of how the *mixing time* of a Markov chain behaves as a function of the guiding dynamical system formally.

Organization of the paper

The rest of the paper is organized as follows. In Section 2 we present the formal statement of our main theorems and the model of grammar acquisition/sexual evolution. In Section 4, we present an overview of the proofs of our main theorem. Due to space constraints, details of the proofs have been omitted from this version and appear in the full version of this paper.

2 Formal statement of our results

In this section we present formal statements of our main results. We begin by introducing the required notation and preliminaries.

Notation

We use boldface letters, e.g., $\mathbf{x}$, to denote column vectors (points), and denote a vector's i^{th} coordinate by x_i. We use $\mathbf{X}$ and $\mathbf{Y}$ (often with time superscripts and coordinate subscripts as appropriate) to denote random vectors. For a function $f : \Delta_m \to \Delta_m$, by f^n we denote the composition of f with itself n times, namely $\underbrace{f \circ f \circ \cdots \circ f}_{n \text{ times}}$. We use $J_f[\mathbf{x}]$ to denote the Jacobian matrix of f at the point $\mathbf{x}$. When the function f is clear from the context, we omit the subscript and simply denote it by $J[\mathbf{x}]$. Similarly, we sometimes use $J^n[\mathbf{x}]$ to denote the Jacobian of f^n at $\mathbf{x}$. We denote by $\mathrm{sp}\,(A)$ the spectral radius of a matrix A and by $(A\mathbf{x})_i$ the sum $\sum_j A_{ij}x_j$.

Dynamical Systems

Let $\mathbf{x}^{(t+1)} = f(\mathbf{x}^{(t)})$ be a *discrete time* dynamical system with update rule $f : \Delta_m \to \Delta_m$. The point $\mathbf{z}$ is called a *fixed point* of f if $f(\mathbf{z}) = \mathbf{z}$. We call a fixed point $\mathbf{z}$ *stable* if, for the Jacobian $J[\mathbf{z}]$ of f, it holds that $\mathrm{sp}\,(J[\mathbf{z}]) < \rho < 1$. A sequence $(f^t(\mathbf{x}^{(0)}))_{t\in\mathbb{N}}$ is called a *trajectory* of the dynamics with $x^{(0)}$ as starting point. A common technique to show that a dynamical system converges to a fixed point is to construct a function $P : \Delta_m \to \mathbb{R}$ such that $P(f(\mathbf{x})) > P(\mathbf{x})$ unless $\mathbf{x}$ is a fixed point. We call P a *potential* function. One of our results deals with dynamical systems that have stable periodic orbits.

▶ **Definition 1.** $C = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ is called a *periodic orbit* of size k if $\mathbf{x}_{i+1} = f(\mathbf{x}_i)$ for $1 \leq i \leq k-1$ and $f(\mathbf{x}_k) = \mathbf{x}_1$. If $\mathrm{sp}\left(J_{f^k}[\mathbf{x}_1]\right) < \rho < 1$, we call C a *stable* periodic orbit (we also use the terminology *stable limit cycle*).

▶ Remark. Since $f : \Delta_m \to \Delta_m$ and hence $\sum_i f_i(\mathbf{x}) = 1$ for all $\mathbf{x} \in \Delta_m$, if we define $h_i(\mathbf{x}) = \frac{f_i(\mathbf{x})}{\sum_i f_i(\mathbf{x})}$ so that $h(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \Delta_m$, we get that $\sum_i \frac{\partial h_i(\mathbf{x})}{\partial x_j} = 0$ for all $j \in [m]$. This means without loss of generality we can assume that the Jacobian $J[\mathbf{x}]$ of f has $\mathbf{1}^\top$ (the all-ones vector) as a left eigenvector with eigenvalue 0.

The definition below quantifies the instability of a fixed point as is standard in the literature. Essentially, an α unstable fixed point is repelling in any direction.

▶ **Definition 2.** Let $\mathbf{z}$ be a fixed point of a dynamical system f. The point $\mathbf{z}$ is called *α-unstable* if $|\lambda_{\min}(J[\mathbf{z}])| > \alpha > 1$ where $\lambda_{\min}$ corresponds to the minimum eigenvalue of the Jacobian of f at the fixed point $\mathbf{z}$, excluding the eigenvalue 0 that corresponds to the left eigenvector $\mathbf{1}^\top$.

Stochastic Evolution

▶ **Definition 3.** Given an $f : \Delta_m \to \Delta_m$ which is smooth,[4] and a population parameter N, we define a Markov chain called the *stochastic evolution guided by f* as follows. The state at time t is a probability vector $\mathbf{X}^{(t)} \in \Delta_m$. The state $\mathbf{X}^{(t+1)}$ is then obtained in the following manner. Define $\mathbf{Y}^{(t)} = f(\mathbf{X}^{(t)})$. Obtain N independent samples from the probability distribution $\mathbf{Y}^{(t)}$, and denote by $\mathbf{Z}^{(t)}$ the resulting counting vector over $[m]$. Then

$$\mathbf{X}^{(t+1)} \coloneqq \frac{1}{N}\mathbf{Z}^{(t)} \quad \text{and therefore } \mathbb{E}[\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}] = f(\mathbf{X}^{(t)}).$$

We call f the *expected motion* of the stochastic evolution.

▶ **Definition 4** (Smooth contractive evolution)**.** A function $f : \Delta_m \to \Delta_m$ is said to be a *smooth contractive evolution* if it is *smooth*[4], has a *unique* fixed point $\mathbf{z}$ in the interior of Δ_m, this unique point is *stable*, and, for every $\varepsilon > 0$, there exists an ℓ such that for any $\mathbf{x} \in \Delta_m$, it holds that $\left\|f^\ell(\mathbf{x}) - \mathbf{z}\right\|_1 < \varepsilon$ (i.e., f converges to the fixed point).

The main result in [20] was Theorem 5 below. This theorem gives a bound on the mixing time of a stochastic evolution guided by a function f that satisfies Definition 4.

▶ **Theorem 5** (Main theorem in [20])**.** *Let f be a smooth contractive evolution, and let $\mathcal{M}$ be the stochastic evolution guided by f on a population of size N. Then, the mixing time of $\mathcal{M}$ is $O(\log N)$.*

Our Results

Given a dynamical system f, one of the main questions that one can ask is does it converge, and if so, how fast. In general, if the behavior of a system is non-chaotic, we expect the system to reach some steady state (e.g., a fixed point or periodic orbit). This steady state might be some (local) optimum solution to a non-linear optimization problem. Therefore, it is important to understand what traits make a dynamical system converge fast. The

[4] For our purposes, we call a function f is *smooth* if it is twice differentiable in the relative interior of Δ_m with bounded second derivative.

existence of many fixed points which are unstable can slow down the speed of convergence of a dynamical system. In the case of the stochastic evolution guided by f, one would expect the existence of multiple unstable fixed points to similarly slow down the mixing time. Nevertheless, our Theorem 6 shows rapid mixing in the presence of α-unstable fixed points. Additionally, we change the assumption *convergence to the fixed point* in 5 to the assumption that for all $\mathbf{x} \in \Delta_m$ the limit $\lim_{t\to\infty} f^t(\mathbf{x})$ exists and is equal to some fixed point $\mathbf{z}$, i.e., as in 5, there are no limit cycles.

▶ **Theorem 6.** *Let $f : \Delta_m \to \Delta_m$ be twice differentiable in the interior of Δ_m with bounded second derivative. Assume that $f(\mathbf{x})$ has a finite number of fixed points $\mathbf{z}_0, \ldots, \mathbf{z}_l$ in the interior, where $\mathbf{z}_0$ is a stable fixed point, i.e., $\mathrm{sp}\,(J[\mathbf{z}_0]) < \rho < 1$ and $\mathbf{z}_1, \ldots, \mathbf{z}_l$ are α-unstable fixed points ($\alpha > 1$). Furthermore, assume that $\lim_{t\to\infty} f^t(\mathbf{x})$ exists for all $\mathbf{x} \in \Delta_m$. Then, the stochastic evolution guided by f has mixing time $O(\log N)$.*

In our second result, we allow f to have multiple stable fixed points (in addition to any number of unstable fixed points). For this setting, we prove that the stochastic evolution guided by f has mixing time $e^{\Omega(N)}$. Our phase transition result on a linguistic/sexual evolution model discussed in Section 2.1 relies crucially on Theorem 7.

▶ **Theorem 7.** *Let $f : \Delta_m \to \Delta_m$ be continuously differentiable in the interior of Δ_m. Assume that $f(\mathbf{x})$ has at least two stable fixed points in the interior $\mathbf{z}_1, \ldots, \mathbf{z}_l$, i.e., $\mathrm{sp}\,(J[\mathbf{z}_i]) < \rho_i < 1$ for $i = 1, 2, \ldots, l$. Then, the stochastic evolution guided by f has mixing time $e^{\Omega(N)}$.*

Finally, we allow f to have a stable limit cycle. We prove that in this setting the stochastic evolution guided by f has mixing time $e^{\Omega(N)}$. This result seems important for evolutionary dynamics as periodic orbits often appear [23, 21].

▶ **Theorem 8.** *Let $f : \Delta_m \to \Delta_m$ be continuously differentiable in the interior of Δ_m. Assume that $f(\mathbf{x})$ has a stable limit cycle with points $\mathbf{w}_1, \ldots, \mathbf{w}_s$ of size $s \geq 2$ in the sense that $\mathrm{sp}\,(\prod_{i=1}^{s} J[\mathbf{w}_{s-i+1}]) < \rho < 1$. Then the stochastic evolution guided by f has mixing time $e^{\Omega(N)}$.*

2.1 Dynamics of grammar acquisition and sexual evolution

We begin by describing the evolutionary processes for grammar acquisition and sexual evolution. As we will explain, the two turn out to be identical and hence we primarily focus on the model for grammar acquisition in the remainder of the paper.

The starting point of the model is Chomsky's *Universal Grammar* theory [4].[5] In his theory, language learning *is facilitated by a predisposition that our brains have for certain structures of language.* This universal grammar (UG) is believed to be innate and embedded in the neuronal circuitry. Based on this theory, an influential model for how children acquire grammar was given by appealing to evolutionary dynamics for infinite and finite populations respectively in [19] and [12]. We first describe the infinite population model, which is a dynamical system that guides the stochastic, finite population model. Each individual speaks exactly one of the m grammars from the set of inherited UGs $\{G_1, \ldots, G_m\}$; denote by x_i the fraction of the population using G_i. The model associates a *fitness* to every individual on the basis of the grammar she *and others* use. Let A_{ij} be the probability that a person who speaks grammar j understands a randomly chosen sentence spoken by an individual

[5] Like any important problem in the sciences, Chomsky's theory is not uncontroversial; see [10] for an in-depth discussion.

using grammar i. This can be viewed as the fraction of sentences according to grammar i that are also valid according to grammar j. Clearly, $A_{ii} = 1$. The pairwise *compatibility* between two individuals speaking grammars i and j is $B_{ij} \coloneqq \frac{A_{ij}+A_{ji}}{2}$, and the fitness of an individual using G_i is $f_i \coloneqq \sum_{j=1}^{m} x_j B_{ij}$, i.e., the probability that such an individual is able to meaningfully communicate with a randomly selected member of the population.

In the reproduction phase each individual produces a number of offsprings proportional to her fitness. Each child speaks one grammar, but the exact learning model can vary and allows for the child to incorrectly learn the grammar of her parent. We define the matrix Q where the entry Q_{ij} denotes the probability that the child of an individual using grammar i learns grammar j (i.e. Q is column stochastic matrix); once a child learns a grammar it is fixed and she does not later use a different grammar. Thus, the frequency x'_i of the individuals that use grammar G_i in the next generation will be

$$x'_i = g_i(\mathbf{x}) \coloneqq \sum_{j=1}^{m} \frac{Q_{ji} x_j (B\mathbf{x})_j}{\mathbf{x}^\top B \mathbf{x}}$$

(with $g : \Delta_m \mapsto \Delta_m$ encoding the update rule). Nowak et al. [19] study the symmetric case, i.e., $B_{ij} = b$ and $Q_{ij} = \tau \in (0, 1/m]$ for all $i \neq j$ and observe a threshold: When τ, which can be thought of as quantifying the *error of learning or mutation*, is above a critical value, the only stable fixed point is the uniform distribution (all $1/m$) and below it, there are multiple stable fixed points.

Finite population models can be derived from the linguistic dynamics in a standard way. We describe the Wright-Fisher finite population model for the linguistic dynamics. The population size remains N at all times and the generations are non-overlapping. The current state of the population is described by the frequency vector $\mathbf{X}^{(t)}$ at time t which is a random vector in Δ_m and notice also that the population that uses G_i is $NX_i^{(t)}$. How does one generate $\mathbf{X}^{(t+1)}$? To do this, in the replication (R) stage, one first replaces the individuals that speak grammar G_i in the current population by $NX_i^{(t)}(B(N\mathbf{X}^{(t)}))_i$ and the total population has size $N^2 \mathbf{X}^{(t)\top} B \mathbf{X}^{(t)}$.[6] In the selection (S) stage, one selects N individuals from this population by sampling independently with replacement. Since the evolution is error prone, in the mutation (M) stage, the grammar of each individual in this intermediate population is mutated independently at random according to the matrix Q to obtain frequency vector $\mathbf{X}^{(t+1)}$. Given these rules, note that

$$\mathbb{E}[\mathbf{X}^{(t+1)} | \mathbf{X}^{(t)}] = g(\mathbf{X}^{(t)}).$$

In other words, in expectation, fixing $\mathbf{X}^{(t)}$, the next generation's frequency vector $\mathbf{X}^{(t+1)}$ is exactly $g(\mathbf{X}^{(t)})$, where g is the linguistic dynamics. Of course, this holds only for one step of the process. This process is a Markov chain with state space $\{(y_1, \ldots, y_m) : y_i \in \mathbb{N}, \sum_i y_i = N\}$ of size $\binom{N+m-1}{m-1}$. If $Q > 0$ then it is ergodic (i.e., it is irreducible and aperiodic) and thus has a unique stationary distribution. In our analysis, we consider the symmetric case as in Nowak et al. [19], i.e., $B_{ij} = b$ and $Q_{ij} = \tau \in (0, 1/m]$ for all $i \neq j$.

Note that the linguistics model described above can also be seen as a (finite population) sexual evolution model: Assume there are N individuals and m types. Let $\mathbf{Y}^{(t)}$ be a vector of frequencies at time t, where $\mathbf{Y}_i^{(t)}$ denotes the fraction of individuals of type i. Let F be a fitness matrix where F_{ij} corresponds to the number of offspring of type i, if an individual of

[6] Here we assume that B_{ij} is an positive integer and thus $N^2 X_i^{(t)} (B\mathbf{X}^{(t)})_i$ is an integer since the individuals are whole entities; this can be achieved by scaling and is without loss of generality.

type i chooses to mate with an individual of type j (assume $F_{ij} \in \mathbb{N}$). At every generation, each individual mates with every other individual. It is not hard to show that the number of offspring after the matings will be $N^2(\mathbf{Y}^{(t)\top} F \mathbf{Y}^{(t)})$ and there will be $N^2 \mathbf{Y}_i^{(t)} (F\mathbf{Y}^{(t)})_i$ individuals of type i. After the reproduction step, we select N individuals at random with replacement, i.e., we sample an individual of type i with probability $\frac{\mathbf{Y}_i^{(t)}(F\mathbf{Y}^{(t)})_i}{\mathbf{Y}^{(t)\top} F \mathbf{Y}^{(t)}}$. Finally in the mutation step, every individual of type i mutates with probability τ (*mutation parameter*) to some type j. Let $F_{ii} = A$, $F_{ij} = B$ for all $i \neq j$ with $A > B$ (this is called homozygote advantage) and set $b = \frac{B}{A} < 1$. It is self-evident that this sexual evolution model is identical with the (finite population) linguistic model described above since both end up having the same reproduction, selection and mutation rule. It holds that $E[\mathbf{X}^{(t+1)} | \mathbf{X}^{(t)}] = g(\mathbf{X}^t)$[7] with

$$g_i(\mathbf{x}) = (1-(m-1)\tau)\frac{N^2 x_i (B\mathbf{x})_i}{N^2(\mathbf{x}^\top B\mathbf{x})} + \sum_{j\neq i} \tau \frac{N^2 x_j (B\mathbf{x})_j}{N^2(\mathbf{x}^T B\mathbf{x})} = (1-m\tau)\frac{x_i(B\mathbf{x})_i}{(\mathbf{x}^\top B\mathbf{x})} + \tau$$

where $B_{ii} = 1, B_{ij} = b$ with $i \neq j$.[8] For the Markov chains described above (symmetric case) we can prove the following phase transition result.

▶ **Theorem 9.** *There is a critical value τ_c of the* error in learning/mutation *parameter τ such that the mixing time is: (i) $\exp(\Omega(N))$ for $0 < \tau < \tau_c$ and (ii) $O(\log N)$ for $\tau > \tau_c$ where N is the size of the population.*

The theorem below will be used to prove the rapid mixing result for the finite linguistic model when $\tau > \tau_c$. It is used to construct a potential function and show that the deterministic dynamics g converges to fixed points.

▶ **Theorem 10 (Baum and Eagon Inequality [1]).** *Let $P(\mathbf{x}) = P(\{x_{ij}\})$ be a polynomial with nonnegative coefficients homogeneous of degree d in its variables $\{x_{ij}\}$. Let $\mathbf{x} = \{x_{ij}\}$ be any point of the domain $D : x_{ij} \geq 0, \sum_{j=1}^{q_i} x_{ij} = 1, i = 1, \ldots, p, j = 1, \ldots, q_i$. For $\mathbf{x} = \{x_{ij}\} \in D$, let $\Xi(\mathbf{x}) = \Xi\{x_{ij}\}$ denote the point of D whose i,j-th coordinate is*

$$\Xi(\mathbf{x})_{ij} = \left(x_{ij} \frac{\partial P}{\partial x_{ij}}\bigg|_{(\mathbf{x})} \right) \cdot \left(\sum_{j=1}^{q_i} x_{ij} \frac{\partial P}{\partial x_{ij}}\bigg|_{(\mathbf{x})} \right)^{-1}.$$

Then $P(\Xi(\mathbf{x})) > P(\mathbf{x})$ unless $\Xi(\mathbf{x}) = \mathbf{x}$.

3 Preliminaries

Couplings and Mixing Times

Let $\mathbf{p}, \mathbf{q} \in \Delta_m$ be two probability distributions on m objects. A *coupling* $\mathcal{C}$ of $\mathbf{p}$ and $\mathbf{q}$ is a distribution on ordered pairs in $[m] \times [m]$, such that its marginal distribution on the first coordinate is equal to $\mathbf{p}$ and that on the second coordinate is equal to $\mathbf{q}$. Couplings allow a very useful dual characterization of the total variation distance, as stated in the following well known lemma.

▶ **Lemma 11 (Coupling lemma [14]).** *Let $\mathbf{p}, \mathbf{q} \in \Delta_m$ be two probability distributions on m objects. Then,*

$$\|\mathbf{p} - \mathbf{q}\|_{\mathrm{TV}} = \frac{1}{2}\|\mathbf{p} - \mathbf{q}\|_1 = \min_{\mathcal{C}} \mathbb{P}_{(A,B)\sim\mathcal{C}}[A \neq B],$$

where the minimum is taken over all valid couplings $\mathcal{C}$ of $\mathbf{p}$ and $\mathbf{q}$.

---

[7] We use same notation for the update rule as before, i.e. g because it turns out to be the same function.
[8] Observe that this rule is invariant under scaling of fitness matrix B.

▶ **Definition 12** (Mixing time [14]). Let $\mathcal{M}$ be an ergodic Markov chain on a finite state space Ω with stationary distribution π. Then, the *mixing time* $t_{\text{mix}}(\varepsilon)$ is defined as the smallest time such that for any starting state $\mathbf{X}^{(0)}$, the distribution of the state $\mathbf{X}^{(t)}$ at time t is within total variation distance ε of π. The term mixing time is also used for $t_{\text{mix}}(\varepsilon)$ for a *fixed* values of $\varepsilon < 1/2$.

A well-known technique for obtaining upper bounds on mixing times is to use the Coupling Lemma above. Suppose $\mathbf{X}^{(t)}$ and $\mathbf{Y}^{(t)}$ are two evolutions of an ergodic chain $\mathcal{M}$ such that their evolutions are coupled according to some coupling $\mathcal{C}$. Let T be the smallest time such that $\mathbf{X}^{(T)} = \mathbf{Y}^{(T)}$. If it can be shown that $\mathbb{P}[T > t] \leq 1/4$ for every pair of starting states $(\mathbf{X}^{(0)}, \mathbf{Y}^{(0)})$, then it follows that $t_{\text{mix}} \coloneqq t_{\text{mix}}(1/4) \leq t$.

Operators, Norms

The following theorem, stated here only in the special case of the $1 \to 1$ norm, relates the spectral radius with other matrix norms.

▶ **Theorem 13** (Gelfand's formula, specialized to the $1 \to 1$ norm [13]). *For any square matrix A, we have*

$$\operatorname{sp}(A) = \lim_{\ell \to \infty} \left\|A^{\ell}\right\|_{1\to 1}^{1/\ell}.$$

Taylor Theorem (First order Remainder)

▶ **Theorem 14.** *Let $f : \mathbb{R}^m \to \mathbb{R}$ be differentiable and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$. Then there exists some ξ in the line segment from $\mathbf{x}$ to $\mathbf{y}$ such that $f(\mathbf{y}) = f(\mathbf{x}) + \nabla f(\xi)(\mathbf{y} - \mathbf{x})$.*

Concentration

We also mention some standard Chernoff-Hoeffding type bounds that will be used in our later arguments.

▶ **Theorem 15** (Chernoff-Hoeffding bounds [6]). *Let $Z_1, Z_2, \ldots, Z_N$ be i.i.d. Bernoulli random variables with mean μ. We then have for all $\varepsilon > 0$,*

$$\mathbb{P}\left[\left|\frac{1}{N}\sum_{i=1}^{N} Z_i - \mu\right| > \varepsilon\right] \leq 2\exp\left(-2N\varepsilon^2\right).$$

4 Overview of proofs

We begin by explaining the proof technique of Theorem 5 in [20]. In order to prove a bound on the mixing time, the authors constructed a coupling that contracts the distance between two chains. This contraction does not happen at every step, rather at every k steps where k is some constant and depends on the function f. Essentially, it is shown that given two chains $\mathbf{X}^{(t)}, \mathbf{Y}^{(t)}$ that are close to the unique fixed point $\mathbf{z}$ of f, it holds that

$$\left\|\mathbf{X}^{(t+1)} - \mathbf{Y}^{(t+1)}\right\|_1 \approx \left\|J[\mathbf{z}](\mathbf{X}^{(t)} - \mathbf{Y}^{(t)})\right\|_1 \leq \|J[\mathbf{z}]\|_1 \left\|(\mathbf{X}^{(t)} - \mathbf{Y}^{(t)})\right\|_1$$

with high probability due to Chernoff bounds. Thus, the ℓ_1 norm of the Jacobian captures the contraction if it indeed exists. However it might be the case that $\|J[\mathbf{z}]\|_1 > 1$. On the positive side, using Gelfand's Theorem they were able to show a k-step contraction, since

$$\left\|J^k[\mathbf{z}]\right\|_1 \approx \left(\operatorname{sp}(J[\mathbf{z}])\right)^k < \rho^k < 1$$

for some $k \in \mathbb{N}$. Our proofs also use the idea of Gelfand's formula to show contraction/expansion (in Theorems 7 and 6 respectively) and also make use of Theorem 5). Nevertheless, there are important technical barriers that need to be crossed in order to prove our results as explained below.

4.1 Overview of Theorem 6

The main difficulty to prove this theorem is the existence of multiple unstable fixed points in the simplex from which the Markov chain should get away fast. As before, we study the time T required for two stochastic evolutions with arbitrary initial states $\mathbf{X}^{(0)}$ and $\mathbf{Y}^{(0)}$, guided by some function f, to collide. By the conditions of Theorem 6, function f has a unique stable fixed point $\mathbf{z}_0$ with

$$\mathrm{sp}\left(J[\mathbf{z}_0]\right) < \rho < 1.$$

Additionally, it has α-unstable fixed points. Moreover, for all starting points $\mathbf{x}_0 \in \Delta_m$, the sequence $(f^t(\mathbf{x}_0))_{t\in\mathbb{N}}$ has a limit. We can show that there exists constant c_0 such that $\mathbb{P}\left[T > c_0 \log N\right] \leq \frac{1}{4}$, from which it follows that $t_{\mathrm{mix}}(1/4) \leq c_0 \log N$. In order to show collision after $O(\log N)$ steps, it suffices first to run each chain *independently* for $O(\log N)$ steps. We first show that with probability $\Theta(1)$, each chain will reach $B(\mathbf{z}_0, \frac{1}{N^{1-\varepsilon}})$ after at most $O(\log N)$ steps, for some $\varepsilon > 0$.[9] As long as this is true, the coupling constructed in [20] can be used to show collision (see Section 3 for the definition of a coupling). To explain why our claim holds, we break the proof into three parts.

(a) First, it is shown that as long as the state of the Markov chain is within $o\left(\frac{\log^{2/3} N}{\sqrt{N}}\right)$ in ℓ_1 distance from some α-unstable fixed point $\mathbf{w}$, then, with probability $\Theta(1)$, it reaches distance $\Omega\left(\frac{\log^{2/3} N}{\sqrt{N}}\right)$ after $O(\log N)$ steps. Step (a) has the technical difficulty that as long as a chain starts from a $o(\frac{1}{\sqrt{N}})$ distance from an unstable fixed point, the variance of the process dominates the expansion due to the fact the fixed point is unstable.

(b) Assuming (a), we show that with probability $1 - \frac{1}{\mathrm{poly}(N)}$ the Markov chain reaches distance $\Theta(1)$ from any unstable fixed point after $O(\log N)$ steps.

(c) Finally, if the Markov chain has $\Theta(1)$ distance from any unstable fixed point (the fixed points have pairwise ℓ_1 distance independent of N, i.e., they are "well separated"), it will reach some $\frac{1}{N^{1-\varepsilon}}$-neighborhood of the stable fixed point $\mathbf{z}_0$ exponentially fast (i.e., after $O(\log N)$ steps). For showing (a) and (b), we must prove an expansion argument for $\|f^t(\mathbf{x}) - \mathbf{w}\|_1$ as t increases, where $\mathbf{w}$ is an α-unstable fixed point and also taking care of the random perturbations due to the stochastic evolution. Ideally what we want (but is not true) is the following to hold:

$$\left\|f^{t+1}(\mathbf{x}) - \mathbf{w}\right\|_1 \geq \alpha \left\|f^t(\mathbf{x}) - \mathbf{w}\right\|_1,$$

i.e., one step expansion. The first important fact is that f^{-1} is well-defined in a small neighborhood of $\mathbf{w}$ due to the Inverse Function Theorem, and it also holds that

$$\left\|f^t(\mathbf{x}) - \mathbf{w}\right\|_1 \approx \left\|J^{-1}[\mathbf{w}](f^{t+1}(\mathbf{x}) - \mathbf{w})\right\|_1 \leq \left\|J^{-1}[\mathbf{w}]\right\|_1 \left\|f^{t+1}(\mathbf{x}) - \mathbf{w}\right\|_1,$$

where $\mathbf{x}$ is in some neighborhood of $\mathbf{w}$ and $J^{-1}[\mathbf{w}]$ is the *pseudoinverse* of $J[\mathbf{w}]$ (see the remark in Section 2). However even if $\mathbf{w}$ is α-unstable and $\mathrm{sp}\left(J^{-1}[\mathbf{w}]\right) < \frac{1}{\alpha}$, it can hold that

[9] $B(\mathbf{x}, r)$ denotes the open ball with center $\mathbf{x}$ and radius r in ℓ_1, which we call an *r-neighborhood* of $\mathbf{x}$.

$\left\|J^{-1}[\mathbf{w}]\right\|_1 > 1$. At this point, we use Gelfand's formula (Theorem 13) as in the proof of [20]. Since $\lim_{t\to\infty}(\|A^t\|_1)^{1/t} \to \mathrm{sp}\,(A)$, for all $\varepsilon > 0$, there exists a k_0 such that for all $k \geq k_0$ we have

$$\left|\left\|A^k\right\|_1 - (\mathrm{sp}\,(A))^k\right| < \varepsilon.$$

We use this important theorem to show that for small $\varepsilon > 0$, there exists a k such that

$$\left\|f^t(\mathbf{x}) - \mathbf{w}\right\|_1 \approx \left\|(J^{-1}[\mathbf{w}])^k(f^{t+k}(\mathbf{x}) - \mathbf{w})\right\|_1 \leq \frac{1}{\alpha^k}\left\|f^{t+k}(\mathbf{x}) - \mathbf{w}\right\|_1,$$

where we used the fact that

$$\left\|(J^{-1}[\mathbf{w}])^k\right\|_1 < (\mathrm{sp}\left(J^{-1}[\mathbf{w}]\right))^k - \varepsilon \leq \frac{1}{\alpha^k}.$$

By taking advantage of the continuity of the $J^{-1}[\mathbf{x}]$ around the unstable fixed point $\mathbf{w}$, we can show expansion for every k steps of the dynamical system. It remains to show for (a) and (b) how one can handle the perturbations due to the randomness of the stochastic evolution. In particular, if $\left\|\mathbf{X}^{(0)} - \mathbf{w}\right\|_1$ is $o\left(\frac{1}{\sqrt{N}}\right)$, even with the expansion we have from the deterministic dynamics (as discussed above), variance dominates. We examine case (b) first, which is relatively easy (the *drift* dominates at this step). Due to Chernoff bounds, the difference $\left\|\mathbf{X}^{(t+k)} - \mathbf{w}\right\|_1 - \left\|f^k(\mathbf{X}^{(t)}) - \mathbf{w}\right\|_1$ is $O\left(\sqrt{\frac{\log N}{N}}\right)$ (this captures the deviation on running the stochastic evolution for k steps vs running the deterministic dynamics for k steps, both starting from $\mathbf{X}^{(t)}$) with probability $1 - \frac{1}{\mathrm{poly}(N)}$. Since $\left\|\mathbf{X}^{(t)} - \mathbf{w}\right\|_1$ is $\Omega\left(\frac{\log^{2/3} N}{\sqrt{N}}\right)$, then

$$\left\|\mathbf{X}^{(t+k)} - \mathbf{w}\right\|_1 \geq (\alpha^k - o_N(1))\left\|\mathbf{X}^{(t)} - \mathbf{w}\right\|_1.$$

For (a), first we show that with probability $\Theta(1)$, after one step the Markov chain has distance $\Omega(\frac{1}{\sqrt{N}})$ of $\mathbf{w}$. This claim just uses properties of the multinomial distribution. After reaching distance $\Omega\left(\frac{1}{\sqrt{N}}\right)$, we can use again the idea of expansion and being careful with the variance and we can show expansion with probability at least $\frac{1}{2}$, every k steps. Then we can show that with probability at least $\frac{1}{\log^{2/3} N}$, distance $\frac{\log^{2/3} N}{\sqrt{N}}$ is reached after $O(\log\log N)$ steps and basically we finish with (b). For (c), we use a couple of technical lemmas from [20], we explain in words below: Let Δ be some compact subset of Δ_m, where we have excluded all the α-unstable fixed points along with some open ball around each unstable fixed point of constant radius. We can show that given that the initial state of the Markov chain belongs to Δ, it reaches a $B(\mathbf{z}_0, \frac{1}{N^{1-\varepsilon}})$ for some $\varepsilon > 0$ as long as the dynamical system converges for all starting points in Δ (and it should converge to the stable fixed point $\mathbf{z}_0$). We have roughly that the dynamical system converges exponentially fast for every starting point in B to the stable fixed point $\mathbf{z}_0$ and that with probability $1 - \frac{1}{\mathrm{poly}(n)}$ two arbitrary chains independently will reach a $\frac{1}{N^{\varepsilon}}$ neighborhood of the stable fixed point $\mathbf{z}_0$. Therefore by $(a), (b), (c)$ and the coupling from [20], we conclude the proof of Theorem 6.

4.2 Overview of Theorems 7 and 8

To prove Theorem 8, we make use of Theorem 7, i.e., we reduce the case of the stable limit cycle to the case of multiple stable fixed points. If s is the length of the limit cycle, roughly the bound $e^{\Omega(N)}$ on the mixing time loses a factor $\frac{1}{s}$ compared to the case of multiple stable

fixed points. We now present the ideas behind the proof of Theorem 7. First as explained above, we can show contraction after k steps (for some constant k) for the deterministic dynamics around a stable fixed point $\mathbf{z}$ with $\text{sp}(\text{J}[\mathbf{z}]) < \rho < 1$, i.e.,

$$\left\|f^{t+k}(\mathbf{x}) - \mathbf{z}\right\|_1 \approx \left\|J^k[\mathbf{z}]\right\|_1 \left\|f^t(\mathbf{x}) - \mathbf{z}\right\|_1 \leq \rho^k \left\|f^t(\mathbf{x}) - \mathbf{z}\right\|_1 .$$

To do that, we use Gelfand's formula, Taylor's theorem and continuity of $J[\mathbf{x}]$ where $\mathbf{x}$ lies in a neighborhood of the fixed point $\mathbf{z}$. Hence, due to the above contraction of the ℓ_1 norm and the concentration of Chernoff bounds, it takes a long time for the chain $\mathbf{X}^{(t)}$ to get out of the region of attraction of the fixed point $\mathbf{z}$. Technically, the error that aggregates due to the randomness of the stochastic evolution guided by f does not become large due to the convergence of the series $\sum_{i=0}^{\infty} \rho^i$. Hence, we focus on the error probability, namely the probability the stochastic evolution guided by f deviates a lot from the dynamical system with rule f if both have same starting point after one step. Since this probability is exponentially small, i.e., it holds that

$$\left\|f(\mathbf{X}^{(0)}) - \mathbf{X}^{(1)}\right\|_1 > \varepsilon m$$

with probability at most $2me^{-2\varepsilon^2 N}$, an exponential number of steps is required for the above to be violated. Finally, as we have shown that it takes exponential time to get out of the region of attraction of a stable fixed point $\mathbf{z}$ we do the following easy (common) trick. Since the function has at least two fixed points, we start the Markov chain very close to the fixed point that its neighborhood has mass at most $1/2$ in the stationary distribution (this can happen since we have at least 2 fixed points that are well separated). Then, after exponential number of steps, it will follow that the total variation distance between the distribution of the chain and the stationary will be at least $1/4$.

4.3 Overview of Theorem 9

Below we give the necessary ingredients of the proof of Theorem 9. Our previous results, along with some analysis on the fixed points of g (function of Linguistic Dynamics) suffice to show the phase transition result. To prove Theorem 9, initially we show that the model (finite population) is essentially a stochastic evolution (see Definition 3) guided by g as defined in Section 2.1 and proceed as follows: We prove that in the interval $0 < \tau < \tau_c$, the function g has multiple fixed points whose Jacobian have spectral radius less than 1. Therefore due to Theorem 7 discussed above, the mixing time will be exponential in N. For $\tau = \tau_c$ a bifurcation takes place which results in function g of linguistic dynamics having only one fixed point inside simplex (specifically, the uniform point $(1/m, \ldots, 1/m)$). In dynamical systems, a local bifurcation occurs when a parameter (in particular the mutation parameter τ) change causes two (or more) fixed points to collide or the stability of an equilibrium (or fixed point) to change. To prove fast mixing in the case $\tau_c < \tau \leq 1/m$, we make use of the result in [20] (see Theorem 5). One of the assumptions is that the dynamical system with g as update rule needs to converge to the unique fixed point for all initial points in simplex. To prove convergence to the unique fixed point, we define a Lyapunov function P such that

$$P(g(\mathbf{x})) > P(\mathbf{x}) \text{ unless } \mathbf{x} \text{ is a fixed point.} \tag{1}$$

As a consequence, the (infinite population) linguistic dynamics converge to the unique fixed point $(1/m, \ldots, 1/m)$. To show Equation (1), we use an inequality that dates back in 1967 (see Theorem 10, [1]), which intuitively states the discrete analogue of proving that for a gradient system $\frac{d\mathbf{x}}{dt} = \nabla V(\mathbf{x})$ it is true that $\frac{dV}{dt} \geq 0$.

References

1 L. Baum and J. Eagon. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73:360–363, 1967.

2 Michel Benaïm. Dynamics of stochastic approximation algorithms. In *Seminaire de probabilites XXXIII*, pages 1–68. Springer, 1999.

3 Erick Chastain, Adi Livnat, Christos Papadimitriou, and Umesh Vazirani. Algorithms, games, and evolution. *Proceedings of the National Academy of Sciences*, 2014.

4 Noam A. Chomsky. Rules and Representations. *Behavioral and Brain Sciences*, 3(127):1–61, 1980.

5 Narendra Dixit, Piyush Srivastava, and Nisheeth K. Vishnoi. A finite population model of molecular evolution: Theory and computation. *Journal of Computational Biology*, 19(10):1176–1202, 2012.

6 Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009.

7 Richard Durrett. *Probability models for DNA sequence evolution*. Springer, 2008.

8 Stewart N Ethier and Thomas G Kurtz. *Markov processes: characterization and convergence*, volume 282. John Wiley & Sons, 2009.

9 Warren J. Ewens. *Mathematical Population Genetics I. Theoretical Introduction*. Springer, 2004.

10 W.T. Fitch. *The Evolution of Language*. Approaches to the Evolution of Language. Cambridge University Press, 2010.

11 J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

12 Natalia L. Komarova and Martin A. Nowak. Language dynamics in finite populations. *Journal of Theoretical Biology*, 221(3):445–457, 2003.

13 Erwin Kreyszig. *Introductory Functional Analysis with Applications*. Wiley, 1978.

14 David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2008.

15 Yun Long, Asaf Nachmias, and Yuval Peres. Mixing time power laws at criticality. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, FOCS'07, pages 205–214, Washington, DC, USA, 2007. IEEE Computer Society.

16 J. Maynard-Smith and E. Szathmary. *The Major Transitions in Evolution*. New York: Oxford University Press, 1997.

17 Ruta Mehta, Ioannis Panageas, and Georgios Piliouras. Natural selection as an inhibitor of genetic diversity: Multiplicative weights updates algorithm and a conjecture of haploid genetics. In *Innovations in Theoretical Computer Science*, 2015.

18 M.A. Nowak. *Evolutionary Dynamics*. Harvard University Press, 2006.

19 Martin A. Nowak, Natalia L. Komarova, and Partha Niyogi. Evolution of universal grammar. *Science*, 2001.

20 Ioannis Panageas, Piyush Srivastava, and Nisheeth K. Vishnoi. Evolutionary dynamics in finite populations mix rapidly. *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, 2016.

21 Christos H. Papadimitriou and Nisheeth K. Vishnoi. On the computational complexity of limit cycles in dynamical systems. In *Proceedings of the 2016 Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, page 403, 2016.

22 Robin Pemantle. When are touchpoints limits for generalized Pólya urns? *Proceedings of the American Mathematical Society*, pages 235–243, 1991.

23 Georgios Piliouras, Carlos Nieto-Granda, Henrik I. Christensen, and Jeff S. Shamma. Persistent patterns: Multi-agent learning beyond equilibrium and utility. In *AAMAS*, pages 181–188, 2014.

24 Kushal Tripathi, Rajesh Balagam, Nisheeth K. Vishnoi, and Narendra M. Dixit. Stochastic simulations suggest that HIV-1 survives close to its error threshold. *PLoS Comput Biol*, 8(9):e1002684, 09 2012.

25 Nisheeth K. Vishnoi. The speed of evolution. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1590–1601, 2015.

26 Nicholas C Wormald. Differential equations for random processes and random graphs. *The annals of applied probability*, pages 1217–1235, 1995.

Information Cascades on Arbitrary Topologies*

Jun Wan†[1], Yu Xia‡[2], Liang Li[3], and Thomas Moscibroda[4]

1 **The Institute for Theoretical Computer Science(ITCS), Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China**
wanj12@mails.tsinghua.edu.cn

2 **The Institute for Theoretical Computer Science(ITCS), Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China**
xiay12@mails.tsinghua.edu.cn

3 **Microsoft Research Asia, Beijing, China**
liangl@microsoft.com

4 **Microsoft Research Asia, Beijing, China**
moscitho@microsoft.com

Abstract

In this paper, we study information cascades on graphs. In this setting, each node in the graph represents a person. One after another, each person has to take a decision based on a private signal as well as the decisions made by earlier neighboring nodes. Such information cascades commonly occur in practice and have been studied in complete graphs where everyone can overhear the decisions of every other player. It is known that information cascades can be fragile and based on very little information, and that they have a high likelihood of being wrong.

Generalizing the problem to arbitrary graphs reveals interesting insights. In particular, we show that in a random graph $G(n, q)$, for the right value of q, the number of nodes making a wrong decision is logarithmic in n. That is, in the limit for large n, the fraction of players that make a wrong decision tends to zero. This is intriguing because it contrasts to the two natural corner cases: empty graph (everyone decides independently based on his private signal) and complete graph (all decisions are heard by all nodes). In both of these cases a constant fraction of nodes make a wrong decision in expectation. Thus, our result shows that while both too little and too much information sharing causes nodes to take wrong decisions, for exactly the right amount of information sharing, asymptotically everyone can be right. We further show that this result in random graphs is asymptotically optimal for any topology, even if nodes follow a globally optimal algorithmic strategy. Based on the analysis of random graphs, we explore how topology impacts global performance and construct an optimal deterministic topology among layer graphs.

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

Keywords and phrases Information Cascades, Herding Effect, Random Graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.64

* The full version of this paper can be found in http://arxiv.org/abs/1604.07166

† This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00301, the National Natural Science Foundation of China Grant 61361136003.

‡ Part of this work was done when this author visited Microsoft Research Asia.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 64; pp. 64:1–64:14

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

1 Introduction

An Information Cascade occurs when a person observes the actions of others and then – in spite of possible contradictions to his/her own private information – follows these same actions. A cascade develops when people "abandon their own information in favor of inferences based on earlier people's actions"[12]. Information Cascades frequently occur in everyday life. Commonly cited examples include the choice of restaurants when being in an unknown place people choose the restaurant that already has many guests over a comparatively empty restaurant, or hiring interview loops where interviewers follow earlier interviewer's decisions if they are not sure about the candidate. Notice that information cascades are not irrational behavior; on the contrary, they occur precisely because people rationally decide based on inferences derived from earlier people's actions.

The simple herding experiment by Anderson and Holt illustrates Information Cascades [3, 4](see also Chapter 16 in [12]). In this experiment, an urn contains three marbles, either two red and one blue (majority red), or one red and two blue (majority blue). The players do not know whether the urn is majority red or blue. One by one, the players privately pick one marble from the urn, check its color, return it to the urn, and then publicly announce their guess as to whether the urn is majority red or majority blue. The first and second player will naturally base their guesses on the colors of the marble they picked, thus their guesses reveals their private signals. For any subsequent player however, her rational guess may not reflect her own signal. For example, suppose the first two players both guess *red*. In this case, it is rational for the third player to also guess *majority-red* regardless of the color of the marble she picked. Indeed, the third player makes her decision on a rational inference based on the first two guesses. Since her guess does therefore not reveal any further information about the urn to any subsequent player, *every* subsequent player will guess the urn to be *majority-red*. The example shows that information cascades can be based on very little actual information and thus fragile; and they can be wrong. Indeed, in the above example with urns, it can be shown that with probability 1/5, a "wrong cascade" occurs, i.e., all players (except from possibly a few at the beginning) will guess wrongly.

The standard model for information cascades studies the process in which players make decisions sequentially based on their own private signals as well as the set of decisions made by earlier players [6, 7, 23]. In this paper, we interpret and generalize the traditional information cascade setting as a game in a graph. Each player is a node, and an edge between two nodes v and w means that w can hear about v's guess (assuming w is after v in the order of decision-making). Thus the traditional information cascade model corresponds to a complete graph (all players hear the decisions of all other players). At the other end of the spectrum, the empty graph means that every player decides independently of all other players, purely based on their own private signal. Casting the information cascade problem in this graph setting allows us to study the range in between the two extreme points of complete and empty graphs.

Studying this range in between reveals fascinating insights. Figure 1 shows the expected number of wrong guesses in the above 3-marble-urn experiment in a random graph $G(n, q)$ topology, for different values of n and q. In the empty graph ($q = 0$), if all nodes take their decisions independently, 1/3 of the players are wrong. In the complete graph ($q = 1$), 1/5 of the players are wrong on average as discussed above. However, the interesting thing is that for some values in between these two extremes, the number of wrong decisions is significantly less. Indeed, it seems that for the right value of q and $n \to \infty$, the number of wrong decisions tends to 0.

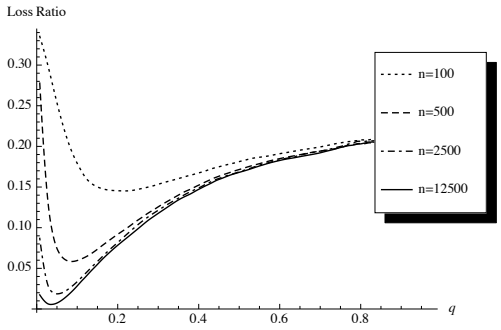


Figure 1 Performance of random graphs for different q and n.

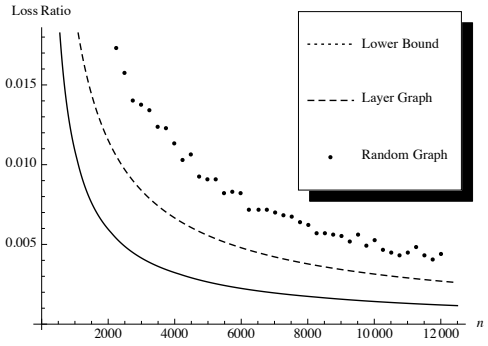


Figure 2 Performance of different topologies and strategies.

These observations are intriguing: It looks like that if people share too much information, a constant fraction of the population is wrong because of bad information cascades occurring. If people share too little information, a larger constant fraction of the population is wrong because the players take their decisions too independently, relying too much on their private signal which has a constant probability of being wrong. But, if exactly the right amount of information is shared, then it seems that in the limit, *all players* (at least asymptotically) take the correct decision.

In this paper, we study this phenomenon. We prove that, indeed, in a random graph the number of wrong nodes is at most $O(\log n)$ for the optimal value of q (Section 3). We then study arbitrary graph topologies and show that $O(\log n)$ wrong nodes is optimal in a strong sense (Section 4). Specifically, even in the best possible topology, there are at least $\Omega(\log n)$ wrong nodes. This result holds even if a global oracle tells each node whether it should (a) base its decision solely on its private signal (thus revealing this signal as additional information to all its neighbors) or (b) base its decision on the majority of private signal and neighboring decisions as in the cascade model above. In other words, even if nodes can "sacrifice" themselves to reveal additional information to their neighbors and even in the best possible topology $\Omega(\log n)$ wrong nodes is a lower bound. Finally, we derive an optimal deterministic topology from among a family of *layer graphs* (Section 5). Detailed proofs and analysis can be found in [22].

2 Related Work

Sequential decision-making has been studied in various areas including politics, economics and computer science[21, 6, 7, 14, 12]. The primary concern on the Bayesian learning model[7, 19, 6, 23, 20, 5, 1] is under what conditions asymptotically correct information cascades occur. For specific graph topologies such as complete graphs and line graphs, conditions on the private signals were addressed to guarantee the correctness of cascades[19, 9]. For arbitrary graph topologies, the approach of Acemoglu et al. [1] is intuitively quite consistent to our k-layer topology(see Section 6) and can be used to explain why our random network and selfishless decision-making algorithm achieve global optimality. While their approach focuses on the asymptotic probability of correct cascades, our result can quantitatively bound the expectation number of incorrect nodes.

There has been research on different sequential decision making models in graphs. For example, Chierichetti et al. [10] study different algorithms for finding appropriate orderings

to maximize the fraction making correct decisions and Hajiaghayi et al. [16] and Hajiaghayi et al. [15] generalizes the model and improve related bounds. However, notice that the threshold decision-making processes studied in these works fundamentally differ from the information cascade setting we consider in this work. Indeed, the effect of too little/too much information sharing being bad as shown in Figure 1 is not observed in such threshold models.

There also exists an impressive body of work on sequential and non-sequential decision on arbitrary graphs that however do not capture information cascades as exemplified in Anderson and Holt's herding experiment. Typically, each node updates its opinion through repeated averaging with neighbors. General conditions for convergence to consensus have been developed[2, 11, 13]. Intrigued by the observation that consensus is usually not reached in real world[18], Bindal et al. [8] use a game theoretic approach to study the equilibrium of the dynamical process and measure the cost of disagreement via the Price of Anarchy[17].

3 Preliminaries

We introduce the formal definitions of our model. There are n nodes (numbered $1, 2, \cdots n$) whose neighboring relationship is depicted by a graph $G = ([n], E)$. All nodes make decisions sequentially according to their numbers in order to guess a global ground truth value $b \in \{0, 1\}$. When making its decision, each node can only obtain a random partial information on b. That is, when node i observes b, it can only get a *private signal* s_i which equals b with probability $p > 0.5$ or equals $1 - b$ with probability $1 - p$. The decision-making of a node not only depends on its private signal observed from b, but also on the decisions made by its previous neighbors. Note that the neighboring decisions may or may not be based on those nodes' private signals. More formally, let c_i be the *output decision* or *guess* of node i and c^i be the *decision vector* $(c_1, c_2, \cdots, c_i)$, if $L_i : \{0, 1\}^{i-1} \times \{0, 1\} \to \{0, 1\}$ is the *decision-making algorithm* for node i, we have in general $c_i = L_i(c^{i-1}, s_i)$.

Given the graph G and decision-making algorithms $L_1, L_2, \cdots, L_n$, we use $\mathcal{E}_G(L_1, \cdots, L_n)$ to denote the expected number of nodes that output the wrong value $1 - b$. When it is clear from the context, we may abbreviate this notation to $\mathcal{E}_G$, $\mathcal{E}(L_1, \cdots, L_n)$, or simply $\mathcal{E}$. The global objective of this sequentially decision-making process is to minimize $\mathcal{E}_G(L_1, \cdots, L_n)$, which is equivalent to maximizing the expected number of nodes who guess the ground truth value correctly. We will show that such an optimization task can be achieved by adjusting the graph topology or the decision-making algorithms.

Let $\mathcal{E}_i$ be the *failure probability* that node i outputs $1 - b$. As a node often makes inferences based on others' decisions without knowing their private signals, it is intuitively understandable that a node's probability of correct decision-making can be quantified by the number of private signals it can infer.

In reality, the Majority Algorithm is one of the most popular and practical algorithms for decision-makings. This kind of "following the herd" algorithm can often achieve a locally optimal effect. In this paper, we use Maj_k to denote the Majority Algorithm taking input bits of length k. We just use Maj if k is clear from the context. In Chapter 16 of [12], Easley and Kleinberg shows that the Majority Algorithm is optimal when a node observes multiple independent signals.

▶ **Claim 1.** *For any node i seeking to maximize $\mathcal{E}_i$, when it observes multiple signals(including its own private signal), its optimal algorithm is to output the majority of these observed signals.*

However, Anderson and Holt's experiment shows that if all nodes apply the Majority Algorithm, it is possible that essentially all of the nodes guess incorrectly, leading to an

information cascade on the wrong side. In this paper, we address this problem and analyze the impact of topology and algorithms on information cascades.

4 Random Graphs

In this section, we analyze the performance of the Majority Algorithm on random graphs. Conventionally, $G(n,q)$ denotes the random graph model that generates a random graph with n nodes and each pair of nodes are connected by an edge with probability q. Different connection probabilities induces completely different topologies, and thus dramatic changes in $\mathcal{E}_{G(n,q)}$. As introduced in Section 1, when q equals 0 or 1 corresponding to the empty or complete topology, the expected number of wrong output decisions are both $\Theta(n)$.

In this section, we show that there exists a q such that the Majority Algorithm can achieve only $\Theta(\log n)$ expected wrong output decisions:

▶ **Theorem 2.** *There exists a connection probability $q = \Theta(1/\log n)$ such that in $G(n,q)$, when all nodes apply the Majority Algorithm, we have $\mathcal{E}_{G(n,q)} = \Theta(\log n)$.*

We can also demonstrate the optimality of this connection probability by showing a lower bound for the expected number of wrong decisions:

▶ **Theorem 3.** *For any connection probability q, when all nodes apply the Majority Algorithm, the expected number of wrong outputs in $G(n,q)$ is lower bounded by $\Theta(\log n)$, i.e. $\mathcal{E}_{G(n,q)} = \Omega(\log n)$.*

A key ingredient to our proof is to bound the failure probability $\mathcal{E}_i$ for each node. Applying the Chernoff Bound and the Union Bound, we can further bound the overall $\mathcal{E}_{G(n,q)}$. The following are two technical lemmas for bounding the failure probabilities:

▶ **Lemma 4.** *If a constant fraction $f > 0.5$ of the first i nodes are correct (resp. wrong), node $i+1$'s failure (resp. correct) probability $\mathcal{E}_{i+1}$ is upper bounded by $e^{-\Theta(iq)}$.*

▶ **Lemma 5.** *If a constant fraction $f > 0.5$ of the first i nodes are correct, s.t. $\frac{f}{1-f} \geq \sqrt{\frac{q}{1-q}}$, then node $i+1$'s failure probability is upper bounded by p.*

4.1 Proof of Theorem 2

In this subsection, we provide a detailed proof of Theorem 2.

The reason why random graphs behave well for the right value of q is that randomness defers the process of information cascades. The fewer neighbors, the more likely a not will output its private signal, thereby 1) having a high probability of being wrong, but 2) revealing important information to its neighbors. When $q = \Theta(1/\log n)$, with high probability each of the first $\Theta(\log n)$ nodes have at most one neighbor. By definition of Majority Algorithm, any node with only one neighbor will be forced to output its own private signal.

Using Lemma 4, we can prove that an established cascade among the first $\log n/q$ nodes decides the outputs of all later nodes with high probability:

▶ **Lemma 6.** *If among the first $\log n/q$ nodes, only a small constant fraction $f < 0.5$ output wrongly, then for the later $n - (\log n/q)$ nodes, the expected number of wrong outputs is at most $O(1)$.*

Lemma 6 is insufficient to bound the $\Theta(\log n)$ expected failure nodes as required by Theorem 2, in that it only bounds the loss of later nodes in the sequence. It could be the

case that the first $\log n/q = \Theta(\log^2 n))$ nodes all fail. To bound the overall $\mathcal{E}$, it is essential to analyze the performance of the first $\log n/q$ nodes. We can use an induction argument to show that for the optimal q, the first $\log n/q$ nodes are majority-correct with high probability:

▶ **Lemma 7.** *Let* $\delta = \frac{1}{2}(p + \frac{\sqrt{p}}{\sqrt{p}+\sqrt{1-p}})$. *There exists connection probability* $q_{opt} = \Theta(1/\log n)$ *such that the first* $\Theta(\log n/q)$ *nodes contains at least* δ *portion of correct outputs with probability* $1 - O(n^{-1}\log n)$.

Proof. We can prove Lemma 7 by induction. Consider dividing the first $\Theta(\log n/q)$ nodes into $\Theta(\log n)$ segments, where each segment contains $\Theta(1/q) = \Theta(\log n)$ many nodes. We analyze each segment independently and show that

- There exists $q_1 = \Theta(1/\log n)$, such that the first segment contains δ portion of correct outputs with probability at least $1 - O(n^{-1})$.
- If the first i segments contain δ portion of correct outputs, then the $(i+1)^{th}$ segment will also contain δ portion of correct outputs with probability $1 - O(n^{-1})$.

By a single Union Bound, we can combine these two results and show that the first $\Theta(\log n/q)$ nodes contain δ portion of correct outputs with probability $1 - \log n \cdot O(n^{-1})$. ◀

Lemma 7 and 4 together imply a $\Theta(\log n)$ upper bound for the expected number of wrong outputs among the first $\Theta(\log n/q)$ nodes:

▶ **Lemma 8.** *If* $q = q_{opt}$, *the first* $\Theta(\log n/q)$ *nodes' expect to have at most* $\Theta(\log n)$ *many wrong outputs.*

Proof of Theorem 2. With Lemma 6, 7 and 8 proved, the expected number of wrong output decisions under connection probability q_{opt} is bounded by

$$\begin{aligned}\mathcal{E}_{G(n,q)} = \sum_{i=1}^{n}\mathcal{E}_i &= \sum_{i=1}^{\log n/q}\mathcal{E}_i + \sum_{i=\log n/q+1}^{n}\mathcal{E}_i \\ &\leq \Theta(\log n) + (1 - O(n^{-1}\log n)) \cdot O(1) + O(n^{-1}\log n) \cdot n = \Theta(\log n).\end{aligned} \tag{1}$$

which completes the proof. ◀

4.2 Proof of Theorem 3

In the previous section, we prove that for the optimal connection probability $q_{opt} = \Theta(1/\log n)$, the expected number of wrong outputs is reduced to $\Theta(\log n)$. However, it remains a problem whether we can move beyond $\Theta(\log n)$. In this section, we prove Theorem 3 which states that the bound in Theorem 2 is asymptotically optimal.

Proof of Theorem 3. We prove this theorem for two separate cases, namely when $q = O(1/\log n)$ and $q = \omega(1/\log n)$.

When $q = O(1/\log n)$, the intuition is that we need at least $\Theta(1/q)$ nodes before accumulating an actual influential cascade. For the i^{th} nodes where $i \leq 1/q$, its chance of being isolated is $(1-q)^i + iq(1-q)^{i-1} \geq (1-q)^{1/q} \sim 1/e$. Therefore the node's failure probability is at least $\mathcal{E}_i = (1-p) \cdot \Pr[\text{isolated}] = (1-p)/e$. This lower bounds the expected number of failure nodes by $(1-p)/(eq) = \Theta(1/q)$.

When $q = \omega(1/\log n)$, a wrong cascade occurs with high probability, thus resulting in a significant number of failure nodes. With probability $(1-p)^{\Theta(1/q)}$, all of the first $\Theta(1/q)$ nodes observe a wrong signal and output the wrong guesses. Using Lemma 4, we can show that with high probability, the majority of later nodes follow this wrong cascade. Therefore, the total number of failure nodes is at least $(1-p)^{\Theta(1/q)} \cdot \Theta(n) = n^{1-o(1)} = \Omega(\log n)$. ◀

5 General Lower Bound

In this section, we design a non-constructive scheme that finds the optimal decision-making algorithms for general graphs. Given the graph $G = ([n], E)$, our goal is to find the set of algorithms $\{L_i\}_{i=1}^n$ such that $(L_1, \cdots, L_n) = \arg\min_{L'_1,\cdots,L'_n} \mathcal{E}_G(L'_1, \cdots, L'_n)$.

An important use of the non-constructive scheme is to provide a general lower bound for arbitrary topology. For any set of decision-making algorithms in a topology G, we can simulate it on a complete graph by considering only edges in G. Thus the minimal $\mathcal{E}$ for complete graphs is a general lower bound for arbitrary topology:

▶ **Theorem 9.** *The expected number of wrong nodes $\mathcal{E}$ under the optimal decision-making algorithms of complete graphs lower bounds the $\mathcal{E}$ of any algorithms in any topology.*

From our previous discussion on random graphs, we know that the expected number of wrong guesses $\mathcal{E}$ highly depends on the number of nodes revealing their private signals. This inspires us to make the following definitions:

▶ **Definition 10.** Node i reveals **valid** information under c^{i-1} if and only if node i outputs its private signal under c^{i-1}, i.e. $c_i = L_i(c^{i-1}, s_i) = s_i$. Furthermore, we denote Valid$(\cdot)$ as a function that extracts a vector of valid information out of a decision vector, i.e. c_j is in the vector Valid(c^i) if and only if c_j is valid.

▶ **Definition 11.** A node i's **reveal set** RS_i is the set of c^{i-1} which causes node i to reveal **valid** information.

Note that any valid information is correct with probability p and is independent of other nodes. Using the same Bayesian argument[12], we can prove a similar lemma as Claim 1 in Section 3, which states that a node's guess is beneficial for later nodes if and only if the guess is valid:

▶ **Lemma 12.** *For a node i seeking to minimize its failure probability $\mathcal{E}_i$, the optimal decision-making algorithm is to perform the Majority Algorithm on Valid$(c^{i-1}) \bigcup \{s_i\}$, i.e. $c_i = Maj($ Valid$(c^{i-1}), s_i)$.*

5.1 A non-constructive optimal algorithm scheme for general graphs

In this section, we provide a general scheme for finding the optimal decision-making algorithms of all nodes in arbitrary topologies. Our scheme is non-constructive in that it neither explicitly specifies what the optimal algorithms are, nor shows how to find them efficiently.

Given the underlying topology, all the nodes decide their algorithms sequentially in a *greedy* way as follows. Node 1 publicly announces L_1, based on its own rationality, then node 2 announces L_2 with the knowledge of L_1, etc(see Algorithm 1). Any node i will base its knowledge on $L_1, \cdots, L_{i-1}$ when deciding L_i. Each node designs its own decision-making algorithm in order to locally minimizes the failure probability. Denote this construction scheme as $GC(\cdot)$, the abbreviation of "greedy construction", then for node i, we have $L_i = GC(L_1, \cdots, L_{i-1})$. We can prove by contradiction that such a locally optimal scheme lead to an overall optimality:

▶ **Theorem 13.** *$L_1, L_2, \ldots, L_n$ constructed as in Algorithm 1 minimizes the expected number of wrong nodes, i.e. $\mathcal{E}(L_1, L_2, \ldots, L_n)$.*

Algorithm 1 A non-constructive optimal algorithm scheme for general graphs

1: Given $L_1, \cdots, L_{n-1}$, node n constructs L_n that aims at minimizing its own failure probability $\mathcal{E}_n$.
2: Given $L_1, \cdots, L_{n-2}$, and also the fact that node n is greedy, node $n-1$ constructs L_{n-1} such that the overall loss of him and node n is minimized.
3: This process continues. Each L_i greedily minimizes the expected number of wrong nodes after among $\{i, \cdots, n\}$ given $L_1, \cdots, L_{i-1}$.
4: Node 1 knows that all later nodes are "greedy". Their algorithms $L_2, \cdots, L_n$ can all be written as a function of L_1. It then constructs L_1 such that $\mathcal{E}$ is minimized.
5: Knowing what L_1 is, we can backtrack L_2, and recursively all the output algorithms L_i.

5.2 Optimal algorithms for complete graphs

In this section, we specify the optimal decision-making algorithms for complete graphs and thus provide a general lower bound for our model(by Theorem 17). Several intrinsic properties regarding information cascades in complete graph will also be presented.

We start with a lemma showing that optimal algorithm will either reveal **valid** information or perform Majority Algorithm on all previous guesses.

▶ **Lemma 14.** *In the optimal algorithm, a node either reveals **valid** information or apply Majority Algorithm on all previous outputs, i.e.*

$$L_i = \begin{cases} s_i & c^{i-1} \in RS_i \\ Maj(c^{i-1}) & c^{i-1} \notin RS_i \end{cases}.$$

It is worth pointing out several non-trivial points of Lemma 14 : (a) the Majority Algorithm is performed on previous guesses only and ignores its own private signal; (b) the Majority Algorithm is performed on *all* previous guesses, not only on the valid guesses. An established result in the proof of Lemma 14 is that the Majority Algorithm will cascade on complete graphs, i.e. if a node performs Majority Algorithm, all later nodes will also perform Majority Algorithm. This implies the existence of a switching point, where all nodes prior to this point reveal their private signals, and all later nodes perform Majority Algorithm based on former nodes' signals. If we can estimate the position of this switching point, then an estimation of $\mathcal{E}$ is achieved.

Lemma 14 specifies a node's action outside the reveal set. However, to get an explicit representation of L_i, an understanding of the reveal set itself is required. We introduce the following lemma that fills this gap.

Denote $\text{diff}(c^{i-1}) = (\#\ 1 \text{ in Valid}(c^{i-1})) - (\#\ 0 \text{ in Valid}(c^{i-1}))$, which serves as a criteria to measure the strength of valid information in previous decision vector $c^{i-1} = (c_1, \cdots, c_{i-1})$.

▶ **Lemma 15.** *The reveal set of a node i can be explicitly expressed with respect to some parameters $\delta_n(\cdot)$, where $RS_i = \{c^{i-1} : |diff(c^{i-1})| \geq \delta_n(i)\}$.*

Proof. This lemma follows from the fact that a node outputs based on the Bayesian probability for the ground truth bit b, which depends solely upon $\text{diff}(c^{i-1})$. Given c^{i-1}, the Bayesian probability for b is

$$\begin{cases} \Pr[b=0|c^{i-1}] = \frac{1}{1+(\frac{p}{1-p})^{\text{diff}(c^{i-1})}} \\ \Pr[b=1|c^{i-1}] = \frac{1}{1+(\frac{1-p}{p})^{\text{diff}(c^{i-1})}} \end{cases}. \tag{2}$$

Lemma 14 implies that for each node, (a) if the previous decision vector convinces it that b equals $\text{Maj}(c^{i-1})$ with high probability, it follows the majority of former output decisions; (b) otherwise, it tries to provide more information by revealing its own private signal. As implied by Equation (2), the larger $|\text{diff}(c^{i-1})|$ is, the more likely $b = \text{Maj}(\text{Valid}(c^{i-1}))$. This lead us to conclude the existence of a threshold $\delta_n(i)$ such that L_i applies Majority Algorithm if and only if $|\text{diff}(c^{i-1})| \geq \delta_n(i)$. ◀

Finally, given i and n as input, we show how to efficiently derive $\delta_n(i)$ in average $O(\log n)$ time. Denote $\mathcal{E}(i,d)$ to be the expected number of wrong nodes given that $|\text{diff}(c^{i-1})| = d$. The idea is to use recursion to derive $\mathcal{E}(i,d)$ for all i and d, in the process of which $\{\delta_n(i)|i\}$ may be calculated. If a node k chooses to reveal its private signal, $\mathcal{E}(k,d)$ is updated as

$$\mathcal{E}(k,d) = q_1 \cdot \mathcal{E}(k+1, d+1) + (1-q_1) \cdot \mathcal{E}(k+1, d-1),$$

where q_1 is the probability that node k's private signal matches the majority of former guesses. Similarly, if node k chooses to do Majority Algorithm, $\mathcal{E}(k,d)$ is updated as

$$\mathcal{E}(k,d) = q_2 \cdot \left(n - \frac{k+d}{2}\right) + (1-q_2) \cdot \frac{k+d}{2},$$

where q_2 is the probability that the majority of former outputs is correct. Therefore, we can calculate $\{\mathcal{E}(i,d)\}$ in time $O(n^2)$. A further improvement can be made by exploiting the properties of $\delta_n()$: $\delta_n(i+1) - 1 \leq \delta_n(i) \leq \delta_n(i+1) + 1$, Thus to calculate $\{\delta_n(i)\}$, it suffices to calculate $\{\mathcal{E}(i,d) \mid d < \delta_n(i), i \geq n\}$, which requires only $n \cdot \max_i\{\delta_n(i)\} = O(n \log n)$ time complexity.

▶ **Lemma 16.** *Given n as input, we can calculate the set $\{\delta_n(i)\}$ in $O(n \log n)$ time.*

5.3 General lower bound for our model

Finally we analyze the expected number of wrong nodes for the optimal algorithms in complete graphs, and provide a general $\Theta(\log n)$ lower bound for the model.

▶ **Theorem 17.** *The expected number of wrong nodes $\mathcal{E}$ for any topology and any algorithm is at least $\Theta(\log n)$.*

Proof. It suffices to prove that the $\mathcal{E}$ of the optimal algorithms in complete graph is bounded by $\Theta(\log n)$. In the proof of Lemma 14, we develop the concept of a "switching point", where all nodes prior to this point reveal valid information and all nodes afterwards perform Majority Algorithm. Denote m as a random variable of the switching point's position. We prove that at least one of the following happens: (a) $m \geq (\log_{p/(1-p)} n)/2$; or (b) $\mathcal{E}$ is greater than $\sqrt{n}/2$.

If $m < (\log_{p/(1-p)} n)/2$, then from Equation (2), we know that the majority of revealed signals are wrong with probability at least $(1 + (p/(1-p))^m)^{-1} > n^{-0.5}/2$. So the expected number of wrong nodes is lower bounded by $(n-m)/\sqrt{n}$ which is asymptotically greater than $\sqrt{n}/2$. Therefore, for the optimal output algorithms in complete graph, $\mathcal{E}$ is at least

$$\min\left((1-p)\log_{p/(1-p)} n \Big/ 2, \sqrt{n}\Big/2\right) = \Omega(\log n).$$ ◀

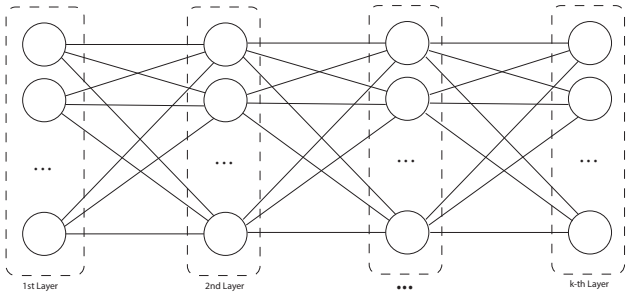


Figure 3 A k-layer graph.

6 Optimal K-Layer Topology

In section 3, we show that the optimal $\mathcal{E}$ for random graphs is $\Theta(\log n)$, which is asymptotically the same as the general lower bound in Theorem 17. Yet the question remains what is the actual optimal topology for the Majority Algorithm.

In this section, we propose a family of layer graphs and search for the optimal topology among this family. We claim, without proof, that the optimal topology of layer graphs is actually the optimal topology for Majority Algorithm.

To find the optimal topology, it helps to first understand the hidden insights behind small overall $\mathcal{E}$. In the optimal algorithms for complete graphs, nodes first judge the strength of the current cascade, and then decide whether to follow the cascade or reveal their own signals to strengthen the cascade. Such a *think-before-acting* way of decision-making guarantees the correctness probability of any established cascade, and thus results in good overall performance. We hope to know whether such *think-before-acting* could make it possible for Majority Algorithm to achieve optimality simply by adjusting the topology. This inspires us the following definition of layer graphs.

▶ **Definition 18** (Definition of layer graphs)**.** A graph is said to have k layers if it can be separated into k disjoint groups, $S_1, \cdots, S_k$, where any node in group S_i is connected to and only to all nodes in S_{i-1}. See Figure 3 for an example.

▶ Algorithm 19. Given a k-layer graph G, we consider how nodes perform in G. First of all, similar to the optimal algorithms in complete graph, we have $|S_1|$ many nodes revealing **valid** information at the very front. If there exists a cascade in S_1 (the number of one choices outmatches another by at least two), then all later nodes follow this cascade. Otherwise, nodes in S_2 reveal their private signals. This process continues until a cascade happens in some layer. In this sense, layer graphs do contains the think-before-acting way of decision-making.

In the following sections, we find the optimal topology among layer graphs, and show that the expected number of wrong nodes $\mathcal{E}$ for such optimal topology is also $\Theta(\log n)$. This optimal $\mathcal{E}$ will be compared to previous bound and results, from which we will be able to glimpse the limit of Majority Algorithm. Throughout this section, if not otherwise mentioned, we will assume that the output algorithm is Majority Algorithm. We denote the expected number of wrong nodes on a k-layer topology $(S_1, \cdots, S_n)$ as $\mathcal{E}(|S_1|, \cdots, |S_n|)$.

6.1 Optimal topology for layer graphs

Remark 19 provides an intuitive way to calculate the $\mathcal{E}$ of any layer graph. Given i independent signals, we denote $p_w(i)$ as the probability that these signals generate a wrong cascade,

and $p_n(i)$ as the probability that these signals does not generate cascade in either side. A recursion regarding $\mathcal{E}$ of any layer graph can be shown to be:

$$\mathcal{E}(a_1, \cdots, a_k) = (1-p) \cdot a_1 + p_w(a_1) \cdot (n - a_1) + p_n(a_1) \cdot \mathcal{E}(a_2, \cdots, a_k), \tag{3}$$

where $(1-p)a_1$ is the expected number of wrong nodes among the first layer, $n - a_1$ is the expected number of wrong nodes among later layers under wrong cascade, and $\mathcal{E}(a_2, \cdots, a_k)$ is the expected number of wrong nodes of later layers under no cascade. By extending the recursive term, we can simplify Equation (3) into

$$\mathcal{E}(a_1, \cdots, a_k) = \sum_{i=1}^{k} \Big(\prod_{j=1}^{i-1} p_n(a_j) \cdot \Big((1-p) \cdot a_i + p_w(a_i) \cdot (n - \sum_{j=1}^{i-1} a_j) \Big) \Big). \tag{4}$$

▶ Algorithm 20. Our goal is to estimate $\mathcal{E}$ of the optimal layer graph to the $\Theta(\log n)$ level. Any approximation of $\mathcal{E} + o(\log n)$ would be satisfying. This relaxation releases us from getting an exact optimum and allows us to make proper adjustments that greatly reduce the difficulty of the calculation. For example, in the derivation of Equation (3) and (4), we assume without loss of generality that each layer has an even number of nodes. This will result in $O(1)$ changes in the optimal parameters, which is tolerable.

To find a set of parameters $\{k, a_1, \cdots, a_k\}$ such that $\mathcal{E}(a_1, \cdots, a_k)$ is minimized, we need the following basic steps:

- We first show that in Equation (3), the contribution of $p_n(a_1)\mathcal{E}(a_2, \cdots, a_k)$ is limited and may be discarded without much change to the optimal parameters. Therefore, it suffices to consider the optimization of a_1 in the equation

$$\arg\min_{a_1} f(a_1) = \arg\min_{a_1} \Big((1-p) \cdot a_1 + p_w(a_1) \cdot (n - a_1) \Big). \tag{5}$$

- We then solve the equation $f(x+1) - f(x) = 0$, which has a unique solution. It can be shown that $f(a_1)$ first decreases then increases with respect to a_1. Thus the solution for $f(x+1) - f(x) = 0$ offers an approximation to the optimal a_1 with only $O(1)$ error.
- After solving the optimal size of the first layer, we can apply this method recursively to calculate the optimal size of all layers.

The proof for the above three results are complex and brute-force. First, we present a lemma that addresses result 2. It is worth pointing out that Equation 5 is the expected loss when we have only two layers, with the first layer of size a_1 and the second layer of size $n - a_1$. Therefore, result 2 is equivalent to finding an optimal topology among two layer graphs. For convenience, we denote $s = 1/(4p(1-p))$.

▶ **Lemma 21.** *For the optimal topology among two layer graphs, its first layer has size*

$$\log_s n - \log_s(\log_s n)/2 + O(1).$$

▶ **Theorem 22.** *The optimal layer topology has $k = n/\log_s n + o(n/\log_s n)$ many layers. The first layer has $a_1 = \log_s n$ many nodes. The size of layer i may be written as a recursion of the size of layer $i-1$,*

$$a_i \sim \log_s(s^{a_{i-1}} - a_{i-1}). \tag{6}$$

In other words, the optimal topology satisfies the following structural properties:

- *The sizes of layers gradually decrease, and the number of layers with size $\log_s n - i$ is $(s-1)n/(s^{i+1}(\log_s n - i))$.*

- *The first* $(s-1)n/(s\log_s n)$ *layers have size* $\log_s n$.
- *The following* $(s-1)n/(s^2(\log_s n-1))$ *layers have size* $\log_s n-1$, *and so on.*

We believe that the layer topology provided in Theorem 22 is actually the optimal topology for Majority Algorithm. However, we have not yet come up with any rigorous proof to verify our conjecture. We will leave this as an open problem for future work.

▶ **Conjecture 23.** *The layer topology provided in Theorem* 22 *is the optimal topology for Majority Algorithm.*

6.2 Experiments on the optimal parameters

For layer graphs, Equation (6) can be used to calculate the optimal parameters with high precision. Thus the results in Theorem 22 is a very tight approximation, which works flawlessly if we only seek to analyze the complexity of $\mathcal{E}$. However, in real life, users might wish to achieve the exact optimal parameters. In this case, the constant error in our equations can not be neglected. Here, we introduce an algorithm that searches for the exact optimal topology in average $O(1)$ run time.

Recall that layer graphs satisfy the following properties. For a k-layer structure $(a_1,\cdots,a_k)$, if the first layer cascades, the rest of the nodes follow this cascaded result. Otherwise, the rest of the nodes become equivalent to a $(k-1)$-layer structure, with each layer's size being $(a_2,\cdots,a_k)$ (or $(a_2+1,\cdots,a_k)$ if a_1 is odd). Therefore, for a fixed a_1, the optimal k and $a_2,\cdots,a_k$ should be chosen such that,

- If a_1 is even,

$$(k-1,a_2,\cdots,a_k)=\arg\max_{k',a_1',\cdots,a_k'}\mathcal{E}\Big(a_1',\cdots,a_k'\Big|\sum_{i=1}^{k'}a_i'=n-a_1\Big). \tag{7}$$

- If a_1 is odd,

$$(k-1,a_2+1,\cdots,a_k)=\arg\max_{k',a_1',\cdots,a_k'}\mathcal{E}\Big(a_1',\cdots,a_k'\Big|\sum_{i=1}^{k'}a_i'=n-a_1+1\Big). \tag{8}$$

This implies that given the optimal layer topologies for all $n'<n$, calculating the optimal layer topology for n should be easy. We can simplify the problem into an optimization over a_1, instead of the optimization over many parameters. A further analysis shows that the optimal layer topology for n nodes and $n+1$ nodes cannot differ by too much. More specifically, denote the optimizing parameter for n nodes as $(k_n,a_1^n,\cdots,a_{k_n}^n)$, then $|a_1^n-a_1^{n+1}|\le 1$. Using this, together with the recursive idea in Equation 7 and 8, we can design an algorithm that runs in time $O(n)$, and find the optimal layer topology for all $n'\le n$. The amortized running time of this algorithm is only $O(1)$.

7 Conclusion

In this paper, we discussed information cascades on various network topologies. We provide a non-constructive optimal algorithm scheme for general graphs, solve the scheme for complete graph and achieve a general lower bound for our model. We also studied Majority Algorithm in random graphs and layer graphs, the minimal $\mathcal{E}$ of which was shown to be asymptotically the same with our general lower bound. From the experiment results, a gap between the general lower bound and layer graphs can be observed. We believe this to be a result of

the difference in the model setting, i.e. Majority Algorithm is weaker than optimal general algorithms.

Future work in this area may include the study of the following scenarios.

- The nodes' order of decision-making is no longer fixed and given, but instead randomly sampled from all permutations.
- The topology is fixed and we are only able to add or remove a fixed portion of the edges. The goal is to minimize $\mathcal{E}$ under this constraint.
- *Plant nodes* in the network. These nodes could sacrifice themselves to reveal their true private signal. How should a topology designer control and position these plant nodes in the topology?

References

1 Daron Acemoglu, Munther A. Dahleh, Ilan Lobel, and Asuman Ozdaglar. Bayesian Learning in Social Networks. *Review of Economic Studies*, 78(4):1201–1236, 2011. URL: `https://ideas.repec.org/a/oup/restud/v78y2011i4p1201-1236.html`.

2 Daron Acemoglu and Asuman Ozdaglar. Opinion dynamics and learning in social networks. *Dynamic Games and Applications*, 1(1):3–49, 2010.

3 Lisa R Anderson and Charles A Holt. Classroom games: Information cascades. *The Journal of Economic Perspectives*, 10(4):187–193, 1996.

4 Lisa R Anderson and Charles A Holt. Information cascades in the laboratory. *The American economic review*, pages 847–862, 1997.

5 Abhijit Banerjee and Drew Fudenberg. Word-of-mouth learning. *Games and Economic Behavior*, 46(1):1–22, January 2004.

6 Abhijit V Banerjee. A simple model of herd behavior. *The Quarterly Journal of Economics*, pages 797–817, 1992.

7 Sushil Bikhchandani, David Hirshleifer, and Ivo Welch. A theory of fads, fashion, custom, and cultural change as informational cascades. *Journal of political Economy*, pages 992–1026, 1992.

8 David Bindel, Jon Kleinberg, and Sigal Oren. How bad is forming your own opinion. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 57–66, 2011.

9 Boğaçhan Çelena and Shachar Kariv. Observational learning with imperfect information. *Games and Economic Behavior*, 47:72–86, 2004.

10 Flavio Chierichetti, Jon Kleinberg, and Alessandro Panconesi. How to schedule a cascade in an arbitrary graph. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC'12, pages 355–368, New York, NY, USA, 2012. ACM.

11 Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69:118–121, 1974.

12 David Easley and Jon Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

13 Benjamin Golub and Matthew O. Jackson. Naive learning in social networks: convergence, influence, and the wisdom of crowds, 2010.

14 Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, May 1978.

15 MohammadTaghi Hajiaghayi, Hamid Mahini, and David Malec. The polarizing effect of network influences. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC'14, pages 131–148, New York, NY, USA, 2014. ACM.

16 MohammadTaghi Hajiaghayi, Hamid Mahini, and Anshul Sawant. Scheduling a cascade with opposing influences. In *Algorithmic Game Theory: 6th International Symposium, SAGT 2013, Aachen, Germany, October 21-23, 2013. Proceedings*, pages 195–206, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

17 Elias Koutsoupias and Christos Papadimitriou. Worst-case equilibria. In *STACS 99: 16th Annual Symposium on Theoretical Aspects of Computer Science Trier, Germany, March 4–6, 1999 Proceedings*, pages 404–413, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

18 David Krackhardt. A plunge into networks. *Science*, 326(5949):47–48, 2009. doi:10.1126/science.1167367.

19 Lones Smith and Peter Sørensen. Pathological outcomes of observational learning. *ECONOMETRICA*, 68:371–398, 1999.

20 Lones Smith and Peter Sørensen. Rational social learning with random sampling, 2008. URL: http://lonessmith.com/sites/default/files/rational.pdf.

21 Arthur W Brian. Competing technologies, increasing returns, and lock-in by historical events. *The Economic Journal*, 99(394):116–131, 1989.

22 Jun Wan, Yu Xia, Liang Li, and Thomas Moscibroda. Information cascades on arbitrary topologies, 2016. URL: http://arxiv.org/abs/1604.07166.

23 Ivo Welch. Sequential sales, learning, and cascades. *The Journal of finance*, 47(2):695–732, 1992.

Analysing Survey Propagation Guided Decimation on Random Formulas*†

Samuel Hetterich

Goethe University, Mathematics Institute, Frankfurt, Germany
hetterich@math.uni-frankfurt.de

Abstract

Let $\vec{\Phi}$ be a uniformly distributed random k-SAT formula with n variables and m clauses. For clauses/variables ratio $m/n \leq r_{k\text{-SAT}} \sim 2^k \ln 2$ the formula $\vec{\Phi}$ is satisfiable with high probability. However, no efficient algorithm is known to provably find a satisfying assignment beyond $m/n \sim 2k\ln(k)/k$ with a non-vanishing probability. Non-rigorous statistical mechanics work on k-CNF led to the development of a new efficient "message passing algorithm" called *Survey Propagation Guided Decimation* [Mézard et al., Science 2002]. Experiments conducted for $k = 3, 4, 5$ suggest that the algorithm finds satisfying assignments close to $r_{k\text{-SAT}}$. However, in the present paper we prove that the basic version of Survey Propagation Guided Decimation fails to solve random k-SAT formulas efficiently already for $m/n = 2^k(1+\varepsilon_k)\ln(k)/k$ with $\lim_{k\to\infty}\varepsilon_k = 0$ almost a factor k below $r_{k\text{-SAT}}$.

1998 ACM Subject Classification G.2.1 Combinatorics

Keywords and phrases Survey Propagation Guided Decimation, Message Passing Algorithm, Graph Theory, Random k-SAT

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.65

1 Introduction

Random k-SAT instances have been known as challenging benchmarks for decades [9, 28, 33]. The simplest and most intensely studied model goes as follows. Let $k \geq 3$ be an integer, fix a density parameter $r > 0$, let n be a (large) integer and let $m = \lceil rn \rceil$. Then $\Phi = \Phi_k(n, m)$ signifies a k-CNF chosen uniformly at random among all $(2n)^{km}$ possible formulas. With k, r fixed the random formula is said to enjoy a property *with high probability* if the probability that the property holds tends to 1 as $n \to \infty$.

The conventional wisdom about random k-SAT has been that the problem of finding a satisfying assignment is computationally most challenging for r below but close to the *satisfiability threshold* $r_{k-\text{SAT}}$ where the random formula ceases to be satisfiable w.h.p. [28]. Whilst the case $k = 3$ may be the most accessible from a practical (or experimental) viewpoint, the picture becomes both clearer and more dramatic for larger values of k. Asymptotically the k-SAT threshold reads $r_{k-\text{SAT}} = 2^k \ln 2 - (1 + \ln 2)/2 + \varepsilon_k$, where $\varepsilon_k \to 0$ in the limit of large k [14]. However, the best current algorithms are known to find satisfying assignments in polynomial time merely up to $r \sim 2^k \ln k/k$ [11]. In fact, standard heuristics such as Unit Clause Propagation bite the dust for even smaller densities, namely $r = c2^k/k$ for a certain

* A full version of this paper is available at http://arxiv.org/abs/1602.08519.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 278857–PTCC

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 65; pp. 65:1–65:12

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

absolute constant $c > 0$ [17]. The same goes (provably) for various DPLL-based solvers [1, 30]. Hence, there is a factor of about $k/\ln k$ between the algorithmic threshold and the actual satisfiability threshold.

In the early 2000s physicists put forward a sophisticated but non-rigorous approach called the *cavity method* to tackle problems such as random k-SAT both analytically and algorithmically. In particular, the cavity method yields a *precise* prediction as to the value of $r_{k-\mathrm{SAT}}$ for any $k \geq 3$ [24, 26], which was recently verified rigorously for sufficiently large values of k [14]. Additionally, the cavity method provided a heuristic explanation for the demise of simple combinatorial or DPLL-based algorithms well below $r_{k-\mathrm{SAT}}$. Specifically, the density $2^k \ln k/k$ marks the point where the geometry of the set of satisfying assignments changes from (essentially) a single connected component to a collection of tiny well-separated clusters [22]. In fact, a typical satisfying assignment belongs to a "frozen" cluster, i.e., there are extensive long-range correlations between the variables. The cluster decomposition as well as the freezing prediction have largely been verified rigorously [29, 3] and we begin to understand the impact of this picture on the performance of algorithms [2].

But perhaps most remarkably, the physics work has led to the development of a new efficient "message passing algorithm" called *Survey Propagation Guided Decimation* to overcome this barrier [6, 21, 27, 31]. More precisely, the algorithm is based on a heuristic that is designed to find whole frozen clusters not only single satisfying assignments by identifying each cluster by the variables determined by long-range correlations and locally "free" variables. Thus, by its very design *Survey Propagation Guided Decimation* is built to work at densities where frozen clusters exist. Although the experimental performance for small k is outstanding this yields no evidence of a relation between the occurrence of frozen clusters and the success of the algorithm. Yet not even the physics methods lead to a precise explanation of these empirical results or to a prediction as to the density up to which we might expect SP to succeed for general values of k. In effect, analysing SP has become one of the most important challenges in the context of random constraint satisfaction problems.

The present paper furnishes the first rigorous analysis of `SPdec` (the basic version of) Survey Propagation Guided Decimation for random k-SAT. We give a precise definition and detailed explanation below. Before we state the result let us point out that two levels of randomness are involved: the choice of the random formula $\vec{\Phi}$, and the "coin tosses" of the randomized algorithm `SPdec`. For a (fixed, non-random) k-CNF Φ let success(Φ) denote the probability that `SPdec`(Φ) outputs a satisfying assignment. Here, of course, "probability" refers to the coin tosses of the algorithm only. Then, if we apply `SPdec` to the *random* k-CNF $\vec{\Phi}$, the success probability success($\vec{\Phi}$) becomes a random variable. Recall that $\vec{\Phi}$ is unsatisfiable for $r > 2^k \ln 2$ w.h.p..

▶ **Theorem 1.** *There is a sequence* $(\varepsilon_k)_{k\geq 3}$ *with* $\lim_{k\to\infty} \varepsilon_k = 0$ *such that for any* k, r *satisfying* $2^k(1+\varepsilon_k)\ln(k)/k \leq r \leq 2^k \ln 2$ *we have* $\mathrm{success}(\vec{\Phi}) \leq \exp(-\Omega(n))$ *w.h.p.*

If the success probability is exponential small in n sequentially running `SPdec` a sub-exponential number of times will not find a satisfying assignment w.h.p. rejecting the hypotheses that `SPdec` solves random k-SAT formulas efficiently for considered clauses/variables ratio. Thus, Theorem 1 shows that `SPdec` does not outclass far simpler combinatorial algorithms for general values of k. Even worse, in spite of being designed for this very purpose, the SP algorithm does *not* overcome the barrier where the set of satisfying assignments decomposes into tiny clusters asymptotically. This is even more astonishing since it is possible to *prove* the existence of satisfying assignments up to the satisfiability threshold rigorously based on the cavity method but algorithms designed by insights of this approach fail far below that threshold. Nevertheless, let me note that the insights gained from Theorem 1

is actually in line with some non-rigorous physics work on the SP algorithm. Still, there is some arguing if there is any connection between the failure of algorithms and either the clustering or the so called freezing phenomenon. Both, neither the connection to clustering nor to freezing have been rigorously proven yet.

We are going to describe the SP algorithm in the following section. Let us stress that Theorem 1 pertains to the "vanilla" version of the algorithm. Unsurprisingly, more sophisticated variants with better empirical performance have been suggested, even ones that involve backtracking [23]. Also the first version introduced by Mézard, Parisi and Zecchina [27] contained a bias towards "frozen" variables for the choice of the variable at each decimation step. However, the basic version of the SP algorithm analysed in the present paper arguably (regarding the physicists picture of freezing, correlation decay, replica symmetry assumption [26]) encompasses all the conceptually important features of the SP algorithm.

The only prior rigorous result on the Survey Propagation algorithm is the work of Gamarnik and Sudan [19] on the k-NAESAT problem (where the goal is to find a satisfying assignment whose complement is satisfying as well). However, Gamarnik and Sudan study a "truncated" variant of the algorithm where only a bounded number of message passing iterations is performed. The main result of [19] shows that this version of Survey Propagation fails for densities about a factor of $k/\ln^2 k$ below the NAE-satisfiability threshold and about a factor of $\ln k$ above the density where the set of NAE-satisfying assignments shatters into tiny clusters. Though, experimental data and the conceptional design of the SP algorithm suggest that it exploits its strength in particular by iterating the message passing iterations a unbounded number of times that depends on n. In particular, to gather information from the set of messages they have to converge to a fixed point which turns out to happen only after a number of iterations of order $\ln(n)$.

An in-depth introduction to the cavity method and its impact on combinatorics, information theory and computer science can be found in [25, 26].

2 The SPdec algorithm

The proof of Theorem 1 is by extension of the prior analysis [10] of the much simpler *Belief Propagation Guided Decimation* algorithm. To outline the proof strategy and to explain the key differences, we need to discuss the SP algorithm in detail. For a k-CNF Φ on the variables $V = \{x_1, \dots, x_n\}$ we generally represent truth assignments as maps $\sigma : V \to \{-1, 1\}$, with -1 representing "false" and 1 representing "true". Survey Propagation is an efficient message passing heuristic on the factor graph $G(\Phi)$. The factor graph of Φ is a bipartite graph representation of Φ where each clause and each variable is represented by a vertex. Two vertices are incident if the corresponding variable is contained in the corresponding clause [26].

Before explaining the Survey Propagation heuristic, we explain the simpler Belief Propagation heuristic and emphasize the main extensions later on. To define the messages involved we denote the ordered pair (x, a) with $x \to a$ and similarly (a, x) with $a \to x$ for each $x \in V$ and $a \in N(x)$, where $N(x)$ denotes the neighborhood in the factor graph $G(\Phi)$. The messages are iteratively sent probability distributions $(\mu_{x\to a}(\zeta))_{x\in V_t, a\in N(x), \zeta\in\{-1,1\}}$ over $\{-1, 1\}$. In each iteration messages are sent from variables to adjacent clauses and back. After setting initial messages due to some initialization rule the messages sent are obtained by applying a function to the set of incoming messages at each vertex. Both the initialization and the particular update rules at the vertices are specifying the message passing algorithm. The messages are updated $\omega(n)$ times which may or may not depend on n. A detailed explanation of the Belief Propagation heuristic can be found in [8, p. 519].

It is well known that the Belief Propagation messages on a tree converge after updating the messages two times the depth of the tree to a fixed point. Moreover, in this case for each variable the marginal distribution of the uniform distribution on the set of all satisfying assignments can be computed by the set of the fixed point messages. Since $G(\Phi)$ for constant clauses/variables ratio contains only a small number of short cycles one may expect that on the base of the Belief Propagation messages a good estimate of the marginal distribution of the uniform distribution on the set of all satisfying assignments of Φ could be obtained. Besides the fact that it is not even clear that the messages converge to a fixed point on arbitrary graphs this is of course only a weak heuristic explanation which is refuted by [10]. However, at each decimation step using the Belief Propagation heuristic the Belief Propagation guided decimation algorithm assigns one variable due to the estimated marginal distribution to -1 or 1. Simplifying the formula and running Belief Propagation on the simplified formula and repeating this procedure would lead to a satisfying assignment chosen uniformly at random for sure if the marginals were correct at each decimation step.

Let us now introduce the Survey Propagation heuristic. As mentioned above the geometry of the set of satisfying assignments comes as a collection of tiny well-separated clusters above density $2^k \ln(k)/k$. In that regime a typical solution belongs to a "frozen" cluster. That is all satisfying assignments in such a frozen cluster agree on a linear number of frozen variables. Thus, identifying these frozen variables gives a characterization of the whole cluster. Flipping one of these variables leads to a set of unsatisfied clauses only containing additional frozen variables. Satisfying one of these clauses leads to further unsatisfied clauses of this kind ending up in an avalanche of necessary flippings to obtain a satisfying assignment. This ends only after a linear number of flippings. Given a satisfying assignment with identified frozen variables each satisfying assignment that disagrees on one of these frozen variables has linear distance therefore belonging to a different cluster.

This picture inspires the definition of *covers* as generalized assignments $\sigma \in \{-1, 0, 1\}^n$ such that

- each clause either contains a true literal or two 0 literals and
- for each variable $x \in V$ that is assigned -1 or 1 exists a clause $a \in N(x)$ such that for all $y \in N(a) \setminus \{x\}$ we have $\operatorname{sign}(y, a) \cdot \sigma(y) = -1$.

These two properties mirrors the situation in frozen clusters where assigning a variable to the value 0 indicates that these variable supposes to be free in the corresponding cluster which is obtained by only flipping 0 variables to one of the values -1 or 1. However, implementing the concept of covers, Survey Propagation is a heuristic of computing the marginals over the set of covers by using the Belief Propagation update rules on covers. This leads to the equations given by Figure 1. For a more detailed explanation of the freezing phenomenon we point the reader to [29]. For a deeper discussion on covers we refer to [12].

We are now ready to state the `SPdec` algorithm.

▶ **Algorithm 2.** `SPdec`(Φ)
Input: A k-CNF Φ on $V = \{x_1, \ldots, x_n\}$. *Output:* An assignment $\sigma : V \to \{-1, 1\}$.

0. Let $\Phi_0 = \Phi$.
1. For $t = 0, \ldots, n-1$ do
2. Use SP to compute $\mu^{[\omega]}_{x_{t+1}}(\Phi_t)$.
3. Assign

$$\sigma(x_{t+1}) = \begin{cases} 1 & \text{with probability } \mu^{[\omega]}_{x_{t+1}}(\Phi_t) \\ -1 & \text{with probability } 1 - \mu^{[\omega]}_{x_{t+1}}(\Phi_t). \end{cases} \tag{7}$$

4. Obtain a formula Φ_{t+1} from Φ_t by substituting the value $\sigma(x_{t+1})$ for x_{t+1} and simplifying.
5. Return the assignment σ.

For real numbers $0 \leq x, y \leq 1$ such that $\max\{x, y\} > 0$ we define

$$\psi_\zeta(x,y) = \begin{cases} xy \cdot \Psi(x,y) & \text{if } \zeta = 0 \\ (1-x)y \cdot \Psi(x,y) & \text{if } \zeta = 1 \\ (1-y)x \cdot \Psi(x,y) & \text{if } \zeta = -1 \end{cases}, \qquad \Psi(x,y) = (x+y-xy)^{-1}$$

If $x = y = 0$ set $\psi_0(0) = 0$ and $\psi_{\pm 1}(0) = \frac{1}{2}$. Define for all $x \in V_t, a, b \in N(x), \zeta \in \{-1, 0, 1\}$ and $\ell \geq 0$

$$\mu^{[0]}_{x\to a}(\pm 1) = \frac{1}{2}, \qquad \mu^{[0]}_{x\to a}(0) = 0, \qquad \mu^{[\ell]}_{b\to x}(0) = 1 - \prod_{y \in N(b)\setminus\{x\}} \mu^{[\ell]}_{y\to b}(-\text{sign}(y,b)) \tag{1}$$

$$\pi^{[\ell+1]}_{x\to a}(\pm 1) = \prod_{b \in N(x,\pm 1)\setminus\{a\}} \mu^{[\ell]}_{b\to x}(0) \tag{2}$$

$$\mu^{[\ell+1]}_{x\to a}(\zeta) = (SP(\mu^{[\ell]}))_{x\to a}(\zeta) = \psi_\zeta(\pi^{[\ell]}_{x\to a}(1), \pi^{[\ell]}_{x\to a}(-1)). \tag{3}$$

Let $\omega = \omega(k, r, n) \geq 0$ be any integer-valued function. Define

$$\pi^{[\omega+1]}_x(\Phi_t, \pm 1) = \prod_{b \in N(x,\pm 1)} \mu^{[\omega]}_{b\to x}(0) \tag{4}$$

$$\mu^{[\omega]}_x(\Phi_t, \zeta) = \psi_\zeta(\pi^{[\omega+1]}_x(\Phi_t, 1) \cdot \pi^{[\omega+1]}_x(\Phi_t, -1)) \tag{5}$$

$$\mu^{[\omega]}_x(\Phi_t) = \frac{\mu^{[\omega]}_x(\Phi_t, 1)}{\mu^{[\omega]}_x(\Phi_t, 1) + \mu^{[\omega]}_x(\Phi_t, -1)} = \mu^{[\omega]}_x(\Phi_t, 1) + \frac{1}{2}\mu^{[\omega]}_x(\Phi_t, 0). \tag{6}$$

Figure 1 The Survey Propagation equations that are the Belief Propagation equations on covers.

Let us emphasize that the value $\mu^{[\omega]}_{x_{t+1}}(\Phi_t)$ in Step 2 of `SPdec` is the estimated marginal probability over the set of covers of variable x_{t+1} in the simplified formula to take the value 1 plus one half the estimated marginal probability over the set of covers in the simplified formula to take the value 0. This makes sense since by the heuristic explanation a variable assigned to the value 0 is free to take either value 1 or -1. Thus, our task is to study the SP operator on the decimated formula Φ_t.

3 Proof of Theorem 1

The probabilistic framework used in our analysis of `SPdec` was introduced in [10] for analysing the *Belief Propagation Guided Decimation* algorithm. The most important technique in analysing algorithms on the random formula $\vec{\Phi}$ is the "method of deferred decisions", which traces the dynamics of an algorithm by differential equations, martingales, or Markov chains. It actually applies to algorithms that decide upon the value of a variable x on the basis of the clauses or variables at small bounded distance from x in the factor graph [5]. Unfortunately, the `SPdec` algorithm at step t explores clauses at distance 2ω from x_t where $\omega = \omega(n)$ may tend to infinity with n. Therefore, the "defered decisions" method does not apply and to prove Theorem 1 a fundamentally different approach is needed.

We will basically reduce the analysis of `SPdec` to the problem of analysing the SP operator on the random formula $\vec{\Phi}^t$ that is obtained from $\vec{\Phi}$ by substituting "true" for the first t variables $x_1, \ldots, x_t$ and simplifying (see Theorem 3 below). In the following sections we will prove that this decimated formula has a number of simple to verify quasirandomness

properties with very high probability. Finally, we will show that it is possible to trace the Survey Propagation algorithm on a formula Φ enjoying this properties.

Applied to a fixed, non-random formula Φ on $V = \{x_1, \ldots, x_n\}$, `SPdec` yields an assignment $\sigma : V \to \{-1, 1\}$ that may or may not be satisfying. This assignment is random, because `SPdec` itself is randomized. Hence, for any fixed Φ running `SPdec`(Φ) induces a probability distribution β_Φ on $\{-1, 1\}^V$. With $\mathcal{S}(\Phi)$ the set of all satisfying assignments of Φ, the "success probability" of `SPdec` on Φ is just

$$\mathrm{success}(\Phi) = \beta_\Phi(\mathcal{S}(\Phi)). \tag{8}$$

Thus, to establish Theorem 1 we need to show that in the *random* formula

$$\mathrm{success}(\vec{\Phi}) = \beta_{\vec{\Phi}}(s(\vec{\Phi})) = \exp\left(-\Omega(n)\right) \tag{9}$$

is exponentially small w.h.p. To this end, we are going to prove that the measure $\beta_{\vec{\Phi}}$ is "rather close" to the uniform distribution on $\{-1, 1\}^V$ w.h.p., of which $\mathcal{S}(\vec{\Phi})$ constitutes only an exponentially small fraction. However, to prove Theorem 1 we prove that the entropy of the distribution $\beta_{\vec{\Phi}}$ is large. Let us stress that this is not by Moser's entropy compression argument which works up to far smaller clauses/variables ratios [32].

3.1 Lower bounding the entropy

Throughout the paper we let $\rho_k = (1 + \varepsilon_k)\ln(k)$ where $(\varepsilon_k)_{k\geq 3}$ is the sequence promised by Theorem 1 and let $r = 2^k\rho$ where $\rho \geq \rho_k$.

For a number $\delta > 0$ and an index $i > t$ we say that x_i is (δ, t)-*biased* if

$$\left|\mu_{x_i}^{[\omega]}(\Phi^t, 1) - \frac{1}{2}\left(1 - \mu_{x_i}^{[\omega]}(\Phi^t, 0)\right)\right| > \delta. \tag{10}$$

Moreover Φ is (δ, t)-*balanced* if no more than $\delta(n - t)$ variables are (δ, t)-biased.

If $\vec{\Phi}$ is (δ, t)-balanced, then by the basic symmetry properties of $\vec{\Phi}$ the probability that x_{t+1} is (δ, t)-biased is bounded by δ. Furthermore, given that x_{t+1} is not (δ, t)-biased, the probability that `SPdec` will set it to "true" lies in the interval $[\frac{1}{2} - \delta, \frac{1}{2} + \delta]$. Consequently,

$$\left|\frac{1}{2} - \mathbb{P}\left[\sigma(x_{t+1}) = 1 | \vec{\Phi} \text{ is } (\delta, t)\text{-balanced}\right]\right| \leq 2\delta. \tag{11}$$

Thus, the smaller δ the closer $\sigma(x_{t+1})$ comes to being uniformly distributed. Hence, if (δ, t)-balancedness holds for all t with a "small" δ, then β_Φ will be close to the uniform distribution on $\{-1, 1\}^V$.

To put this observation to work, let $\theta = 1 - t/n$ be the fraction of unassigned variables and define

$$\delta_t = \exp(-c\theta k), \qquad \Delta_t = \sum_{s=1}^{t}\delta_t \qquad \text{and} \qquad \hat{t} = \left(1 - \frac{\ln(\rho)}{c^2 k}\right)n, \tag{12}$$

where $c > 0$ is a small enough absolute constant.

The following result provides the key estimate by providing that at any time t up to $\hat{t}$ with sufficiently high probability $\vec{\Phi}$ is (δ_t, t)-balanced with a sufficiently small δ_t to finally prove Theorem 1.

▶ **Proposition 3.** *For any k, r satisfying $2^k\rho_k/k < r \leq 2k\ln 2$ there is $\xi = \xi(k, r) \in [0, \frac{1}{k}]$ so that for n large enough the following holds. For any $0 \leq t \leq \hat{t}$ we have*

$$\Pr\left[\vec{\Phi} \textit{ is } (\delta_t, t)\textit{-balanced}\right] \geq 1 - \exp\left[-3\xi n - 10\Delta_t\right]. \tag{13}$$

3.2 Tracing the Survey Propagation Operator

To establish Proposition 3 we have to prove that $\vec{\Phi}$ is (δ_t, t)-balanced with probability very close to one. Thus, our task is to study the SP operator defined in (1) to (3) on $\vec{\Phi}^t$. Roughly speaking, Proposition 3 asserts that with probability very close to one, most of the messages $\mu^{[\ell]}_{x\to a}(\pm 1)$ are close to $\frac{1}{2}(1-\mu^{[\ell]}_{x\to a}(0))$. To obtain this bound, we are going to proceed in two steps: we will exhibit a small number of *quasirandomness properties* and show that these hold in $\vec{\Phi}^t$ with the required probability. Then, we prove that *deterministically* any formula that has these properties is (δ_t, t)-balanced.

3.2.1 The "typical" value of $\pi^{[\ell]}_{x\to a}(\zeta)$

First of all recall that the messages sent from a variable x to a clause $a \in N(x)$ are obtained by

$$\psi_\zeta(\pi^{[\ell]}_{x\to a}(1), \pi^{[\ell]}_{x\to a}(-1)) \qquad \text{for } \zeta \in \{-1, 0, 1\}. \tag{14}$$

This in mind, we claim a strong statement that both $\pi^{[\ell]}_{x\to a}(1)$ and $\pi^{[\ell]}_{x\to a}(-1)$ are very close to a "typical" value $\pi[\ell]$ for most of the variables $x \in V_t$ and clauses $a \in N(x)$ at any iteration step ℓ under the assumption that the set of biased variables is small at time $\ell - 1$. Assuming that

$$\pi^{[\ell]}_{x\to a}(1) = \pi^{[\ell]}_{x\to a}(-1) = \pi[\ell]$$

we of course obtain unbiased messages by

$$\mu^{[\ell]}_{x\to a}(\pm 1) = \psi_1(\pi[\ell]) = \psi_{-1}(\pi[\ell]) = \frac{1}{2}(1 - \mu^{[\ell]}_{x\to a}(0)).$$

The products $\pi^{[\ell]}_{x\to a}(\zeta)$ are nothing else but the product of the messages

$$\mu^{[\ell-1]}_{b\to x}(0) = 1 - \prod_{y\in N(b)\setminus\{x\}} \mu^{[\ell-1]}_{y\to b}(-\operatorname{sign}(y,b))$$

sent from all clauses $b \in N(x, \zeta) \setminus \{a\}$ to x. Therefore, we define inductively $0 \le \pi[\ell] \le 1$ to be the product of this kind over a "typical" neighborhood. The term "typical" refers to the expected number of clauses of all lengths that contain at most one additional biased variable. Focusing on those clauses will suffice to get the tightness result of the biases. Moreover, we assume that all of the messages $\mu^{[\ell-1]}_{y\to b}(-\operatorname{sign}(y,b))$ sent from variables to clauses in such a typical neighborhood are $\psi_{\operatorname{sign}(y,b)}(\pi[\ell-1], \pi[\ell-1])$ which is claimed to be a good estimation of most of the messages sent at time $\ell - 1$. Additionally, define $\tau[\ell] = (1 - \psi_0(\pi[\ell]))$ as the estimate of the sum $\mu^{[\ell]}_{x\to a}(1) + \mu^{[\ell]}_{x\to a}(-1)$. Let us emphasize that there is no "unique" $\pi[\ell]$ and the way it is obtained in the following is in some sense the canonical and convenient choice to sufficiently bound the biases for most of the messages.

Generally, let $T \subset V_t$ and $x \in V_t$. Then the expected number of clauses of length j that contain x and at most one other variable from the set T is asymptotically

$$\mu_{j,\le 1}(T) = 2^j\rho \cdot \Pr\left[\operatorname{Bin}(k-1,\theta) = j-1\right] \cdot \Pr\left[\operatorname{Bin}\left(j-1, \frac{|T|}{\theta n}\right) < 2\right]. \tag{15}$$

Indeed, the expected number of clauses of $\vec{\Phi}$ that x appears in equals $km/n = kr = 2^k\rho$. Furthermore, each of these gives rise to a clause of length j in $\vec{\Phi}^t$ iff exactly $j-1$ among the other $k-1$ variables in the clauses are from V_t while the $k-j$ remaining variables are

in $V \setminus V_t$ and occur with negative signs. (If one of them had a positive sign, the clause would have been satisfied by setting the corresponding variable to true. It would thus not be present in $\vec{\Phi}^t$ anymore.) Moreover, at most one of the $j-1$ remaining variables is allowed to be from the set T. The fraction of variables in T in V_t equals $\frac{|T|}{\theta n}$. Finally, since x appears with a random sign in each of these clauses the expected number of clauses of length j that contain x and at most one other variable from the set T is asymptotically $\mu_{j,\leq 1}(t)/2$.

Additionally let $0 \leq p \leq 1$ and define

$$\tau(p) = 1 - \psi_0(p) \qquad \text{and} \qquad \pi(T,p) = \prod_{j=0.1\theta k}^{10\theta k} \left(1 - (2/\tau(p))^{-j+1}\right)^{\mu_{j,\leq 1}(T)/2}. \tag{16}$$

Moreover, let

$$\Pi(T,p) = \sum_{j=0.1\theta k}^{10\theta k} \frac{\mu_{j,\leq 1}(T)}{2} \cdot (2/\tau(p))^{-j+1}$$

be the approximated absolute value of the logarithm of $\pi(T,p)$.

For a fixed variable $x \in V_t$ the expected number of clauses that contain more than one additional variable from a "small" set T for a "typical" clause length $0.1\theta k \leq j \leq 10\theta k$ is very close to the expected number of all clauses of that given length. Thus, the actual size of T will influence $\pi(T,p)$ but this impact is small if T is small and the following bounds on $\pi(T,p)$ can be achieved.

▶ **Lemma 4.** *Let $T \subset V_t$ of size $|T| \leq \delta\theta n$ and $0 \leq p \leq 2\exp(-\rho)$. Then $\exp(-2\rho) \leq \pi(T,p) \leq 2\exp(-\rho)$.*

3.2.2 Bias

First of all let us define the bias not only for the 1 and -1 messages but also for the 0 messages. Hence, for $\ell \geq 0, x \in V_t$ and $a \in N(x)$ let

$$\Delta^{[\ell]}_{x\to a} = \mu^{[\ell]}_{x\to a}(1) - \frac{1}{2}\left(1 - \mu^{[\ell]}_{x\to a}(0)\right) \qquad \text{and} \tag{17}$$

$$E^{[\ell]}_{x\to a} = \frac{1}{2}\left(\mu^{[\ell]}_{x\to a}(0) - \psi_0(\pi[\ell])\right). \tag{18}$$

We say that $x \in V_t$ is *ℓ-biased* if

$$\max_{a\in N(x)} |\Delta^{[\ell]}_{x\to a}| > 0.1\delta \qquad \text{or} \qquad \max_{a\in N(x)} |E^{[\ell]}_{x\to a}| > 0.1\delta\pi[\ell] \tag{19}$$

and *ℓ-weighted* if

$$\max_{a\in N(x)} |E^{[\ell]}_{x\to a}| > 10\pi[\ell]. \tag{20}$$

Let $B[\ell]$ be the set of all ℓ-biased variables and $B'[\ell]$ be the set of all ℓ-weighted variables. Obviously, by definition, we have $B'[\ell] \subset B'[\ell]$.

Writing $\mu^{[\ell]}_{x\to a}(\mathrm{sign}(x,a))$ in terms of the biases we obtain

$$\begin{aligned}\mu^{[\ell]}_{x\to a}(\mathrm{sign}(x,a)) &= \frac{1}{2}(1-\psi_0(\pi[\ell])) - \left(E^{[\ell]}_{x\to a} + \mathrm{sign}(x,a)\Delta^{[\ell]}_{x\to a}\right)\\ &= \tau[\ell]/2 - \left(E^{[\ell]}_{x\to a} + \mathrm{sign}(x,a)\Delta^{[\ell]}_{x\to a}\right)\end{aligned} \tag{21}$$

We are going to prove that $|\Delta^{[\ell]}_{x\to a}|$ and $|E^{[\ell]}_{x\to a}|$ are small for most x and $a \in N(x)$. That is, given the $\Delta^{[\ell]}_{x\to a}$ and $E^{[\ell]}_{x\to a}$ we need to prove that the biases $\Delta^{[\ell+1]}_{x\to a}$ and $E^{[\ell+1]}_{x\to a}$ do not

"blow up". The proof is by induction where the hypothesis is that at most $\delta_t\theta n$ variables are ℓ-biased and at most $\delta^2\theta n$ variables are ℓ-weighted and our goal is to show that the same holds true for $\ell+1$.

3.2.3 The quasirandomness property

We will now exhibit a few simple quasirandomness properties that $\vec{\Phi}^t$ is very likely to possess. Based only on these graph properties we identify potentially ℓ-biased or ℓ-weighted variables. In turn, we prove that variables in the complement of these sets are surely not ℓ-biased resp. ℓ-weighted. Moreover, we show that these sets are small enough with sufficiently high probability.

To state the quasirandomness properties, fix a k-CNF Φ. Let Φ^t denote the CNF obtained from Φ by substituting "true" for $x_1,\dots,x_t$ and simplifying $(1 \le t \le n)$. Let $V_t = \{x_{t+1},\dots,x_n\}$ be the set of variables of Φ^t. Let $\delta = \delta_t$. With $c > 0$ we let $k_1 = \sqrt{c}\theta k$. For a variable $x \in V_t, \zeta \in \{1,-1\}$ and a set $T \subset V_t$ let

$$\begin{aligned}
\mathcal{N}(x,\zeta) &= \{b \in N(x,\zeta) : 0.1\theta k \le |N(b)| \le 10\theta k\}, \\
\mathcal{N}_{\le 1}(x,T,\zeta) &= \{b \in \mathcal{N}(x,\zeta) : |N(b) \cap T \setminus \{x\}| \le 1\}, \\
\mathcal{N}_i(x,T,\zeta) &= \{b \in \mathcal{N}(x,\zeta) : |N(b) \cap T \setminus \{x\}| = i\} \text{ for } i \in \{0,1\}, \\
N_1(x,T,\zeta) &= \{b \in N(x,\zeta) : |N(b) \setminus T| \ge k_1 \wedge |N(b) \cap T \setminus \{x\}| = 1\}, \\
N_{>1}(x,T,\zeta) &= \{b \in N(x,\zeta) : |N(b) \setminus T| \ge k_1 \wedge |N(b) \cap T \setminus \{x\}| > 1\}.
\end{aligned}$$

Thus, $\mathcal{N}_{\le 1}(x,T,\zeta)$ is the set of all clauses a that contain x with $\operatorname{sign}(x,a) = \zeta$ (which may or may not be in T) and at most one other variable from T. In addition, there is a condition on the length $|N(b)|$ of the clauses b in the decimated formula Φ^t. Having assigned the first t variables, we should "expect" the average clause length to be θk. The sets $\mathcal{N}_i(x,T,\zeta)$ are a partition of $\mathcal{N}_{\le 1}(x,T,\zeta)$ separating clauses that contain exactly one additional variable from $T \setminus \{x\}$ and clauses that contain none.

Q1. No more than $10\delta\theta n$ variables occur in clauses of length less than $\theta k/10$ or greater than $10\theta k$ in Φ_t. Moreover, there are at most $10^{-4}\delta\theta n$ variables $x \in V_T$ such that
$$(\theta k)^3\delta \cdot \sum_{b \in N(x,\zeta)} 2^{-|N(b)|} > 1.$$

Q2. For any set $T \subset V_t$ of size $|T| \le s\theta n$ such that $\delta^5 \le s \le 10\delta$ and any $p \in (0,1]$ there are at most $10^{-3}\delta^2\theta n$ variables x such that for one $\zeta \in \{-1,1\}$ either
$$\left|\Pi(T,p) - \sum_{b \in \mathcal{N}_{\le 1}(x,T,\zeta)} (2/\tau(p))^{1-|N(b)|}\right| > 2\delta/1000 \quad \text{or}$$
$$\sum_{b \in \mathcal{N}_1(x,T,\zeta)} 2^{-|N(b)|} > 10^4\rho\theta k s \quad \text{or}$$
$$\sum_{b \in \mathcal{N}_{\le 1}(x,T,\zeta)} 2^{-|N(b)|} > 10^4\rho.$$

Q3. If $T \subset V_t$ has size $|T| \le \delta\theta n$, then there are no more than $10^{-4}\delta\theta n$ variables x such that at least for one $\zeta \in \{-1,1\}$
$$\sum_{b \in N_{>1}(x,T,\zeta)} 2^{|N(b)\cap T\setminus\{x\}|-|N(b)|} > \delta/(\theta k).$$

Q4. For any $0.01 \le z \le 1$ and any set $T \subset V_t$ of size $|T| \le 100\delta\theta n$ we have
$$\sum_{b : |N(b)\cap T| \ge z|N(b)|} |N(b)| \le \frac{1.01}{z}|T| + 10^{-4}\delta\theta n.$$

Q5. For any set $T \subset V_t$ of size $|T| \leq 10\delta\theta n$, any $p \in (0,1]$ and any $\zeta \in \{-1,1\}$ the linear operator $\Lambda(T,\mu,\zeta) : \mathbb{R}^{V_t} \to \mathbb{R}^{V_t}$,

$$\Gamma = (\Gamma_y)_{y\in V_t} \mapsto \left\{ \sum_{b\in\mathcal{N}_{\leq 1}(x,T,\zeta)} \sum_{y\in N(b)\setminus\{x\}} (2/\tau(p))^{-|N(b)|} \operatorname{sign}(y,b)\Gamma_y \right\}$$

has norm $\| \Lambda(T,\mu,\zeta) \|_\square \leq \delta^4\theta n$, where for a real $b \times a$ matrix Λ we let $\|\Lambda\|_\square = \max_{\zeta\in\mathbb{R}^a\setminus\{0\}} \frac{\|\Lambda\zeta\|_1}{\|\zeta\|_\infty}$.

▶ **Definition 5.** Let $\delta > 0$. We say that Φ is (δ,t)-*quasirandom* if ***Q0-Q5*** are satisfied.

Condition **Q0** simply bounds the number of redundant clauses and the number of variables of very high degree; it is well-known to hold for random k-CNFs w.h.p. Apart from a bound on the number of very short/very long clauses, **Q1** provides a bound on the "weight" of clauses in which variables $x \in V_t$ typically occur, where the weight of a clause b is $2^{-|N(b)|}$. Moreover, **Q2** and **Q3** provide that there is no small set T for which the total weight of the clauses touching that set is very big. In addition, **Q2** (essentially) requires that for most variables x the weights of the clauses where x occurs positively/negatively should approximately cancel. Further, **Q4** provides a bound on the lengths of clauses that contain many variables from a small set T. Finally, the most important condition is **Q5**, providing a bound on the cut norm of a signed, weighted matrix, representation of Φ^t.

▶ **Proposition 6.** *There is a sequence $(\varepsilon_k)_{k\geq 3}$ with $\lim_{k\to\infty}\varepsilon_k = 0$ such that for any k, r satisfying $2^k(1+\varepsilon_k)\ln(k)/k \leq r \leq 2^k \ln 2$ there is $\xi = \xi(k,r) \in [0, \frac{1}{k}]$ so that for n large and $\delta_t, \hat{t}$ as in (12) for any $1 \leq t \leq \hat{t}$ we have*

$$P\left[\Phi \text{ is } (\delta_t,t)\text{-quasirandom}\right] \geq 1 - \exp\left(-10(\xi n + \Delta_t)\right)$$

▶ **Theorem 7.** *There is a sequence $(\varepsilon_k)_{k\geq 3}$ with $\lim_{k\to\infty}\varepsilon_k = 0$ such that for any k, r satisfying $2^k(1+\varepsilon_k)\ln(k)/k \leq r \leq 2^k \ln 2$ and n sufficiently large the following is true.*

Let Φ be a k-CNF with n variables and m clauses that is (δ_t,t)-quasirandom for some $1 \leq t \leq \hat{t}$. Then Φ is (δ_t,t)- balanced.

The proof of Proposition 6 is a necessary evil: it is long, complicated and based on standard arguments. Theorem 7 together with Proposition 6 yields Proposition 3.

3.2.4 Setting up the induction

To prove Theorem 7 we proceed by induction over ℓ. In particular we define sets $T[\ell]$ and $T'[\ell]$ that contain variables that are potentially ℓ-biased or ℓ-weighted only depending on the graph structure and the size of the sets $T[\ell-1]$ and $T'[\ell-1]$. The exact definition of the sets $T[\ell]$ and $T'[\ell]$ which inspired the quasirandomness properties are omitted in this extended abstract. It actually will turn out that $T[\ell] \subset B\ell$ and $T'[\ell] \subset B'\ell$. Since we are going to trace the SP operator on Φ^t iterated from the initial set of messages $\mu^{[0]}_{x\to a}(\pm 1) = \frac{1}{2}$ and $\mu^{[0]}_{x\to a}(0) = 0$ for all $x \in V_t$ and $a \in N(x)$ we set $T[\ell] = T'[\ell] = \emptyset$ and $\pi[0] = 0$ such that $\tau[0] = 1$. Now we define inductively

$$\pi[\ell+1] = \pi\left(T[\ell], \pi[\ell]\right), \quad \Pi[\ell+1] = \Pi\left(T[\ell], \pi[\ell]\right) \quad \text{and} \quad \tau[\ell+1] = \tau\left(\pi[\ell+1]\right).$$

▶ **Proposition 8.** *Assume that $\pi[\ell] \leq 2\exp(-\rho)$. We have $B[\ell] \subset T[\ell]$ and $B'[\ell] \subset T'[\ell]$ for all $\ell \geq 0$.*

Furthermore, we establish the following bounds on the size of $T[\ell]$ and $T'[\ell]$. Since the sets are defined by graph properties independent from the actual state of the algorithm the quasirandomness properties suffice to obtain

▶ **Proposition 9.** *If Φ is (δ_t, t)-quasirandom, we have $T[\ell] < \delta\theta n, T'[\ell] < \delta^2\theta n$ and $\pi[\ell] \leq 2\exp(-\rho)$ for all $\ell \geq 0$.*

Finally, let us give an idea how this is actually proved. We aim to prove that for most variables $x \in V_t$ for all $a \in N(x)$ simultaneously for both $\zeta \in \{-1, 1\}$ the values $\pi^{[\ell]}_{x\to a}(\zeta)$ are close to a typical value which is estimated by $\pi[\ell]$ for each iteration. Let us define for $x \in V_t, a \in N(x)$ and $\zeta \in \{1, -1\}$

$$\begin{aligned} P^{[\ell+1]}_{\leq 1}(x \to a, \zeta) &= \prod_{b \in \mathcal{N}_{\leq 1}(x, T[\ell], \zeta)\setminus\{a\}} \mu^{[\ell]}_{b\to x}(0) \\ P^{[\ell+1]}_{>1}(x \to a, \zeta) &= \prod_{b \in N(x,\zeta)\setminus(\{a\}\cup\mathcal{N}_{\leq 1}(x,T[\ell],\zeta))} \mu^{[\ell]}_{b\to x}(0). \end{aligned}$$

We obtain

$$\pi^{[\ell]}_{x\to a}(\zeta) = P^{[\ell]}_{\leq 1}(x \to a, \zeta) \cdot P^{[\ell]}_{>1}(x \to a, \zeta). \tag{22}$$

We show that the first factor representing the product over messages sent by clauses of typical length (regarding the decimation time t) and exposed to at most one additional variable from $T[\ell]$ is close to $\pi[\ell+1]$ simultaneously for $\zeta \in \{-1, 1\}$ for all variables $x \in V \setminus T'[\ell+1]$ and all $a \in N(x)$. Additionally, we prove that the second factor representing the product over messages sent by clauses of atypical length or exposed to at least two additional variables from $T[\ell]$ is close to one simultaneously for $\zeta \in \{-1, 1\}$ for all variables $x \in V \setminus T[\ell+1]$ and all $a \in N(x)$.

Acknowledgements I thank my supervisor Amin Coja-Oghlan for supportive conversations and helpful comments on the final version of this paper.

References

1. D. Achlioptas, P. Beam, M. Molloy Exponential bounds for DPLL below the satisfiability threshold. Proc. SODA (2004) .
2. D. Achlioptas, A. Coja-Oghlan: Algorithmic barriers from phase transitions. Proc. 49th FOCS (2008) 793–802.
3. D. Achlioptas, A. Coja-Oghlan, F. Ricci-Tersenghi: On the solution-space geometry of random constraint satisfaction problems. Random Structures and Algorithms **38** (2011) 251–268.
4. D. Achlioptas, Y. Peres: The threshold for random k-SAT is $2^k \ln 2 - O(k)$. Journal of the AMS **17** (2004) 947–973.
5. D. Achlioptas, G. Sorkin: Optimal myopic algorithms for random 3-SAT. Proc.41st FOCS (2000) 590–600.
6. A. Braunstein, M. Mézard, R. Zecchina: Survey propagation: an algorithm for satisfiability. Random Structures and Algorithms **27** (2005) 201–226.
7. A. Braunstein, R. Mulet, A. Pagnani, M. Weigt, R. Zecchina: Polynomial iterative algorithms for coloring and analysing random graphs. Phys. Rev. E **68** (2003) 036702.
8. A. Braunstein, R. Zecchina: Survey and Belief Propagation on Random k-SAT. Lecture Notes in Computer Science, Springer Berlin (2003).

9 P. Cheeseman, B. Kanefsky, W. Taylor: Where the *really* hard problems are. Proc. IJCAI (1991) 331–337.

10 A. Coja-Oghlan: On belief propagation guided decimation for random k-SAT. Proc. 22nd SODA (2011) 957–966.

11 A. Coja-Oghlan: A better algorithm for random k-SAT. SIAM J. Computing **39** (2010) 2823–2864.

12 A. Coja-Oghlan: The asymptotic k-SAT threshold. Proc. 46th STOC (2014) 804–813.

13 A. Coja-Oghlan, A. Y. Pachon-Pinzon: The decimation process in random k-SAT. SIAM Journal on Discrete Mathematics **26** (2012) 1471–1509.

14 J. Ding, A. Sly, N. Sun: Proof of the satisfiability conjecture for large k. Proc. 47th STOC (2015) 59–68.

15 U. Feige, E. Mossel, D. Vilenchik: Complete convergence of message passing algorithms for some satisfiability problems. Theory of Computing **9** (2013) 617–651.

16 E. Friedgut: Sharp thresholds of graph properties, and the k-SAT problem. J. AMS **12** (1999) 1017–1054.

17 A. Frieze, S. Suen: Analysis of two simple heuristics on a random instance of k-SAT. Journal of Algorithms **20** (1996) 312–355.

18 D. Gamarnik, M. Sudan: Limits of local algorithms over sparse random graphs. Proc. 5th ITCS (2014) 369–376.

19 D. Gamarnik, M. Sudan: Performance of the Survey Propagation-guided decimation algorithm for the random NAE-K-SAT problem. arXiv 1402.0052 (2014).

20 S. Janson, T. Łuczak, A. Ruciński: Random Graphs, Wiley 2000.

21 L. Kroc, A. Sabharwal, B. Selman: Message-passing and local heuristics as decimation strategies for satisfiability. Proc 24th SAC (2009) 1408–1414.

22 F. Krzakala, A. Montanari, F. Ricci-Tersenghi, G. Semerjian, L. Zdeborova: Gibbs states and the set of solutions of random constraint satisfaction problems. Proc. National Academy of Sciences **104** (2007) 10318–10323.

23 R. Marino, G. Parisi, F. Ricci-Tersenghi: The backtracking Survey Propagation algorithm for solving random K-SAT problems. arXiv 1508.05117 (2015).

24 S. Mertens, M. Mézard, R. Zecchina: Threshold values of random K-SAT from the cavity method. Random Struct. Alg. **28** (2006) 340–373.

25 S. Mertens, C. Moore: The Nature of Computation. Oxford University Press 2011.

26 M. Mézard, A. Montanari: Information, physics and computation. Oxford University Press 2009.

27 M. Mézard, G. Parisi, R. Zecchina: Analytic and algorithmic solution of random satisfiability problems. Science **297** (2002) 812–815.

28 D. Mitchell, B. Selman, H. Levesque: Hard and easy distribution of SAT problems. Proc. 10th AAAI (1992) 459–465.

29 M. Molloy: The freezing threshold for k-colourings of a random graph. Proc. 43rd STOC (2012) 921–930.

30 R. Monasson: A Generating Function Method for the Average-Case Analysis of DPLL. Proc. APPROX-RANDOM (2005) 402–413.

31 A. Montanari, F. Ricci-Tersenghi, G. Semerjian: Solving constraint satisfaction problems through Belief Propagation-guided decimation. Proc. 45th Allerton (2007).

32 R. Moser A constructive proof of the Lovász local lemma. Proc. 41st STOC (2009) 343–350.

33 J. Pearl: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

34 F. Ricci-Tersenghi, G. Semerjian: On the cavity method for decimated random constraint satisfaction problems and the analysis of belief propagation guided decimation algorithms. J. Stat. Mech. (2009) P09001.

Approximation Algorithms for Aversion k-Clustering via Local k-Median*

Anupam Gupta[1], Guru Guruganesh[2], and Melanie Schmidt[3]

1 **Computer Science Department, Carnegie Mellon University, Pittsburgh, USA**
2 **Computer Science Department, Carnegie Mellon University, Pittsburgh, USA**
3 **Institute of Computer Science, University of Bonn, Bonn, Germany**

Abstract

In the *aversion k-clustering problem*, given a metric space, we want to cluster the points into k clusters. The cost incurred by each point is the distance to the furthest point in its cluster, and the cost of the clustering is the sum of all these per-point-costs. This problem is motivated by questions in generating automatic abstractions of extensive-form games.

We reduce this problem to a *"local" k-median problem* where each facility has a prescribed radius and can only connect to clients within that radius. Our main results is a constant-factor approximation algorithm for the aversion k-clustering problem via the local k-median problem. We use a primal-dual approach; our technical contribution is a non-local rounding step which we feel is of broader interest.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Approximation algorithms, clustering, k-median, primal-dual

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.66

1 Introduction

In this paper, we consider the following clustering problem: given a metric space (X, D) with a set of clients, and a parameter k, the goal is to group the clients into k clusters. For each client j, the cost incurred by j is its distance to the furthest client in its cluster. The cost of the clustering is the sum over all clients, more precisely, of the cost incurred by the client. We call this problem the *aversion k-clustering problem*.

This question is motivated by a problem in developing abstractions of extensive-form games. Since finding equilibria in large extensive form games is computationally expensive, one appealing approach if speeding things up is to develop an abstraction of this game. Since the abstraction is typically much smaller, existing algorithms can be used to solve them to find optimal strategies, which can be mapped back to the original game. However, there is often some loss in going to the abstraction. Recent work of Kroer and Sandholm [19] on automated abstraction algorithms proposed a following way to model this: since several states of the original game may be collapsed into a single state in the abstraction, the loss for each original state is its distance (in a suitably defined metric) to the furthest state that is collapsed with it. The overall loss is the sum of per-state losses. This is precisely the aversion k-clustering problem we study in this paper.

* This work was done in part when M. Schmidt was visiting the Computer Science Department at Carnegie Mellon University, supported by the German Academic Exchange Service (DAAD). A. Gupta and G. Guruganesh were supported in part by NSF awards CCF-1319811 and CCF-1536002

To the best of our knowledge, no prior approximation algorithms were known for the aversion k-clustering problem. Although it is related in spirit to several other clustering problems, it has some interesting and unique features. Indeed, something that makes this problem difficult is its high "sensitivity". To explain this, observe that in problems like k-median, if we re-assign a single client j to a new cluster C, loosely this changes the cost by the distance of the client to the new cluster. However, in aversion k-clustering, reassigning client j to a new cluster C may also significantly change the cost of all other clients in C, since j may become their new furthest client. This creates problems for most standard techniques used for facilty-location problems. Another facility-location problem with a similar high-sensitivity property is the *min-sum clustering* problem, for which only logarithmic approximations are known via HST embeddings and a non-trivial dynamic program [4, 5]. Since our objective is not linear (due to each client paying the distance to its furthest cluster-mate), we cannot even use tree embeddings.

The main result of the paper is the following:

▶ **Theorem 1.** *There is a constant-factor algorithm for the aversion k-clustering problem.*

1.1 Our Techniques

A few words about our techniques. To solve aversion k-clustering, we first move to a related problem that is more convenient to deal with: in the *local k-median problem*, each potential facility location in the metric space has a "range" R_i associated with it. Like in k-median, we need to open k facilities, to minimize the sum of distances from clients to their assigned facilities. However, we now additionally require that each client j is assigned to some facility i at distance at most R_i. This problem is NP-hard to approximate, but for our purpose it is sufficient to solve the relaxed version where clients can still connect at distance $O(\mathsf{R}_i)$.

The (relaxed) locality restriction causes many of the standard techniques for k-median, like local-search and LP-rounding, not to extend to this problem. (In fact, we do not know of a constact factor approximation algorithm for local k-median which only violates locaility constraints by a constant factor). However, we are successful in extending a primal-dual technique to the instances which arise from the aversion k-clustering problem. The following theorem is our main technical result, from which Theorem 1 follows immediately.

▶ **Theorem 2.** *There is an approximation algorithm for the local k-median problem that achieves a constant-factor approximation for instances arising from the aversion k-clustering problem. Its solutions violate the locality constraints by a constant factor.*

We use the primal-dual framework of Jain and Vazirani: we find two solutions that open k_1 and k_2 facilities (such that $k_1 < k < k_2$) such that the "average" of these two solutions has low cost and opens k facilities. This part of the analysis is well-understood by now and we omit details because of space limitations. We can view this average solution as a "well-behaved" LP solution, which we now have to round to integrally open k facilities.

The main problem with this rounding is the locality constraint — typical algorithms tend to round some fractional facility up to 1, round down close-by fractional facilities to zero to maintain the total facility mass at k, and reroute clients to the newly opened facility without increasing the cost by much. However, the locality constraint in our problem means that such simple rounding approaches fail. For example, the facility that we open may have a very small R_i value, and can only serve clients that are very close to it. However, the clients who want to be rerouted may be too far from this facility to satisfy the locality constraint, even if it is relaxed to $\gamma\mathsf{R}_i$ for some constant γ.

Our main technical contribution, and the novel ingredient of our rounding algorithm is a *non-local rounding approach*. We first transform the fractional solution so that its support is a forest. Then we partition this forest into carefully chosen subtrees, so that all the clients in each particular subtree can be reassigned simultaneously without violating the locality. Now choosing the least expensive of these subtrees to reassign gives us a solution with k facilities and a constant-factor approximation for instances arising from the aversion k-clustering problem. We feel that this non-local rounding will be useful in other contexts, and hence be of independent interest.

Related work. Approximation algorithms for facility location problems have been studied for a long time. Indeed, many approximation techniques have been developed while investigating these problems (see [22]). The problem closest to the *local k-median problem* is naturally the *metric k-median* problem. The first constant-factor for this problem was due to Charikar et al. [8] via rounding the LP; subsequently, primal-dual algorithms were given by Jain and Vazirani [17] and Charikar and Guha [7], a local-search algorithm was given by Arya et al. [2]. The recent approach of Li and Svensson [21] gave a $2.73+\varepsilon$-approximation, which was improved to $2.675+\varepsilon$ by Byrka et al. [6]. The current NP-hardness is a $1+2/e$-factor due to Jain, Mahdian, Saberi [16]. The related problem of *uncapacitated metric facility location* has constant-factor approximations via most approximation techniques: see, e.g., the book of [22]. The current best approximation factor is 1.488 due to Li [20], and the hardness is an 1.463-factor due to Guha and Khuller [13].

The k-median problem sums over each cluster, the sum of distances of clients to their cluster center. Instead of taking the sum of distances within each cluster, we could take the maximum distance within each cluster; this gives the *sum of cluster diameters* problem, for which a $O(1)$-factor is due to Charikar and Panigrahy [9]. And instead of summing diameters over the clusters, if we take the maximum diameter over all clusters, we get the *k-center* problem, for which a 2-approximation is due to Gonzalez [12], and Hochbaum and Shmoys [14], and a matching hardness is due to Hsu and Nemhauser [15].

Another related problem is the *min-sum clustering problem*, where we sum over the clusters of the distances between all pairs within the cluster. Bartal et al. [4] give a $O(\varepsilon^{-1}\log^{1+\varepsilon} n)$-approximation, which was recently improved to $O(\log n)$ by Behsaz [5]. There are easy examples where these problems differ from aversion k-clustering by arbitrarily large factors. Moreover, the non-linearity of our objective function means that we cannot use tree embedding results to even get a logarithmic approximation.

Our algorithm takes a primal-dual approach pioneered by Jain and Vazirani [17]; while solving the Lagrangian relaxation and getting a Lagrangian-multiplier preserving algorithm follows relatively easily, the main contribution is in the non-local rounding algorithm. This adds to the body of work exploring such primal-dual techniques, which include the work of Charikar and Panigrahy [9] to give a $O(1)$-factor approximation for the sum of cluster diameters, and Chuzhoy and Rabani [10] in their $O(1)$-factor bicriteria approximation for the capacitated k-median problem. Non-local roundings of a different flavor were also recently used for the capacitated k-center problem by Cygan et al. [11] and An et al. [1]. To the best of our understanding, our rounding technique is different from these previous works.

2 Preliminaries

Let (X, D) be a metric space and let $C \subseteq X$ be the set of *clients* and $F \subseteq X$ be the set of *facilities*. The *aversion k-clustering problem* is the task to partition C into a collection $\mathcal{C}$ of

k disjoint subsets $C_1, \dots, C_k$ with $C = \cup_{i=1}^{k} C_i$ such that

$$c_a(\mathcal{C}) := \sum_{\ell=1}^{k} \sum_{j_1 \in C_\ell} \max_{j_2 \in C_\ell} D(j_1, j_2) \tag{P1}$$

is minimized.

For the *local k-median problem*, we additionally get a *radius* (or *range*) R_i for every $i \in F$. We seek a set $\mathcal{F} \subseteq F$ of k facilities that minimizes

$$c_l(\mathcal{F}) := \sum_{j \in C} \min_{\substack{i \in \mathcal{F}, \\ D(i,j) \leq \mathsf{R}_i}} D(i, j).$$

This differs from the classical k-median problem in that a client can only be assigned to a facility if it lies within the facility's radius. It is possible that there is no set of k facilities which can service all clients. If this is the case, we define the minimum clustering cost as infinity. In the following claim, we show that it is NP-hard to decide whether we are in this case or not.

▶ **Claim 1.** Deciding feasibility of a local k-median instance is NP-hard.

Proof. We use a well-known reduction from set cover. Let $\mathcal{S}$ be a set of sets over a universe $\mathcal{U}$. We construct a metric space that contains a facility i_S for every set $S \in \mathcal{S}$ and a client j_u for every element $u \in \mathcal{U}$. The distance between j_u and i_S is one if $u \in S$ and two otherwise. Observe that this is a metric. We set the radius R_{i_S} of all facilities to one. Observe that there is a feasible solution for this local k-median instance if and only if the set cover instance has a solution with at most k sets. Since deciding whether a set cover instance has a solution with at most k sets is NP-hard [18], it is also NP-hard to decide whether there is a feasible solution for the local k-median problem. ◀

Any approximation algorithm has to decide whether there is a feasible solution or not. Hence, we allow the locality constraint to be violated; i.e. a client may connect to a facility i if it is within a radius of $\gamma \mathsf{R}_i$ for a constant γ. We say a solution is a (γ, ψ) bicriteria solution if the solution violates the locality constraints by a factor of γ and has cost at most ψ times the optimal (with respect to the original problem).

3 Solving the aversion k-clustering problem via the local k-median problem

We show that the aversion k-clustering problem can be reduced to the local k-median problem by sacrificing a constant factor. The idea is to identify a cluster C_ℓ with a pair of points with largest distance and to use this information to represent clusters by an artificial facility with appropriate radius. More precisely, we define the following metric space. Set $F := \{p_{j_1 j_2} \mid j_1, j_2 \in C\}$ and refer to $p_{j_1 j_2}$ as the *midpoint* of clients j_1 and j_2. To extend D from C to $C \cup F$, we set

$$D(j_1, p_{j_1 j_2}) := D(j_2, p_{j_1 j_2}) = \frac{D(j_1, j_2)}{2} \text{ and } D(p_{j_1 j_2}, p_{j_1 j_2}) := 0$$

for all $j_1, j_2 \in C$. So far, no metric property is violated. Now imagine the incompletely defined metric as a weighted undirected graph G on the vertices $C \cup F$ where some edges are missing. Let D be defined as the shortest path metric in G. This is a metric by definition. It

coincides with the previously defined distances since in a metric, the direct edge must be a shortest path. For the missing edges, we get that

$$D(j, p_{j_1 j_2}) = D(p_{j_1 j_2}, j) = \min\{D(j, j_1), D(j, j_2)\} + \frac{D(j_1, j_2)}{2} \tag{1}$$

for all $j \in C$: The point $p_{j_1 j_2}$ is only connected to j_1 and j_2, thus any path between j and $p_{j_1 j_2}$ has to travel over one of them. Since the edge lengths form a metric, the direct connection between j and j_1 or j_2 is shortest, so either $(j, j_1), (j_1, p_{j_1 j_2})$ or $(j, j_2), (j_2, p_{j_1 j_2})$ is a shortest path. Analogously, we get that

$$D(p_{j_1 j_2}, p_{j_3 j_4}) = D(j_1, j_2)/2 + D(j_3, j_4)/2 + \min\{D(j_1, j_3), D(j_1, j_4), D(j_2, j_3), D(j_2, j_4)\}$$

for all $j_1, j_2, j_3, j_4 \in C$. Finally, we define

$$\mathsf{R}_{p_{j_1 j_2}} := D(j_1, j_2)/2 \tag{2}$$

for all $p_{j_1 j_2} \in F$. Notice that our definition of F allows that $j_1 = j_2$. This ensures that singleton clusters can be expressed. Furthermore, notice that $\mathsf{R}_{p_{j_1 j_1}} = 0$, so the facility $p_{j_1 j_1}$ can only serve j_1 (or clients at the same location).

For any facility p_{j_1, j_2}, we will drop the reference to j_1 and j_2 when it is clear from the context. Hence, $p \in F$ refers to the midpoint of some two clients j_1 and j_2 and the radius of the facility R_p simply refers to half the distance between these points. Intuitively, each new "facility" corresponds to a midpoint of two clients in the original problem. These midpoints allow us to cast the current problem as a k-median problem with the addition of locality constraints placed on each facility.

We now show how solutions for the aversion k-clustering problem and (γ, α) bicriteria solutions for the local k-median problem are related. For a client j, let $F_j^\gamma := \{i \in F \mid D(i, j) \leq \gamma \mathsf{R}_i\}$ be the set of facilities that j is allowed to connect to. The following integer linear program (ILP) which is a (natural) modification of the ILP proposed in [3] minimizes over all feasible (γ, α) bicriteria solutions.

$$\begin{aligned}
\min \sum_{j \in C} \sum_{i \in F_j^\gamma} D(i, j) \cdot x_{i,j} & & & \text{(ILP}^\gamma\text{)} \\
\sum_{i \in F} y_i &\leq k & & \\
\sum_{i \in F_j^\gamma} x_{i,j} &\geq 1 & \forall j \in C & \\
y_i - x_{i,j} &\geq 0 & \forall j \in C, i \in F_j^\gamma & \\
x_{i,j}, y_i &\in \{0, 1\} & \forall j \in C, i \in F_j^\gamma &
\end{aligned}$$

ILP$^\gamma$ has a variable y_i for each $i \in F$ that indicates whether the 'facility' i is opened, and a variable $x_{i,j}$ for any combination of an original point j and a facility $i \in F_j^\gamma$ that says whether j is connected to i.

Let (x, y) be any solution of ILP$^\gamma$ and let $c(x, y)$ be the cost of the solution. We relate the solutions of ILP$^\gamma$ to the problem (P1) by the following lemmas.

▶ **Lemma 3.** *Given a solution (x, y) of ILP$^\gamma$, there exists a solution $\mathcal{C} = \{C_l\}_{l=1}^k$ to* (P1) *which has cost no more than $c_a(\mathcal{C}) \leq (\gamma + 1)c(x, y)$.*

Proof. Since (x, y) is an integral solution, let $\{p_1, \dots, p_k\} \subseteq F$ denote the facilities which are opened. We define the cluster C_i to be the set of clients j such that $x_{p_i,j} = 1$. For any client $j \in C_i$, let $j' \in C_i$ be the client which maximizes $D(j, j')$. Since D is a metric, we know that $D(j, j') \leq D(p_i, j) + D(p_i, j')$. By the locality constraint, it holds that $D(p_i, j') \leq \gamma \mathsf{R}_{p_i}$. By definition of D and R_{p_i}, we know $D(p_i, j) \geq \mathsf{R}_{p_i}$, which implies $D(p_i, j') \leq \gamma D(p_i, j)$. Hence, $D(j, j') \leq (\gamma + 1)D(p_i, j)$. Summing this over all clients, we conclude that $c_a(\mathcal{C}) \leq (\gamma + 1)c(x, y)$. ◀

▶ **Lemma 4.** *Given a solution $\mathcal{C}$ to* (P1)*, we can construct a solution (x, y) to ILP$^\gamma$ (where $\gamma \geq 3$) which has cost $\frac{1}{2}c_a(\mathcal{C}) \leq c(x, y) \leq 2c_a(\mathcal{C})$.*

Proof. Fix a cluster C_i, let $j_1, j_2 \in C_i$ be two clients which maximize $D(j_1, j_2)$. Open facility $p_{j_1 j_2}$ and connect all clients in C_i to it. Notice that it holds $D(j, p_{j_1 j_2}) = \mathsf{R}_{p_{j_1 j_2}} + \min\{D(j, j_1), D(j, j_2)\} \leq 3\mathsf{R}_{p_{j_1 j_2}}$ because $D(j_1, j_2)$ is the maximum distance between two clients in C_i and because $D(j_1, j_2) = 2\mathsf{R}_{p_{j_1 j_2}}$. Thus, the solution is feasible for $\gamma = 3$.

For any client $j \in C_i$, let $j' \in C_i$ be the element which maximizes $D(j, j')$. For the first inequality notice that each client j will pay at least $D(j, p_{j_1 j_2}) \geq \frac{1}{2}D(j_1, j_2) \geq \frac{1}{2}D(j, j')$. Observe that $D(j, j') \geq \max\{D(j, j_1), D(j, j_2)\} \geq (D(j, j_1) + D(j, j_2))/2 \geq D(j_1, j_2)/2$. Combining this with the observation that $D(j, j') \geq \min\{D(j, j_1), D(j, j_2)\}$, we get that

$$D(p_{j_1 j_2}, j) = \min\{D(j, j_1), D(j, j_2)\} + D(j_1, j_2)/2 \leq 2 \cdot D(j, j').$$

Summing over all clients gives the second inequality. ◀

Let opt^γ_{ilp} be the optimal value for ILP$^\gamma$ and let opt_a be the value of an optimal solution for (P1). Lemma 4 implies that $opt^3_{ilp} \leq 2 \cdot opt_a$. Assuming we compute an ψ-approximate solution to the optimal ILP^3 solution that violates the locality constraint by an additional factor of ϱ. Then this solution costs at most $\psi \cdot opt^3_{ilp} \leq 2\psi \cdot opt_a$ and violates the locality constraints by 3ϱ. By Lemma 3, we can then construct a feasible solution for (P1) that costs at most $(3\varrho + 1) \cdot 2\psi \cdot opt_a$.

3.1 Good fractional solutions for the local k-median problem

Since problem ILP$^\gamma$ is NP-hard, we relax the integrality constraints to obtain a linear program. The only difference between the standard k-median relaxation and LP$^\gamma_{\mathcal{P}}$ is the locality constraint, i.e., each client j can only connect to facilities in F^γ_j.

$$\begin{aligned}
\min \quad & \sum_{i,j} D(i,j)x_{i,j} && (\mathrm{LP}^\gamma_{\mathcal{P}}) \\
\text{s.t.} \quad & \sum_{i \in F^\gamma_j} x_{i,j} \geq 1 && \forall j \in C \\
& y_i - x_{i,j} \geq 0 && \forall j \in C, i \in F^\gamma_j \\
& \sum_{i \in F} -y_i \geq -k \\
& x, y \geq 0.
\end{aligned}$$

$$\begin{aligned}
\max \quad & \sum_j \alpha_j - kZ && (\mathrm{LP}^\gamma_{\mathcal{D}}) \\
\text{s.t.} \quad & \\
& \alpha_j \leq D(i,j) + \beta_{i,j} && \forall j \in C, i \in F^\gamma_j \\
& \sum_{j : i \in F_j} \beta_{i,j} \leq Z && \forall i \in F \\
& \alpha, \beta, Z \geq 0.
\end{aligned}$$

The above LP is very similar to the LP for facility location and this fact was exploited by Jain and Vazirani to show that primal-dual solutions to the facility location problem can be transformed into the solutions for the k-median problem. Let LP-F$^\gamma_{\mathcal{P}}$ be the facility location variant of LP$^\gamma_{\mathcal{P}}$, and let LP-F$^\gamma_{\mathcal{D}}$ be its dual:

$$\min \sum_{i\in F, j\in C} D(i,j)x_{i,j} + \sum_{i\in F} f_i y_i \qquad \text{(LP-F}^{\gamma}_{\mathcal{P}}\text{)}$$
$$\text{s.t.} \sum_{i\in F^{\gamma}_j} x_{i,j} \geq 1 \qquad \forall j \in C$$
$$y_i - x_{i,j} \geq 0 \qquad \forall j \in C, i \in F^{\gamma}_j$$
$$x, y \geq 0.$$

$$\max \sum_j \alpha_j \qquad \text{(LP-F}^{\gamma}_{\mathcal{D}}\text{)}$$
$$\text{s.t.}$$
$$\alpha_j \leq D(i,j) + \beta_{i,j} \;\forall j \in C, i \in F^{\gamma}_j$$
$$\sum_{j: i\in F^{\gamma}_j} \beta_{i,j} \leq f_i \qquad \forall i \in F$$
$$\alpha, \beta \geq 0.$$

Augmenting ideas introduced by Jain and Vazirani [17], we obtain integer solutions to LP-F$^{\gamma}_{\mathcal{P}}$. This produces two solutions (x^1, y^1) and (x^2, y^2) that are nearly feasible for LP$^{3\gamma}_{\mathcal{P}}$, but $\sum_i y^1_i = k_1 < k$ and $\sum_i y^2_i = k_2 > k$. A suitable convex combination of the two is a feasible solution for LP$^{3\gamma}_{\mathcal{P}}$ and is a constant factor away from the optimal value of LP$^{\gamma}_{\mathcal{P}}$.

▶ **Lemma 5.** *Given any $\epsilon > 0$ and $\gamma > 0$, there exists a polynomial time algorithm which finds two feasible integer solutions $(x^1, y^1), (x^2, y^2)$ for LP-F$^{3\gamma}_{\mathcal{P}}$ with the following properties:*

1. *$\sum_i y^1_i = k_1$ and $\sum_i y^2_i = k_2$ for two integers $k_1 < k < k_2$.*
2. *Set $\rho = \frac{k_2 - k}{k_2 - k_1}$. The solution $(\hat{x}, \hat{y}) = \rho(x^1, y^1) + (1-\rho)(x^2, y^2)$ is feasible for LP$^{3\gamma}_{\mathcal{P}}$ with cost at most $(3+\epsilon)$ times the optimal solution to LP$^{\gamma}_{\mathcal{P}}$.*

Since the essential ideas behind this lemma use standard techniques, we omit the full proof because of space limitations. The main differences to the standard Jain-Vazirani primal-dual process are as follows: When finding the initial set of open facilities, we restrict clients to paying and connecting only to facilities whose radius they lie in. In the clean-up step, the Jain-Vazirani algorithm selects the finally open facilities by finding an arbitrary independent set of facilities in some graph. We use the freedom to choose any independent set and choose a set that ensures that clients that have to be reassigned (because their original facility was closed) can always be routed to an open facility with higher radius than their original facility.

3.2 Rounding

Given any two integer solutions (x_1, y_1) and (x_2, y_2) for LP-F$^{\gamma}_{\mathcal{P}}$, which open $A, B \subseteq F$ facilities, respectively, we define a weighted bipartite graph $G(x_1, y_1, x_2, y_2)$ as follows. The graph is defined on the vertex set with bipartitions A and B. We connect $i \in A$ to $i' \in B$ if there exists a client j such that $x^1_{i,j} = 1$ and $x^2_{i',j} = 1$. The weight of an edge (i, i') is the number of clients j which satisfy the above requirement.

▶ **Lemma 6.** *The following holds for local k-median instances that arise from the aversion k-clustering problem. Given two integer solutions $(x^1, y^1), (x^2, y^2)$ for LP-F$^{\gamma}_{\mathcal{P}}$ which open facilities $A, B \subseteq F$, respectively, we can construct solutions $(\tilde{x}^1, \tilde{y}^1), (\tilde{x}^2, \tilde{y}^2)$ that satisfy:*

1. *$(\tilde{x}^1, \tilde{y}^1)$ opens facilities A and $(\tilde{x}^2, \tilde{y}^2)$ opens facilities B.*
2. *If $(x_1, y_1), (x_2, y_2)$ are feasible for LP-F$^{\gamma}_{\mathcal{P}}$, then $(\tilde{x}^1, \tilde{y}^1), (\tilde{x}^2, \tilde{y}^2)$ are feasible solutions to LP-F$^{3\gamma}_{\mathcal{P}}$, and they satisfy $c(\tilde{x}^1, \tilde{y}^1) \leq 3\gamma c(x^1, y^1)$ and $c(\tilde{x}^2, \tilde{y}^2) \leq 3\gamma c(x^2, y^2)$.*
3. *The graph $G(\tilde{x}^1, \tilde{y}^1, \tilde{x}^2, \tilde{y}^2)$ is a forest.*

Proof. We will assume that all radii are distinct (we can ensure this, e.g., by adding a tiny amount of noise to all the radii, or by breaking ties consistently). We say that

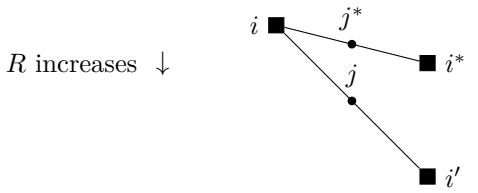


Figure 1 Removing all but one down edge for client i.

an edge $\{i, i'\}$ in $G(x^1, y^1, x^2, y^2)$ is a *down edge for* i if $\mathsf{R}_{i'} > \mathsf{R}_i$. For $i \in A \cup B$, let $\mathfrak{D}(i) := \{i' \mid \{i, i'\}$ is a down edge$\}$ be the set of facilities that are connected to i by down edges. Furthermore, for every $i \in A \cup B$, let i^* be a facility that minimizes $\{\mathsf{R}_i \mid i \in \mathfrak{D}(i)\}$, i.e., i^* is the endpoint of a 'highest' down edge. For each $i \in A \cup B$, we modify assignments as follows. For all clients $j \in C$ connected to i, and to some facility $i' \in \mathfrak{D}(i)$ in (x_1, y_1), (x_2, y_2), we reassign them to now connect to i and i^* in $(\tilde{x}^1, \tilde{y}^1), (\tilde{x}^2, \tilde{y}^2)$. Thus, for all clients $j \in C$ originally connected to i and i^*, the assignment does not change.

Let us calculate the costs of the resulting assignment. Let $i' \in \mathfrak{D}(i)$ be a facility with $i' \neq i^*$ and let j be a client that is reconnected from i' to i^*. Notice that $D(i, i^*) \le \gamma\mathsf{R}_i + \gamma\mathsf{R}_{i^*}$ since at least one client lies within the (γ-expanded) radius of i and i^* simultaneously. We observe that $D(j, i^*) \le D(j, i) + D(i, i^*) \le \gamma(\mathsf{R}_i + \mathsf{R}_i + \mathsf{R}_{i^*}) \le 3\gamma \cdot \mathsf{R}_{i^*}$ by the triangle inequality and by $\mathsf{R}_{i^*} \ge \mathsf{R}_i$. Thus, the new solution violates the locality constraint for j by a factor of at most 3. Since R_{i^*} is the smallest radius for all facilities in $\mathfrak{D}(i)$, it holds that $\mathsf{R}_{i^*} \le \mathsf{R}_{i'}$. Thus, we also have $D(j, i^*) \le 3\gamma\mathsf{R}_{i'}$. Moreover, since the instances arise from local k-median, equations (1) and (2) imply that $D(j, i') \ge \mathsf{R}_{i'}$. (This is the only part of the proof that relies on the local k-median instances arising from aversion k-clustering). Hence we have $D(j, i^*) \le 3\gamma\mathsf{R}_{i'} \le 3\gamma D(j, i')$. Thus the cost of each client j is increased by a factor of at most 3γ, which immediately proves Property 2. (Figure 1 visualizes this calculation).

We do not open or close any facilities, thus Property 1 is true. To see Property 3 holds, note that by the distinct radii assumption, any cycle would contain a facility with two down edges, which is no longer possible after the reassignment. ◀

Lemma 6 transforms our solution such that it corresponds to a forest T on the vertices $A \cup B$. We first assume that T is a tree and later deal with each connected component separately. We use the tree structure to define a depending rounding procedure to combine A and B into an integral solution C with low cost. It will be crucial to look at the difference between the number of vertices from B and A in subtrees of T.

▶ **Definition 7.** For any subtree of $T' \subseteq T$, we define the *deficiency* of T' to be $\mathsf{df}(T') = |B(T')| - |A(T')|$ where $B(T')$ (and $A(T')$) are the vertices from B (and A) in this subtree.

We start with $C = B$. Then we find a subtree T' with $\mathsf{df}(T') = 1$, i.e., one node more from B than from A. We want to close all facilities in $B(T')$, open all facilities in $A(T')$ and reconnect the affected clients. We want that the reassignment follows the assignments in $(\tilde{x}^1, \tilde{x}^2)$, so all facilities in A that are adjacent to $B(T')$ must be contained in $A(T')$. We then iterate this process with more subtrees until C has exactly k vertices. Since we gain one for every subtree, we need $t := k_2 - k$ subtrees until $|C|$ was reduced from k_2 to k. The subtrees must be disjoint on the B side, while the vertices from A can overlap. The following lemma shows that we can find a large set of suitable subtrees from which we can choose the cheapest t later. Figure 2 visualizes two examples.

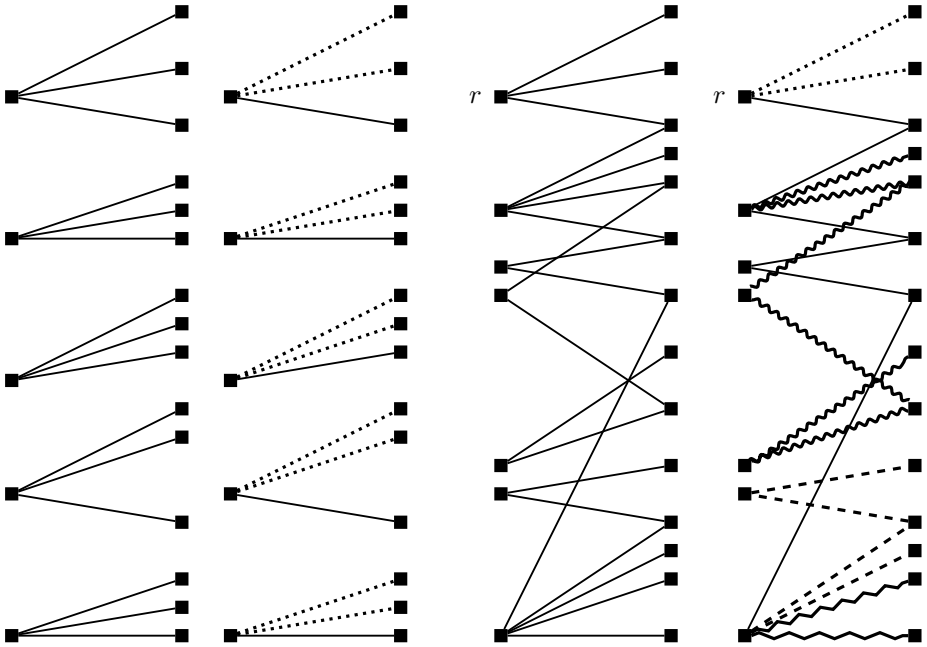


Figure 2 Examples on finding subtrees of $G(x^1, y^1, x^2, y^2)$ with $\mathsf{df}(T_i) = 1$. The two left pictures show a simple special case that also is a worst case for the number of subtrees: The deficiency of the shown forest F is $\mathsf{df}(F) = 2k = 10$ and we get $\mathsf{df}(F)/2 = 5$ subtrees by pairing the nodes from B. The two right pictures show a connected tree T with $\mathsf{df}(T) = 7$ and 4 subtrees with $\mathsf{df}(T_i) = 1$.

▶ **Lemma 8.** *Given any tree T with $\lceil \mathsf{df}(T)/2 \rceil = l$ and root $r \in A$, we can find l subtrees $T_1, \ldots, T_l$ of T with*

1. $\mathsf{df}(T_i) = 1$
2. $B(T_i) \cap B(T_j) = \emptyset$
3. $A(\delta(B(T_i))) \subseteq A(T_i)$

where we use $\delta(X)$ to denote the set of edges from X to $\bar{X}$.

Proof. Let r be the root of T and $c_1, \ldots, c_\nu$ be the children of r. Our proof will proceed with induction on the height of T. By *removing* a subtree T' we mean that we remove all edges that are in T' from T and all vertices except the root of T'.

Induction Hypothesis: There exist subtrees $T_1, \ldots, T_z$, $z \in \mathbb{N}_0$, of T that satisfy:

1. Each subtree T_i is rooted at a vertex in A and satisfies that $\mathsf{df}(T_i) = 1$.
2. After removing $T_1, \ldots, T_z$ from T, the following holds. If $r \in B$ then $\mathsf{df}(T) \leq 1$. If $r \in A$ then $\mathsf{df}(T) \leq 0$.

Base Case: T has height 0 or 1, i.e., it is a star. If $r \in B$, then $\mathsf{df}(T) \leq 1$ because there is only one node from B. If $r \in A$, then we can remove the children in pairs until there are no pairs left. This is because the subtree consisting of r and any two of its children has deficiency 1. Therefore, each pair and the root will correspond to a subtree (one of T_i mentioned in the IH) that we remove. After removing them, T consists of only r or r and one node from B. In both cases, $\mathsf{df}(T) \leq 0$.

Induction Step:

Case $r \in B$: By the induction hypothesis (IH), we can remove some subtrees to ensure that each subtree rooted at $c_1 \ldots c_\nu$ will have deficiency at most 0. Since $\mathsf{df}(T) = \sum_{i=1}^{\nu} \mathsf{df}(T_{\text{rooted at } c_i}) + 1 \leq 1$, we can conclude that this satisfies the first property in the

IH. Since we didn't remove any additional subtrees, the second property is vacuously satisfied.

Case $r \in A$: By the IH, we know that the subtree rooted at each child c_i has deficiency $\mathsf{df}(c_i) \leq 1$. Without loss of generality, let $c_1, \ldots, c_p$ be the children which have deficiency 1 and $c_{p+1} \ldots c_\nu$ have deficiency ≤ 0. If $p \leq 1$, then $\mathsf{df}(T) \leq 0$. If $p \geq 2$, then we remove pairs of children with positive deficiency. Observe that the subtree rooted at r containing only the children c_1 and c_2 has deficiency exactly 1. Hence, these satisfy the second property in the IH. We continue this process until there is at most 1 child which has positive deficiency, at which point the the first property is satisfied. This ensures that the induction step is satisfied.

Notice that each removed subtree has deficiency one. Since we keep the root, the deficiency decreases by two for each removed subtree. When $r \in A$ as assumed in the lemma, then $\mathsf{df}(T)$ is decreased to at most zero. Thus, at least $\lceil \mathsf{df}(T)/2 \rceil$ subtrees are removed. ◀

For a forest F consisting of trees $F_1, \ldots, F_x$, set $\mathsf{df}(F) := \sum_{j=1}^{x} \mathsf{df}(F_j)$. We can find $\lceil \mathsf{df}(F_j)/2 \rceil$ subtrees satisfying the properties of Lemma 8 for every every F_j. Thus, we get

$$\sum_{j=1}^{x} \left\lceil \frac{\mathsf{df}(F_i)}{2} \right\rceil \geq \left\lceil \frac{1}{2} \sum_{j=1}^{x} \mathsf{df}(F_i) \right\rceil = \lceil \mathsf{df}(F)/2 \rceil$$

subtrees, giving the following corollary.

▶ **Corollary 9.** *Given any forest F with $\lceil \mathsf{df}(F)/2 \rceil = l$, we can find l subtrees $T_1, \ldots, T_l$ of F satisfying the properties from Lemma 8.*

We now show Theorem 2. We are given an instance of the local k-median problem that arises from the aversion k-clustering problem. We know that the solutions for the local k-median problem that are induced by the aversion k-clustering instance are feasible for LP-F$^3_{\mathcal{P}}$. Thus, we set $\gamma := 3$. Then we use Lemma 5 and Lemma 6 to get two solutions (x^1, y^1) and (x^2, y^2) so that the graph $G(x^1, y^1, x^2, y^2)$ is a forest, (x^1, y^1) opens k_1 facilities and (x^2, y^2) opens $k_2 \geq k_1$ facilities. Both Lemma 5 and Lemma 6 induce a factor of 3 in the radius violation, so (x^1, y^1) and (x^2, y^2) are feasible for LP-F$^{9\gamma}_{\mathcal{P}}$. Furthermore, the intermediate solutions $(\hat{x}^1, \hat{y}^1)$ and $(\hat{x}^2, \hat{y}^2)$ coming from Lemma 5 have the property that for $\rho = (k_2 - k)/(k_2 - k_1)$, it holds that $\rho \cdot c(\hat{x}^1, \hat{y}^1) + (1 - \rho) \cdot (\hat{x}^2, \hat{y}^2) \leq (3 + \varepsilon) \cdot opt^\gamma_l$. Applying Lemma 6 increases the cost bound by a factor of 3γ. Thus, we know that

$$\rho \cdot c(x^1, y^1) + (1 - \rho) \cdot c(x^2, y^2) \leq (3 + \varepsilon) \cdot 3\gamma \cdot opt^\gamma_l := c^{mix}$$

for $\rho = (k_2 - k)/(k_2 - k_1)$. If (x^1, y^1) or (x^2, y^2) opens exactly k facilities, we are done. Otherwise, $k_1 < k < k_2$. If $\rho \geq 1/2$, simply output (x^1, y^1) which then costs $c(x^1, y^1) \leq 2\rho \cdot c(x^1, y^1) \leq 2c^{mix}$. We assume that this is not the case, i.e., $\rho < 1/2$.

We build a solution C and start with $C = B$. Using Corollary 9, we find $\frac{1}{2}(k_2 - k_1)$ subtrees $T_1, \ldots, T_\ell$ of $G(x^1, y^1, x^2, y^2)$. For each subtree T_i, we can reassign the clients from the facilities in $B(T_i)$ to the facilities in $A(T_i)$. We denote the connection cost for assigning the clients to $A(T_i)$ by $c(T_i)$. Notice that $c(x^1, y^1) \geq \sum_{s=1}^{\ell} c(T_i)$ because every edge of T can only appear in one subtree (since the $B(T_i)$ are pairwise disjoint). Thus, if we choose the $t = k_2 - k$ subtrees $T_{i_1}, \ldots, T_{i_t}$ with the cheapest $c(T_i)$, then

$$\sum_{z=1}^{t} c(T_{i_z}) \leq \frac{t}{\ell} \sum_{s=1}^{\ell} c(T_i) \leq \frac{t}{\ell} c(x^1, y^1) = \frac{k_2 - k}{\frac{1}{2}(k_2 - k_1)} c(x^1, y^1) = 2\rho \cdot c(x^1, y^1).$$

The cost of C starts at $c(x^2, y^2)$ and is increased by at most $2\rho \cdot c(x^1, y^1)$. Thus, the solution costs at most $2\rho \cdot c(x^2, y^2) + c(x^2, y^2) \leq 2\rho \cdot c(x^2, y^2) + 2(1-\rho) \cdot c(x^2, y^2) \leq 2 \cdot c^{mix}$ where we recall that $\rho < 1/2$. Thus, we get an integer solution of cost $2 \cdot (3+\varepsilon) \cdot 3\gamma \cdot opt_l^\gamma$ that is feasible for $\mathrm{LP}_{\mathcal{P}}^{9\gamma}$. That induces a solution for the aversion k-clustering instance that is a constant factor approximation as we described below Lemma 4.

3.3 Improving the Approximation Factor

To improve the final approximation ratio for the aversion k-clustering problem, we observe that the dual variables computed by the primal-dual algorithm can be directly related to the objective of the aversion k-clustering problem. We split each such dual: Let $\alpha_O(j)$ denote the amount that client j pays to open a facility (the subscript O stands for "open"). Using the terminology of Jain and Vazirani, we say a client j is directly connected to facility i if $\beta_{i,j} > 0$ and facility i is open. In this case, $\alpha_O(j) := \beta_{i,j}$. Otherwise, $\alpha_O(j) = 0$. Define $\alpha_C(j) := \alpha(j) - \alpha_O(j)$ (intuitively, this is the connection cost—the subscript C is for connection—that the client has paid for, but for indirectly connected clients we only know that $D(i,j) \geq \alpha_C(j) \geq (1/3)D(i,j)$ is true. For directly connected clients, $\alpha_C(j) = D(i,j)$.).

▶ **Lemma 10.** *At the end of the primal-dual algorithm, if client j connects to facility i, then $\alpha_C(j) \geq \mathsf{R}_i$.*

Proof. If j is directly connected to i, then it is immediate that $\alpha_C(j) = D(i,j) \geq \mathsf{R}_i$.

Suppose that j is indirectly connected to facility i'. In this case, let i be the facility that j was first connected to. Since j is indirectly connected, there has to be a client j' that has special edges to both i and i'. We use $t(i)$ and $t(i')$ to denote the times at which facilities i and i' were respectively opened. Notice that $\alpha_C(j) = \alpha(j)$ by definition of α_C for indirectly connected clients and that $\alpha(j) = t(i)$ because j was connected to i before.

Case $t(i) \geq t(i')$: In this case we know that $\alpha_C(j) = t(i) \geq t(i') \geq D(i', j') \geq \mathsf{R}_{i'}$.

Case $t(i) < t(i')$: Since j' has special edges to i and i', it had tight edges to both before either was opened, i.e., $D(i', j') \leq t(i)$. Thus we can say $\alpha_C(j) = t(i) \geq D(i', j') \geq \mathsf{R}_{i'}$. ◀

Once again, we may assume that the Jain-Vazirani algorithm returns two solutions (x^1, y^1), (x^2, y^2) and their duals (α^1, Z^+) and (α^2, Z^-). It follows from Jain and Vazirani's analysis that the solutions have the following properties.

1. $\sum_i y_i^1 = k_1$ and $\sum_i y_i^2 = k_2$.
2. $\sum_j \alpha_C^1(j) = \sum_j \alpha_j^1 - k_1 \cdot Z^+$ and $\sum_j \alpha_C^2(j) = \sum_j \alpha_j^2 - k_2 \cdot Z^-$
3. $x_{i,j}^1 = 1$ or $x_{i,j}^2 = 1 \implies D(i,j) \leq 3\gamma \mathsf{R}_i$
4. $|Z^+ - Z^-| \leq \epsilon$
5. For $\rho = \frac{k_2 - k}{k_2 - k_1}$, $\rho(\alpha^1, Z^+) + (1-\rho)(\alpha^2, Z^-)$ is feasible for $\mathrm{LP}_{\mathcal{D}}^\gamma$.

For property 2, notice that $\alpha_C^1(j) = \alpha_j^1$ for indirectly connected clients, that $\alpha^1 C(j) = \alpha_j^1 - \beta_{\phi(j)j}$ for directly connected clients (where $\phi(j)$ is the center j is connected to) and that the sum of $\beta_{\phi(j)j}$ over all directly connected clients is just $k_1 Z^+$. The same holds for the second solution. Using property 5, we get that $\rho(\sum_j \alpha_j^1 - k \cdot Z^+) + (1-\rho)(\sum_j \alpha_j^2 - k_2 \cdot Z^-)$ is a lower bound for the optimal value of $\mathrm{LP}_{\mathcal{D}}^\gamma$ and thus also for the optimal value of $\mathrm{LP}_{\mathcal{P}}^\gamma$. Using property 2 and 4, this implies that $\rho(\sum_j \alpha_C^1(j)) + (1-\rho)(\sum_j \alpha_C^2(j)) \leq opt(\mathrm{LP}_{\mathcal{P}}^\gamma) + \varepsilon$. We apply Lemma 6 to replace x^1 and x^2 to ensure that the resulting graph $G(x^1, y^1, x^2, y^2)$ is a forest. Note that the procedure only reassigns the clients to facilities with smaller radius than their currently connected facility. Hence, we can still assume that $x_{i,j}^1 = 1 \implies \alpha_C^1(j) \geq \mathsf{R}_i$

(similarly $x^2_{i,j} = 1 \implies \alpha^2_C(j) \geq \mathsf{R}_i$). However, we may now have solutions that violate the locality constraints by a factor of 9γ.

Now we use the procedure described in Lemma 8 to partition the graph $G(x^1, y^1, x^2, y^2)$ into subtrees $T_1, \ldots T_\ell$ with $\mathsf{df}(T_p) = 1$ for $p \in \{1, \ldots, \ell\}$ and $\ell = \frac{k_2 - k_1}{2}$. Each tree has the property that $A(\delta(B(T_p))) \subseteq A(T_p)$ for all $p \in \{1, \ldots, \ell\}$. Since each edge in this tree represents some set of clients, we use the notation $j \in T_p$ to denote that j is associated with an edge in T_p. Define the cost of the subtree T_p as $\sum_{j \in T_p} \alpha^1_C(j)$. We choose the $k_2 - k$ cheapest such trees. Since choosing all ℓ subtrees will result in a cost of $\sum_j \alpha^1_C(j)$, we can say that the cost of these chosen subtrees is at most $\frac{2(k_2-k)}{k_2-k_1} \sum_j \alpha^1_C(j) = 2\rho \sum_j \alpha^1_C(j)$.

Our rounded solution $(\hat{x}, \hat{y})$ opens all facilities from A that are part of the chosen subtrees and all facilities from B that are not part of any chosen subtree. Notice that since we open $k_2 - k$ subtrees and these satisfy $\mathsf{df}(T_p) = 1$, $\hat{y}$ opens exactly k facilities. The assignments of clients to facilities follow x^1 and x^2, respectively.

Analogously to Lemma 3, we construct a solution to the aversion k-clustering problem based on $\hat{x}, \hat{y}$. In this solution, each client assigned to a facility $p_i \in A$ pays at most

$$D(j, j') \leq D(p_i, j) + D(p_j, j') \leq 9\gamma \mathsf{R}_i + 9\gamma \mathsf{R}_i \leq (2 \cdot 9\gamma)\alpha^1_C(j)$$

where j' is the furthest away client among all that are assigned to p_i. Thus, by our choice of subtrees, all clients assigned to A pay at most $(2 \cdot 9\gamma)2\rho \sum_j \alpha^1_C(j)$ in total. The remaining clients pay at most $(2 \cdot 9\gamma)(\sum_j \alpha^2_C(j))$. As before, we can assume that $\rho \leq 1/2$. We conclude that the cost of $(\hat{x}, \hat{y})$ is bounded by

$$\begin{aligned}
&(2 \cdot 9\gamma)\big(2\rho \sum_j \alpha^1_C(j) + \sum_j \alpha^2_C(j)\big) \\
&\leq 2(2 \cdot 9\gamma)(1+\varepsilon)\big(\rho \sum_j \alpha^1_C(j) + (1-\rho) \sum_j \alpha^2_C(j)\big) \\
&\leq 2(2 \cdot 9\gamma)(1+\varepsilon) opt(\mathrm{LP}^\gamma_{\mathcal{P}}) \\
&\leq 2(2 \cdot 9\gamma)2(1+\varepsilon) opt_a
\end{aligned}$$

where opt_a is the optimal value for the aversion k-clustering instance and the last inequality follows by Lemma 4. Since $\gamma = 3$, the approximation factor is bounded by $216 + \varepsilon$.

4 Final Thoughts and Conclusions

This paper shows a $(216+\epsilon)$-approximation to the aversion k-clustering problem. Our results rely on achieving a constant factor bicriteria approximation for local k-median instances arising from aversion k-clustering problem. Lemma 6 is the only place in our proof where we use that the local k-median instances are generated from aversion k-clustering instances. It remains an open question if we can get a constant factor bicriteria approximation for arbitrary instances of local k-median.

Acknowledgments. We thank Christian Kroer for introducing us to the aversion k-clustering problem, and for enlightening discussions. We thank anonymous reviewers for constructive feedback, in particular for pointing out an oversight in Lemma 6 in an earlier draft.

References

1 Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. *Mathematical Programming*, 154(1-2):29–53, 2015.

2 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

3 Michel Louis Balinski. On finding integer solutions to linear programs. Technical report, DTIC Document, 1964.

4 Yair Bartal, Moses Charikar, and Danny Raz. Approximating min-sum k-clustering in metric spaces. In *Proceedings of the 33rd STOC*, pages 11–20, 2001.

5 Babak Behsaz, Zachary Friggstad, Mohammad R. Salavatipour, and Rohit Sivakumar. Approximation algorithms for min-sum k-clustering and balanced k-median. In *Proceedings of the 42nd ICALP*, pages 116–128, 2015.

6 Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the 26th SODA*, pages 737–756, 2015.

7 Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

8 Moses Charikar, Sudipto Guha, Éva Tardos, and David B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

9 Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *Journal of Computer and System Sciences*, 68(2):417–441, 2004.

10 Julia Chuzhoy and Yuval Rabani. Approximating k-median with non-uniform capacities. In *Proceedings of the 16th SODA*, pages 952–958, 2005.

11 Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. LP rounding for k-centers with non-uniform hard capacities. In *Proceedings of the 53rd FOCS*, pages 273–282, 2012.

12 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

13 Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. *Journal of Algorithms*, 31(1):228–248, 1999.

14 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10:180–184, 1985.

15 Wen-Lian Hsu and George L. Nemhauser. Easy and hard bottleneck location problems. *Discrete Applied Mathematics*, 1:209–215, 1979.

16 Kamal Jain, Mohammad Mahdian, and Amin Saberi. A new greedy approach for facility location problems. In *Proceedings of the 34th STOC*, pages 731–740, 2002.

17 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

18 Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

19 Christian Kroer and Tuomas Sandholm. Extensive-Form Game Imperfect-Recall Abstractions With Bounds. *CoRR*, abs/1409.3302, 2014. also published at the Algorithmic Game Theory workshop at IJCAI, 2015.

20 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222:45–58, 2013.

21 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the 45th STOC*, pages 901–910, 2013.

22 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. Available at `http://www.designofapproxalgs.com`.

The Non-Uniform k-Center Problem

Deeparnab Chakrabarty[1], Prachi Goyal[2], and Ravishankar Krishnaswamy[3]

1 **Microsoft Research, Bangalore, India**
dechakr@microsoft.com
2 **Microsoft Research, Bangalore, India**
t-prgoya@microsoft.com
3 **Microsoft Research, Bangalore, India**
rakri@microsoft.com

Abstract

In this paper, we introduce and study the Non-Uniform k-Center (NUkC) problem. Given a finite metric space (X, d) and a collection of balls of radii $\{r_1 \geq \cdots \geq r_k\}$, the NUkC problem is to find a placement of their centers on the metric space and find the minimum dilation α, such that the union of balls of radius $\alpha \cdot r_i$ around the ith center covers all the points in X. This problem naturally arises as a min-max vehicle routing problem with fleets of different speeds, or as a wireless router placement problem with routers of different powers/ranges.

The NUkC problem generalizes the classic k-center problem when all the k radii are the same (which can be assumed to be 1 after scaling). It also generalizes the k-center with outliers (kCwO for short) problem when there are k balls of radius 1 and ℓ balls of radius 0. There are 2-approximation and 3-approximation algorithms known for these problems respectively; the former is best possible unless P=NP and the latter remains unimproved for 15 years.

We first observe that no $O(1)$-approximation is to the optimal dilation is possible unless P=NP, implying that the NUkC problem is more non-trivial than the above two problems. Our main algorithmic result is an $(O(1), O(1))$-*bi-criteria* approximation result: we give an $O(1)$-approximation to the optimal dilation, however, we may open $\Theta(1)$ centers of each radii. Our techniques also allow us to prove a simple (uni-criteria), optimal 2-approximation to the kCwO problem improving upon the long-standing 3-factor. Our main technical contribution is a connection between the NUkC problem and the so-called firefighter problems on trees which have been studied recently in the TCS community. We show NUkC is as hard as the firefighter problem. While we don't know if the converse is true, we are able to adapt ideas from recent works [4, 1] in non-trivial ways to obtain our constant factor bi-criteria approximation.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Clustering, k-Center, Approximation Algorithms, Firefighter Problem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.67

1 Introduction

Source location and vehicle routing problems are extremely well studied [20, 24, 10] in operations research. Consider the following location+routing problem: we are given a set of k ambulances with speeds $s_1, s_2, \ldots, s_k$ respectively, and we have to find the depot locations for these vehicles in a metric space (X, d) such that any point in the space can be served by some ambulance as fast as possible. If all speeds were the same, then we would place the ambulances in locations S such that $\max_{v \in X} d(v, S)$ is minimized – this is the famous

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 67; pp. 67:1–67:15


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

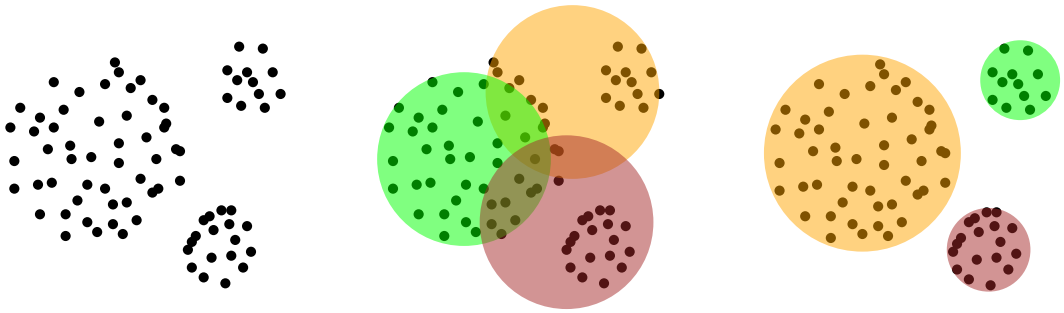


Figure 1 The left figure shows the dataset, the middle figure shows a traditional k-center clustering, and the right figure depicts a non-uniform clustering

k-center problem. Differing speeds, however, leads to non-uniformity, thus motivating the titular problem we consider.

▶ **Definition 1** (The Non-Uniform k-Center Problem (NUkC))**.** The input to the problem is a metric space (X, d) and a collection of k balls of radii $\{r_1 \geq r_2 \geq \cdots \geq r_k\}$. The objective is to find a placement $C \subseteq X$ of the centers of these balls, so as to minimize the dilation parameter α such that the union of balls of radius $\alpha \cdot r_i$ around the ith center covers all of X. Equivalently, we need to find centers $\{c_1, \ldots, c_k\}$ to minimize $\max_{v \in X} \min_{i=1}^{k} \frac{d(v,c_i)}{r_i}$.

As mentioned above, when all r_i's are the same (and equal to 1 by scaling), we get the k-center problem. The k-center problem was originally studied by Gonzalez [11] and Hochbaum and Shmoys [14] as a clustering problem of partitioning a metric space into different clusters to minimize maximum intra-cluster distances. One issue (see Figure 1 for an illustration and refer to [12] for a more detailed explanation) with k-center (and also k-median/means) as an objective function for clustering is that it favors clusters of similar sizes with respect to cluster radii. However, in presence of qualitative information on the differing cluster sizes as is often the case in certain applications, the non-uniform versions of the problem can arguably provide more nuanced solutions. One extreme special case was considered as the "clustering with outliers" problem [8] where a fixed number/fraction of points in the metric space need not be covered by the clusters. In particular, Charikar et al [8] consider (among many problems) the k-center with outlier problem (kCwO, for short) and show a 3-approximation for this problem. It is easy to see that kCwO is a special case of NUkC when there are k balls of radius 1 and ℓ (the number of outliers) balls of radius 0.

Motivated by the aforementioned reasons (both from facility location as well as from clustering settings) In this paper, we investigate the worst-case complexity of the NUkC problem. Gonzalez [11] and Hochbaum and Shmoys [14] give 2-approximations for the k-center problem, and also show that no better factor is possible unless P = NP. Charikar et al [8] give a 3-approximation for the kCwO problem, and this has been the best factor known for 15 years. Given these algorithms, it is natural to wonder if a simple $O(1)$-approximation exists for the NUkC problem. In fact, our first result shows a qualitative distinction between NUkC and these problems: constant-approximations are impossible for NUkC unless P=NP.

▶ **Theorem 2.** *For any constant $c \geq 1$, the* NUkC *problem does not admit a c-factor approximation unless $P = NP$, even when the underlying metric is a tree metric.*

The hardness result is by a reduction from the so-called *resource minimization for fire containment* problem on trees (RMFC-T, in short), a variant of the firefighter problem. To

circumvent the above hardness, we give the following bi-criteria approximation algorithm which is the main result of the paper, and which further highlights the connections with RMFC-T since our algorithms heavily rely on the recent algorithms for RMFC-T [4, 1]. An (a, b)-factor bi-criteria algorithm for NUkC returns a solution which places at most a balls of each type (thus in total it may use as many as $a \cdot k$ balls), and the dilation is at most b times the optimum dilation for the instance which places exactly one ball of each type.

▶ **Theorem 3.** *There is an $(O(1), O(1))$-factor bi-criteria algorithm for the* NUkC *problem.*

Furthermore, as we elucidate below, our techniques also give uni-criteria results when the number of distinct radii is 2. In particular, we get a 2-approximation for the kCwO problem and a $(1 + \sqrt{5})$-approximation when there are only two distinct types of radii.

▶ **Theorem 4.** *There is a 2-approximation for the* kCwO *problem.*

▶ **Theorem 5.** *There is a $(1 + \sqrt{5})$-approximation for the* NUkC *problem when the number of distinct radii is at most 2.*

1.1 Discussion on Techniques

Our proofs of Theorems 2 and 3 shows a strong connection between NUkC and the so-called *resource minimization for fire containment* problem on trees (RMFC-T, in short). This connection is one of the main findings of the paper, so we first formally define this problem.

▶ **Definition 6** (Resource Minimization for Fire Containment on Trees (RMFC-T))**.** Given a rooted tree T as input, the goal is to select a collection of non-root nodes N from T such that (a) every root-leaf path has at least one vertex from N, and (b) $\max_t |N \cap L_t|$ is minimized, where L_t is the tth-*layer* of T, that is, the vertices of T at exactly distance t from the root.

To understand the reason behind the name, consider a fire starting at the root spreading to neighboring vertices each day; the RMFC-T problem minimizes the number of firefighters needed per day so as to prevent the fire spreading to the leaves of T.

It is NP-hard to decide if the optimum of RMFC-T is 1 or not [9, 18]. Given any RMFC-T instance and any $c > 1$, we construct an NUkC instance on a tree metric such that in the "yes" case there is always a placement with dilation $= 1$ which covers the metric, while in the "no" case even a dilation of c doesn't help. Upon understanding our hardness construction, the inquisitive reader may wonder if the reduction also works in the other direction, i.e., whether we can solve NUkC using a reduction to RMFC-T problem. Unfortunately, we do not know if this is true even for two types of radii. However, as we explain below we still can use positive results for the RMFC-T problem to design good algorithms for the NUkC problem.

Indeed, we start off by considering the natural LP relaxation for the NUkC problem and describe an *LP-aware reduction* of NUkC to RMFC-T. More precisely, given a feasible solution to the LP-relaxation for the given NUkC instance $\mathcal{I}$, we describe a procedure to obtain an instance $\mathcal{I}'$ of RMFC-T, *and* also a feasible fractional solution for the natural LP relaxation of the RMFC-T problem on $\mathcal{I}'$. Moreover, given any feasible *integral* solution to the $\mathcal{I}'$, we can obtain a feasible solution to $\mathcal{I}$ which dilates the radii by a constant factor. An *LP-based* ρ-approximation to RMFC-T would then imply $(\rho, O(1))$-bi-criteria approximation algorithms for NUkC. Plugging in the result of Chalermsook and Chuzhoy [4], we directly obtain an $(O(\log^* n), O(1))$-bi-criteria approximation for NUkC. We can also obtain Theorem 4 and Theorem 5 since the corresponding RMFC-T instances have no integrality gap.

Here we reach a technical bottleneck: Chalermsook and Chuzhoy [4] also show that the integrality gap of the natural LP relaxation for RMFC-T is $\Omega(\log^* n)$. When combined with

our hardness reduction in Theorem 2 , this also implies a $(\Omega(\log^* n), c)$ integrality gap for any constant $c > 1$ for the natural LP relaxation for NUkC. That is, even if we allow a violation of c in the radius dilation, there is a $\Omega(\log^* n)$-integrality gap in terms of the violation in number of balls opened of each type. For RMFC-T though, Adjiashvili, Baggio and Zenklusen [1] recently showed an improved $O(1)$-approximation bypassing the LP integrality gap. At a very high level, the main technique in [1] is to carefully and efficiently "guess" a subset of the optimum solution, such that the natural LP-relaxation for covering only the uncovered leaves has $O(1)$-integrality gap. However, this guessing procedure crucially uses the tree structure of the given RMFC-T instance. Unfortunately for us though, we get the RMFC-T tree only after solving the LP for NUkC, which already has an $\Omega(\log^* n)$-gap! Nevertheless, inspired by the ideas in [1], we show that we can also efficiently "guess" the positions of a certain number of balls in an optimum solution, such that the standard LP-relaxation for covering the uncovered points has $O(1)$-gap. We can then invoke our reduction to RMFC-T to solve our problem. This is quite delicate, and is the most technically involved part of the paper.

1.2 Related Work and Open Questions

The k-center problem [11, 14] and the k-center with outliers [8] probems are classic problems in approximation algorithms and clustering. These problems have also been investigated under various settings such as the incremental model [6, 23], streaming model [5, 23], and more recently in the map-reduce model [15, 22]. Similarly, the k-median [7, 16, 21, 2] and k-means [16, 17, 13, 19] problems are also classic problems studied extensively in approximation algorithms and clustering. The generalization of k-median to a routing+location problem was also studied recently [10]. It would be interesting to explore the complexity of the non-uniform versions of these problems. Another direction would be to explore if the new non-uniform model can be useful in solving clustering problems arising in practice.

2 Hardness Reduction

In this section, we prove Theorem 2 based on the following NP-hardness [18] for RMFC-T.

▶ **Theorem 7** ([18]). *Given a tree T whose leaves are at the same distance from the root, it is NP-hard to distinguish between the following two cases.*
YES: *There is a solution to the* RMFC-T *instance of value* 1.
NO: *All solutions to the* RMFC-T *instance has value* 2.

Given an RMFC-T instance defined by tree T, we now describe the construction of our NUkC instance. Let h be the height of the tree, and let L_t denote the vertices of the tree at distance exactly t from the root. So, the leaves constitute L_h since all leaves are at the same distance from the root. The NUkC instance, $\mathcal{I}(T)$, is defined by the metric space (X, d), and a collection of balls. The points in our metric space will correspond to the leaves of the tree, i.e., $X = L_h$. To define the metric, we assign a weight $d(e) = (2c+1)^{h-i+1}$ for each edge whose one endpoint is in L_i and the other in L_{i-1}; we then define d be the shortest-path metric on X induced by this weighted tree. Finally, we set $k = h$, and define the k radii $r_1 \geq r_2 \geq \ldots \geq r_k$ iteratively as follows: define $r_k := 0$, and for $k \geq i > 1$, set $r_{i-1} := (2c+1) \cdot r_i + 2(2c+1)$. This completes the NUkC instance. Before proceeding we make the simple observation: for any two leaves u and u' with lca $v \in L_t$, we have $d(u, u') = 2(2c + 1 + (2c+1)^2 + \cdots + (2c+1)^{h-t}) = r_t$. The following lemma proves Theorem 2.

▶ **Lemma 8.** *If T is the YES case of Theorem 7, then $\mathcal{I}(T)$ has optimum dilation $= 2$. If T is the NO case of 7, then $\mathcal{I}(T)$ has optimum dilation $\geq 2c$.*

Proof. Suppose T is in the YES case, and there is a solution to RMFC-T which selects at most 1 node from each level L_t. If $v \in L_t$ is selected, then select a center c_v *arbitrarily* from any leaf in the sub-tree rooted at v and open the ball of radius r_t. We now need to show all points in $X = L_h$ are covered by these balls. Let u be any leaf; there must be a vertex v in some level L_t in u's path to the root such that a ball of radius r_t is opened at c_v. However, $d(u, c_v) \leq d(u, v) + d(v, c_v) = 2r_t$ and so the ball of radius $2r_t$ around c_v covers u.

Now suppose T is in the NO case, and the NUkC instance has a solution with optimum dilation $< 2c$. We build a good solution for the RMFC-T instance N as follows: suppose the NUkC solution opens the radius $< 2c \cdot r_t$ ball around center u. Let v be the vertex on the u-root path appearing in level L_t. We then pick this node in N. Observe two things: first, this ball covers all the leaves in the sub-tree rooted at v since $r_t \geq d(u, u')$ for any such u'. Furthermore, since the NUkC solution has only one ball of each radius, we get that $|N \cap L_t| \leq 1$. Finally, since $d(u, w) \geq 2c \cdot r_t$ for all leaves w not in the sub-tree rooted at v, the ball of radius r_t around u doesn't contain any leaves other than those rooted at v. Contra-positively, since all leaves w are covered in some ball, every leaf must lie in the sub-tree of some vertex picked in N. That is, N is a solution to RMFC-T with value $= 1$ contradicting the NO case. ◀

3 LP-aware reduction from NUkC to RMFC-T

For reasons which will be apparent soon, we consider instances $\mathcal{I}$ of NUkC counting multiplicites. That is, we consider an instance to be a collection of tuples $(k_1, r_1), \ldots, (k_h, r_h)$ to indicate there are k_i balls of radius r_i. So we have $r_1 \geq r_2 \ldots \geq r_h$ and $\sum_{t=1}^{h} k_t = k$. Intuitively, the reason we do this is that if two radii r_t and r_{t+1} are "close-by" then it makes sense to round up r_{t+1} to r_t and increase k_t, losing only a constant-factor loss in the dilation.

LP-relaxation for NUkC. We now state the natural LP relaxation for a given NUkC instance $\mathcal{I}$. For each $p \in X$ and radius type r_i, we have a variable $x_{p,i} \geq 0$ denoting the extent to which we place a ball of radius r_i centered at p. By doing a binary search on the optimal dilation and scaling, we may assume that the optimum dilation is 1. Then, the following linear program must be feasible. In what follows, define $B(q, r_i) = \{p : d(p, q) \leq r_i\}$.

$$\forall p \in X, \qquad \sum_{t=1}^{h} \sum_{q \in B(p, r_t)} x_{q,t} \geq 1 \qquad \text{(NUkC LP)}$$

$$\forall t \in 1, \cdots, h \qquad \sum_{p \in X} x_{p,t} \leq k_t$$

LP-relaxation for RMFC-T. Since we reduce fractional NUkC to fractional RMFC-T, we now state the natural LP relaxation for RMFC-T on a tree T of depth $h + 1$. In fact, we will work with the following budgeted-version of RMFC-T (which is equivalent to the original RMFC-T problem — for a proof, see [1]): Instead of minimizing the maximum number of "firefighters" at any level t (that is $|N \cap L_t|$ where N is the chosen solution), suppose we specify a budget limit of k_t on $|N \cap L_t|$. The goal is the minimize the maximum dilation of these budgets. Then the following is a natural LP relaxation for the budgeted RMFC-T problem on trees. Here $L = L_h$ is the set of leaves, and L_t are the layer t-nodes. For a leaf

node v, we let P_v denote the vertex set of the unique leaf-root path excluding the root.

$$\begin{array}{lll} & \min \alpha & \\ \forall v \in L, & \sum_{u \in P_v} y_u & \geq 1 \\ \forall t \in 1, \cdots, h & \sum_{u \in L_t} y_u & \leq \alpha \cdot k_t \end{array} \qquad \text{(RMFC-T LP)}$$

The LP-aware Reduction to Tree metrics. We now describe our main reduction algorithm, which takes as input an NUkC instance $\mathcal{I} = \{(X, d); (k_1, r_1), \ldots, (k_h, r_h)\}$ and a feasible solution x to NUkC LP, and returns a budgeted RMFC-T instance $\mathcal{I}_T$ defined by a tree T along with budgets for each level, and a feasible solution y to RMFC-T LP with dilation 1. The tree we construct will have height $h + 1$ and the budgeted RMFC-T instance will have budgets precisely k_t at level $1 \leq t \leq h$, and the budget for the leaf level is 0. For clarity, throughout this section we use the word *points* to denote elements of the metric space in $\mathcal{I}$, and the word *vertices/nodes* to denote the tree nodes in the RMFC-T instance that we construct. We build the tree T in a bottom-up manner, where in each round, we pick a set of far-away representative points (the distance scale increases as we move up the tree) and cluster all points to their nearest representative. This is similar to a so-called clustering step in many known algorithms for facility location (see e.g., [7]), but whereas an arbitrary set of far-away representatives would suffice in the facility location algorithms, we need to be careful in how we choose this set to make the overall algorithm work.

Formally, each vertex of the tree T is mapped to some point in X, and we denote the mapping of the vertices at level t by $\psi_t : L_t \to X$. We will maintain that each ψ_t will be injective, so $\psi_t(u) \neq \psi_t(v)$ for $u \neq v$ in L_t. So, ψ_t^{-1} is well defined for the range of ψ_t.

The complete algorithm runs in rounds $h + 1$ to 2 building the tree one level per round. To begin with, the ψ_{h+1} mapping is an arbitrary bijective mapping between $L := L_{h+1}$, the set of leaves of the tree, and the points of X (so, in particular, $|L| = |X|$). We may assume it to be the identity bijection. In each round t, the range of the mappings become progressively smaller, that is[1], $\psi_t(L_t) \supseteq \psi_{t-1}(L_{t-1})$. We call $\psi_t(L_t)$ as the **winners at level** t. We now describe round t. Let $\mathsf{Cov}_t(p) := \sum_{q \in B(p, r_t)} x_{q,t}$ denote the fractional amount the point p is covered by radius r_t balls in the solution x. Also define $\mathsf{Cov}_{\geq t}(p) := \sum_{s \geq t} \mathsf{Cov}_s(p)$ denoting the fractional amount p is covered by radius r_t or smaller balls. Let $\mathsf{Cov}_{h+1}(p) = 0$ for all p. Finally, we add a root vertex and connect it to all vertices in L_1. This gives us the final tree T and a solution y which assigns a value to all non-leaf, non-root vertices of the tree T. The following claim asserts well-definedness of the algorithm.

▶ **Lemma 9.** *The solution y is a feasible solution to* RMFC-T *LP on $\mathcal{I}_T$ with dilation* 1.

Proof. The proof is via two claims for the two different set of inequalities.

▶ **Claim 1.** *For all $1 \leq t \leq h$, we have $\sum_{w \in L_t} y_w \leq k_t$.*

Proof. Fix t. Let $W_t \subseteq X$ denote the winners at level t, that is, $W_t = \psi_t(L_t)$. By definition of the algorithm, $\sum_{w \in L_t} y_w = \sum_{p \in W_t} \mathsf{Cov}_t(p)$. Now note that for any two points $p, q \in W_t$, we have $B(p, r_t) \cap B(q, r_t) = \emptyset$. To see this, consider the first point which enters A in the $(t + 1)$th round when L_t was being formed. If this is p, then all points in the radius $2r_t$ ball

[1] We are using the notation $\psi(X) := \bigcup_{x \in X} \psi(x)$.

Algorithm 1 Round t of the LP-aware Reduction.

Input: Level L_t, subtrees below L_t, the mappings $\psi_s : L_s \to X$ for all $t \le s \le h$.
Output: Level L_{t-1}, the connections between L_{t-1} and L_t, and the mapping ψ_{t-1}.

Define $A = \psi_t(L_t)$ the set of points who are winners at level t.
while $A \neq \emptyset$ **do**
(a) Choose the point $p \in A$ with *minimum coverage* $\mathsf{Cov}_{\ge t}(p)$.
(b) Let $N(p) := \{q \in A : d(p,q) \le 2r_{t-1}\}$ be the set of all nearby points in A to p.
(c) Create a new tree vertex $w \in L_{t-1}$ corresponding to p and set $\psi_{t-1}(w) := p$. *Call p a winner at level $t-1$, and each $q \in N(p) \subseteq A$ a loser to p at this level.*
(d) Create edge (w, v) for tree vertices $v \in \psi_t^{-1}(N(p))$ associated with $N(p)$ at level t.
(e) Set $A \leftarrow A \setminus (N(p))$.
(f) Set $y_w = \mathsf{Cov}_{t-1}(p)$.
end while

are deleted from A. Since the balls are disjoint, the second inequality of NUkC LP implies $\sum_{p \in W_t} \sum_{q \in B(p,r_t)} x_{q,t} \le k_t$. The second summand in the LHS is precisely $\mathsf{Cov}_t(p)$. ◀

▶ **Claim 2.** *For any leaf node $w \in L$, we have $\sum_{v \in P_w} y_v \ge 1$.*

Proof. We start with an observation. Fix a level t and a winner point $p \in W_t$. Let $u \in L_t$ such that $\psi_t(u) = p$. Since $W_t \subseteq W_{t+1} \subseteq \cdots \subseteq W_h$, there is a leaf v in the subtree rooted at u corresponding to p. Moreover, by the way we formed our tree edges in step (d), we have that $\psi_s(w') = p$ for all w' in the (u,v)-path and hence $\sum_{w' \in [u,v]\text{-path}} y_{w'} = \mathsf{Cov}_{\ge t}(p)$.

Now, for contradiction, suppose there is some leaf corresponding to, say point p, such that the root-leaf path has total y-assignment less than 1. Then, pick the point, among all such unsatisfied points p, who appears in a winning set W_t with t *as small as possible*.

By the preceding observation, the total y-assignment p receives on its path from level h to level t is exactly $\mathsf{Cov}_{\ge t}(p)$. Moreover, suppose p loses to q at level $t-1$, i.e., $\psi_t^{-1}(p)$ is a child of $\psi_{t-1}^{-1}(q)$. In particular, it means that q has also been a winner up to level t and so the total y-assignment on q's path upto level t is also precisely $\mathsf{Cov}_{\ge t}(q)$. Additionally, since $\psi_{t-1}^{-1}(q)$ became the parent node for $\psi_t^{-1}(p)$, we know that $\mathsf{Cov}_{\ge t}(q) \le \mathsf{Cov}_{\ge t}(p)$ due to the way we choose winners in step (a) of the while loop. Finally, by our maximality assumption on p, we know that q is fractionally satisfied by the y-solution. Therefore, there is fractional assignment of at least $(1 - \mathsf{Cov}_{\ge t}(q))$ on q's path from nodes in level $t-1$ to level 1. Putting these observations together, we get that the total fractional assignment on p's root-leaf path is at least $\mathsf{Cov}_{\ge t}(p) + (1 - \mathsf{Cov}_{\ge t}(q)) \ge 1$, which results in the desired contradiction. ◀

◀

The following lemma shows that any good *integral* solution to the RMFC-T instance $\mathcal{I}_T$ can be converted to a good integral solution for the NUkC instance $\mathcal{I}$.

▶ **Lemma 10.** *Suppose there exists a feasible solution N to $\mathcal{I}_T$ such that for all $1 \le t \le h$, $|N \cap L_t| \le \alpha k_t$. Then there is a solution to the NUkC instance $\mathcal{I}$ that opens, for each $1 \le t \le h$, at most αk_t balls of radius $\le 2r_{\ge t}$, where $r_{\ge t} := r_t + r_{t+1} + \cdots + r_h$.*

Proof. Construct the NUkC solution as follows: for level $1 \le t \le h$ and every vertex $w \in N \cap L_t$, place the center at $\psi_t(w)$ of radius $2r_{\ge t}$. We claim that every point in X is covered by some ball. Indeed, for any $p \in X$, look at the leaf $v = \psi_{h+1}(p)$, and let $w \in N$ be a node in the root-leaf path. Let $w \in L_t$ for some t. Now observe that $d(p, \psi_t(w)) \le 2r_{\ge t}$;

this is because for any edge (u', v') in the tree where u' is in L_t and is the parent of v', we have that $d(\psi_{t+1}(v'), \psi_{t+1}(u')) < 2r_t$. ◀

This completes the reduction, and we now prove a few results using this.

▶ **Theorem 11.** *There is a polynomial time $(\mathcal{O}(\log^* n), 8)$-bi-criteria algorithm for* NUkC.

Proof. Given any instance $\mathcal{I}$ of NUkC, we first club the radii to the nearest power of 2 to get an instance $\mathcal{I}'$ with radii $(k_1, r_1), \cdots, (k_h, r_h)$ such that an (a, b)-factor solution for $\mathcal{I}'$ is an $(a, 2b)$-solution for $\mathcal{I}$. Now, by scaling, we assume that the optimal dilation for $\mathcal{I}'$ is 1; we let x be the feasible solution to the NUkC LP. Then, using Algorithm 1, we can construct the tree $\mathcal{I}'_T$ and a feasible solution y to the RMFC-T LP. We can now use the following theorem of Chalermsook and Chuzhoy [4]: given any feasible solution to the RMFC-T LP, we can obtain a feasible set N covering all the leaves such that for all t, $|N \cap L_t| \leq \mathcal{O}(\log^* n)k_t$. Finally, we can apply Lemma 10 to obtain a $(\mathcal{O}(\log^* n), 4)$ solution to $\mathcal{I}'$ (since $r_{\geq t} \leq 2r_t$). ◀

Proof of Theorem 4 and Theorem 5. We use the following claim regarding the integrality gap of RMFC-T LP for depth 2 trees.

▶ **Claim 3.** *When $h = 2$ and k_t's are integers, given any fractional solution to* RMFC-T *LP, we can find a feasible integral solution as well.*

Proof. Given a feasible solution y to RMFC-T LP, we need to find a set N such that $|N \cap L_t| \leq k_t$ for $t = 1, 2$. There must exist at least one vertex $w \in L_1$ such that $y_w \in (0, 1)$ for otherwise the solution y is trivially integral. If only one vertex $w \in L_1$ is fractional, then since k_1 is an integer, we can raise this y_w to be an integer as well. So at least two vertices w and w' in L_1 are fractional. Now, without loss of generality, let us assume that $|C(w)| \geq |C(w')|$, where $C(w)$ is the set of children of w. Now for some small constant $0 < \epsilon < 1$, we do the following: $y'_w := y_w + \epsilon$, $y'_{w'} := y_{w'} - \epsilon$, $\forall c \in C(w)$, $y'_c := y_c - \epsilon$, and $\forall c \in C(w')$, $y'_c := y_c + \epsilon$. Note that $y(L_1)$ remains unchanged, $y(L_2)$ can only decrease, and root-leaf paths still add to at least 1. We repeat this till we rule out all fractional values. ◀

To see the proof of Theorem 4, note that an instance of the k-center with outliers problem is an NUkC instance with $(k, 1), (\ell, 0)$, that is, $r_1 = 1$ and $r_2 = 0$. We solve the LP relaxation and obtain the tree and an RMFC-T solution. The above claim implies a feasible integral solution to RMFC-T since $h = 2$, and finally note that $r_{\geq 1} = r_1$ for kCwO, implying we get a 2-factor approximation.

The proof of Theorem 5 is similar. If $r_1 < \theta r_2$ where $\theta = (\sqrt{5} + 1)/2$, then we simply run k-center with $k = k_1 + k_2$. This gives a $2\theta = \sqrt{5} + 1$-approximation. Otherwise, we apply Lemma 10 to get a $2(1 + \frac{1}{\theta}) = \sqrt{5} + 1$-approximation. ◀

We end this section with a general theorem, which is an improvement over Lemma 10 in the case when many of the radius types are close to each other, in which case $r_{\geq t}$ could be much larger than r_t. Indeed, the natural way to overcome this would be to group the radius types into geometrically increasing values as we did in the proof of Theorem 11. However, for some technical reasons we will not be able to bucket the radius types in the following section, since we would instead be bucketing the *number of balls* of each radius type in a geometric manner. Instead, we can easily modify Algorithm 1 to build the tree by focusing only on radius types where the radii grow geometrically.

▶ **Theorem 12.** *Given an* NUkC *instance $\mathcal{I} = \{M = (X, d), (k_1, r_1), (k_2, r_2), \ldots, (k_h, r_h)\}$ and an LP solution x for* NUkC *LP, there is an efficient reduction which generates an* RMFC-T *instance $\mathcal{I}_T$ and an LP solution y to* RMFC-T *LP, such that the following holds:*

(i) *For any two tree vertices $w \in L_t$ and $v \in L_{t'}$ where w is an ancestor of v (which means $t \leq t'$), suppose p and q are the corresponding points in the metric space, i.e., $p = \psi_t(w)$ and $q = \psi_{t'}(v)$, then it holds that $d(p,q) \leq 8 \cdot r_t$.*

(ii) *Suppose there exists a feasible solution N to $\mathcal{I}_T$ such that for all $1 \leq t \leq h$, $|N \cap L_t| \leq \alpha k_t$. Then there is a solution to the* NUkC *instance $\mathcal{I}$ that opens, for each $1 \leq t \leq h$, at most αk_t balls of radius at most $8 \cdot r_t$.*

4 Getting an $(O(1), O(1))$-approximation algorithm

In this section, we improve our approximation factor on the number of clusters from $O(\log^* n)$ to $O(1)$, while maintaining a constant-approximation in the radius dilation. As mentioned in the introduction, this requires more ideas since using NUkC LP one cannot get any factor better than $(O(\log^* n), O(1))$-bi-criteria approximation since any integrality gap for RMFC-T LP translates to a $(\Omega(\log^* n), \Omega(1))$ integrality gap for NUkC LP.

Our algorithm is heavily inspired by the recent paper of Adjiashvili et al [1] who give an $O(1)$-approximation for the RMFC-T problem. In fact, the structure of our algorithms follows the same three "steps" of their algorithm. Given an RMFC-T instance, in [1] the authors first "compress" the input tree to get a new tree whose depth is only logarithmic; next, they give a partial rounding result which saves "bottom heavy" leaves, that is, leaves which are fractionally covered to at least a constant fraction by low level tree nodes; and finally, they give a clever *partial enumeration algorithm* for guessing the nodes from the top levels chosen by the optimum solution. We also proceed in these three steps with the first two being very similar. However, the enumeration step requires new ideas for our problem. In particular, the enumeration procedure in [1] crucially uses the tree structure of the firefighter instance, and the way our reduction generates the tree for the RMFC-T instance is by using the optimal LP solution for the NUkC instance, which in itself suffers from the $\Omega(\log^* n)$ integrality gap. Therefore, we need to devise a more sophisticated enumeration scheme inspired by the one in [1]. Throughout this section, we do not optimize for the constants.

4.1 Part I: Radii Reduction

In this part, we describe a preprocessing step which decreases the number of types of radii. This is similar to Theorem 5 in [1].

▶ **Theorem 13.** *Let $\mathcal{I}$ be an* NUkC *instance with radii $\{r_1, r_2, \cdots, r_k\}$. We can efficiently compute instance $\widehat{\mathcal{I}}$ with radii multiplicities $(k_0, \widehat{r}_0), ..., (k_L, \widehat{r}_L)$ and $L = \Theta(\log k)$ such that:*

(i) *$k_i := 2^i$ for all $0 \leq i < L$ and $k_L \leq 2^L$.*

(ii) *If the* NUkC *instance $\mathcal{I}$ has a feasible solution, then there exists a feasible solution for $\widehat{\mathcal{I}}$.*

(iii) *Given an (α, β)-bi-criteria solution to $\widehat{\mathcal{I}}$, we can efficiently obtain a $(3\alpha, \beta)$-bi-criteria solution to $\mathcal{I}$.*

Proof. For an instance $\mathcal{I}$, we construct the compressed instance $\widehat{\mathcal{I}}$ as follows. Partition the radii into $\Theta(\log k)$ classes by defining barriers at $\widehat{r}_i = r_{2^i}$ for $0 \leq i \leq \lfloor \log k \rfloor$. Now to create instance $\widehat{\mathcal{I}}$, we simply round up all the radii r_j for $2^i \leq j < 2^{i+1}$ to the value $\widehat{r}_i = r_{2^i}$. Notice that the multiplicity of $\widehat{r}_i$ is precisely 2^i (except maybe for the last bucket, where there might be fewer radii rounded up than the budget allowed).

Property (i) follows by construction. Property (ii) follows from the way we rounded up the radii. Indeed, if the optimal solution for $\mathcal{I}$ opens a ball of radius r_j around a point p,

then we can open a ball of radius $\widehat{r}_i$ around p, where i is such that $2^i \le j < 2^{i+1}$. Clearly the number of balls of radius $\widehat{r}_i$ is at most 2^i since OPT uses at most one ball of each radius r_j.

For property (iii), suppose we have a solution $\widehat{S}$ for $\widehat{\mathcal{I}}$ which opens $\alpha 2^i$ clusters of radius $\beta\widehat{r}_i$ for all $0 \le i \le L$. Construct a solution S for $\mathcal{I}$ as follows. For each $1 \le i \le L$, let C_i denote the set of centers where $\widehat{S}$ opens balls of radius $\beta\widehat{r}_i$. In the solution S, we also open balls at precisely these centers with 2α balls of radius r_j for every $2^{i-1} \le j < 2^i$. Since $|C_i| \le \alpha \cdot 2^i$, we can open a ball at every point in C_i; furthermore, since $j < 2^i$, we have $r_j \ge \widehat{r}_i$ and so we cover whatever the balls from $\widehat{S}$ covered.

Finally, we also open the α clusters (corresponding to $i = 0$) of radius $\beta r_1 = \beta\widehat{r}_0$ at the respective centers C_0 where $\widehat{S}$ opens centers of radius $\widehat{r}_0$. Therefore, the total number of clusters of radius type is at most 2α with the exception of r_1, which may have 3α clusters. ◀

4.2 Part II: Satisfying Bottom Heavy Points

One main reason why the above height reduction step is useful, is the following theorem from [1] for RMFC-T instances on trees; we provide a proof sketch for completeness.

▶ **Theorem 14** ([1]). *Given a tree T of height h and a feasible solution y to (*RMFC-T *LP), we can find a feasible integral solution N to* RMFC-T *such that for all $1 \le t \le h$, $|N \cap L_t| \le k_t + h$.*

Proof. Let y be a basic feasible solution of (RMFC-T LP). Call a vertex v of the tree *loose* if $y_v > 0$ and the sum of y-mass on the vertices from v to the root (inclusive of v) is < 1. Let V_L be the set of loose vertices of the tree, and let V_I be the set of vertices with $y_v = 1$. Clearly $N = V_L \cup V_I$ is a feasible solution: every leaf-to-root path either contains an integral vertex or at least two fractional vertices with the vertex closer to root being loose. Next we claim that $|V_L| \le h$; this proves the theorem since $|N \cap L_t| \le |V_I \cap L_t| + |V_L| \le k_t + |V_L|$.

The full proof can be found in Lemma 6, [1] – here is a high level sketch. There are $|L| + h$ inequalities in (RMFC-T LP), and so the number of fractional variables is at most $|L| + h$. We may assume there are no $y_v = 1$ vertices. Now, in every leaf-to-root path there must be at least 2 fractional vertices, and the one closest to the leaf must be non-loose. If the closest fractional vertex to each leaf was unique, then that would account for $|L|$ fractional non-loose vertices implying the number of loose vertices must be $\le h$. This may not be true; however, if we look at *linearly independent* set of inequalities that are tight, we can argue uniqueness as a clash can be used to exhibit linear dependence between the tight constraints. ◀

▶ **Theorem 15.** *Suppose we are given an* NUkC *instance $\widehat{\mathcal{I}}$ with radii multiplicities $(k_0, \widehat{r}_0), (k_1, \widehat{r}_1), \ldots, (k_L, \widehat{r}_L)$ with budgets $k_i = 2^i$ for radius type $\widehat{r}_i$, and an LP solution x to* (NUkC LP) *for $\widehat{\mathcal{I}}$. Let $\tau = \log\log L$, and suppose $X' \subseteq X$ be the points covered mostly by small radii, that is, let $\mathsf{Cov}_{\ge\tau}(p) \ge \frac{1}{2}$ for every $p \in X'$. Then, there is an efficient procedure* round *which opens at most $O(k_t)$ balls of radius $O(\widehat{r}_t)$ for $\tau \le t \le L$, and covers all of X'.*

Proof. The procedure round works as follows: we partition the points of X' into two sets, one set X_U in which the points receive at least $\frac{1}{4}$ of the coverage by clusters of radius $\widehat{r}_i$ where $i \in \{\log\log L, \log\log L + 1, \ldots, \log L\}$, and another set X_B in which the points receive $\frac{1}{4}$ coverage from clusters of levels $t \in \{\log L + 1, \log L + 2, \ldots, L\}$. More precisely, $X_U := \{p \in X' : \sum_{t=\tau}^{\log L} \mathsf{Cov}_t(p) \ge 1/4\}$, and $X_B = X' \setminus X_B$.

Now consider the following LP-solution to (NUkC LP) for $\widehat{\mathcal{I}}$ restricted to X_U: we scale x by a factor 4 and zero-out x on radii type $\widehat{r}_i$ for $i \notin \{\log\log L, \ldots, \log L\}$. By definition of X_U this is a feasible fractional solution; furthermore, the LP-reduction algorithm described

in Section 3 will lead to a tree T of height $\leq \log L$ and fractional solution y for (RMFC-T LP) on T were each $k_i \geq 2^{\log\log L} = \log L$. Applying Theorem 14, we can find an integral solution N with at most $O(k_i)$ vertices at levels $i \in \{\log\log L, \ldots, \log L\}$. We can then translate this solution back using Theorem 12 to NUkC and find $O(k_t)$ clusters of radius $O(\widehat{r}_t)$ to cover all the points X_U. A similar argument, when applied to the smaller radius types $\widehat{r}_t$ for $t \in \{\log L, \ldots, L\}$ can cover the points in X_B. ◀

We now show how we can immediately also get a (very weakly) quasi-polynomial time $O(1)$-approximation for NUkC. Indeed, if we could enumerate the set of clusters of radii $\widehat{r}_t$ for $0 \leq t < \log\log L$, we can then explicitly solve an LP where all the uncovered points need to be fractionally covered by only clusters of radius type $\widehat{r}_t$ for $t \geq \log\log L$. We can then round this solution using Corollary 15 to obtain the desired $O(1)$-approximation for the NUkC instance. Moreover, the time complexity of enumerating the optimal clusters of radii $\widehat{r}_t$ for $0 \leq t < \log\log L$ is $n^{O(\log L)} = n^{O(\log\log k)}$, since the number of clusters of radius at least $\widehat{r}_{\log\log L}$ is at most $O(2^{\log\log L}) = O(\log L)$. Finally, there was nothing special in the proof of Corollary 15 about the choice of $\tau = \log\log L$ — we could set $t = \log^{(q)} L$ to be the q^{th} iterated logarithm of L, and obtain an $O(q)$-approximation. As a result, we get the following corollary. Note that this gives an alternate way to prove Theorem 11.

▶ **Corollary 16.** *For any $q \geq 1$, there exists an $(O(q), O(1))$-factor bi-criteria algorithm for* NUkC *which runs in $n^{O(\log^{(q)} k)}$ time.*

4.3 Part III: Clever Enumeration of Large Radii Clusters

In this section, we show how to obtain the $(O(1), O(1))$-factor bi-criteria algorithm. At a high level, our algorithm tries to "guess" the centers[2] A of large radius, that is $\widehat{r}_i$ for $i \leq \tau := \log\log L = \log\log\log k$, which the optimum solution uses. However, this guessing is done in a cleverer way than in Corollary 16. In particular, given a guess which is consistent with the optimum solution (the initial "null set" guess is trivially consistent), our enumeration procedure generates a list of candidate additions to A of size *at most* $2^\tau \approx \text{poly}\log\log k$ (instead of n), one of which is a consistent enhancement of the guessed set A. This reduction in number of candidates also requires us to maintain a guess D of points where the optimum solution *doesn't* open centers. Furthermore, we need to argue that the "depth of recursion" is also bounded by $\text{poly}\log\log k$; this crucially uses the technology developed in Section 3. Altogether, we get the total time is at most $(\text{poly}\log\log k)^{\text{poly}\log\log k} = o(k)$ for large k. In this extended abstract with page limits, we omit all proofs in this subsection, and point the reader to the full version of our paper [3].

We start with some definitions. Throughout, A and D represent sets of tuples of the form (p, t) where $p \in X$ and $t \in \{0, 1, \ldots, \tau\}$. Given such a set A, we associate a partial solution S_A which opens a ball of radius $22\widehat{r}_t$ at the point p for all p s.t. $(p, t) \in A$. For the sake of analysis, fix an optimum solution OPT. We say the set A is **consistent** with OPT if for all $(p, t) \in A$, there exists a *unique* $q \in X$ such that OPT opens a ball of radius $\widehat{r}_t$ at q and $d(p, q) \leq 11\widehat{r}_t$. In particular, this implies that S_A covers all points which this OPT-ball covers. We say the set D is **consistent** with OPT if for all $(q, t) \in D$, OPT *doesn't* open a radius $\widehat{r}_t$ ball at q (it may open a different radius ball at q though). Given a pair of sets

[2] Actually, we end up guessing centers "close" to the optimum centers, but for this introductory paragraph this intuition is adequate.

(A,D), we define the $\mathsf{minLevel}$ of each point p to be the type of largest radius consistent with our guesses which can cover it, i.e.,

$$\mathsf{minLevel}_{A,D}(p) := 1 + \arg\max_t \{(q,t) \in D \text{ for all } q \in B(p, \widehat{r}_t)\}$$

If (A, D) is a consistent pair and $\mathsf{minLevel}_{A,D}(p) = t$, then this implies in the OPT solution, p is covered by a ball of radius $\widehat{r}_t$ or smaller.

Next, we describe a nuanced LP-relaxation for NUkC. Fix a pair of sets (A, D) as described above. Let X_G be the subset of points in X covered by the partial solution S_A. Fix a subset $Y \subseteq X \setminus X_G$ of points. Define the following LP.

$$\begin{aligned}
\forall p \in Y, \quad & \sum_{t=\mathsf{minLevel}(p)}^{L} \sum_{q \in B(p,\widehat{r}_t)} x_{q,t} \geq 1 && (\mathrm{LP}_{\mathsf{NUkC}}(Y, A, D)) \\
\forall t \in 1, \cdots, h \quad & \sum_{q \in Y} x_{q,t} \leq k_t \\
\forall (p,t) \in A, \quad & x_{p,t} = 1
\end{aligned}$$

The following claim encapsulates the utility of the above relaxation.

▶ **Claim 4.** *If (A, D) is consistent with OPT, then $LP_{\mathsf{NUkC}}(X \setminus X_G, A, D)$ is feasible.*

Finally, for convenience, we define a **forbidden set** $F := \{(p, i) : p \in X, 1 \leq i \leq \tau\}$ which if added to D disallows any large radii balls to be placed anywhere.

Now we are ready to describe the enumeration Algorithm 2. We start with A and D being null, and thus vacuously consistent with OPT. The enumeration procedure ensures that: given a consistent (A, D) tuple, either it finds a good solution using LP rounding (Step 10), or generates candidate additions (Steps 18–20) to A or D ensuring that one of them leads to a larger consistent tuple.

Define $\gamma_0 := 4 \log\log k \cdot \log\log\log k$. The algorithm is run with $\mathsf{Enum}(\emptyset, \emptyset, \gamma_0)$. The proof that we get a polynomial time $(O(1), O(1))$-bi-criteria approximation algorithm follows from three lemmas. Lemma 17 shows that if Step 10 is true with a consistent pair (A, D), then the output in Step 13 is a $(O(1), O(1))$-approximation. Lemma 18 shows that indeed Step 10 is true for γ_0 as set. Finally, Lemma 19 shows that the algorithm runs in polynomial time.

▶ **Lemma 17.** *If (A, D) is a consistent pair such that Step 10 is true, then the solution returned is an $(O(1), O(1))$-approximation algorithm.*

▶ **Lemma 18.** $\mathsf{Enum}(\emptyset, \emptyset, \gamma_0)$ *finds consistent (A, D) such that Step 10 is true.*

▶ **Lemma 19.** $\mathsf{Enum}(\emptyset, \emptyset, \gamma_0)$ *runs in polynomial time for large enough k.*

Algorithm 2 $\mathsf{Enum}(A, D, \gamma)$

Let $X_G = \{p : \exists\,(q,i) \in A \text{ s.t } d(p,q) \leq 22\widehat{r}_i\}$ denote points covered by S_A.
if there is **no** feasible solution to $LP_{\mathsf{NUkC}}(X \setminus X_G, A, D)$ **then**
  Abort. *// Claim 4 implies (A, D) is not consistent.*
else
  Let x^* be a feasible solution to $LP_{\mathsf{NUkC}}(X \setminus X_G, A, D)$.
end if
Let $X_B = \{u \in X \setminus X_G : \mathsf{Cov}_{\geq\tau}(u) \geq \frac{1}{2}\}$ denote bottom-heavy points in x^*
Let S_B be the solution implied by Corollary 15.
// This solution opens $O(k_t)$ balls of radius $O(\widehat{r}_t)$ for $\tau \leq t \leq L$ and covers all of X_B.
Let $X_T = X \setminus (X_G \cup X_B)$ denote the top heavy points in x^*
if $LP_{\mathsf{NUkC}}(X_T, A, F \cup D)$ has a feasible solution x_T **then**
  By definition of F, in x_T we have $\mathsf{Cov}_{\geq\tau}(u) = 1$ for all $u \in X_T$.
  Let S_T be the solution implied by Corollary 15.
  // This solution opens $O(k_t)$ balls of radius $O(\widehat{r}_t)$ for $\tau \leq t \leq L$ and covers all of X_T.
  Output $(S_A \cup S_B \cup S_T)$. *//This is a $(O(1), O(1))$-approximation for the* NUkC *instance.*
else
  for every level $0 \leq t \leq \tau$ **do**
    Let $C_t = \{p \in X_T \text{ s.t } \mathsf{minLevel}_{A,D}(p) = t\}$, the set of points in X_T with minLevel t.
    Use the LP-aware reduction from Section 3 using x^* and the set of points C_t to create tree T_t.
    for every winner p at level t in T_t **do**
      $\mathsf{Enum}(A \cup \{(p,t)\}, D, \gamma - 1)$
      $\mathsf{Enum}(A, D \cup \bigcup_{p' \in B(p, 11\widehat{r}_t)} \{(p', t)\}), \gamma - 1)$
    end for
  end for
end if

References

1. D. Adjiashvili, A. Baggio, and R. Zenklusen. Firefighting on trees beyond integrality gaps. *CoRR*, abs/1601.00271, 2016. URL: `http://arxiv.org/abs/1601.00271`.
2. J. Byrka, T. Pensyl, B. Rybicki, A. Srinivasan, and K. Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2015.
3. D. Chakrabarty, P. Goyal, and R. Krishnaswamy. The non-uniform k-center problem. *Available on arXiv, and authors webpage (May, 2016)*, 2016.
4. P. Chalermsook and J. Chuzhoy. Resource minimization for fire containment. *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.
5. M. Charikar, L. O' Callaghan, and R. Panigrahy. Better streaming algorithms for clustering problems. *ACM Symp. on Theory of Computing (STOC)*, 2003.
6. M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic infomation retrieval. *ACM Symp. on Theory of Computing (STOC)*, 1997.
7. M. Charikar, S. Guha, D. Shmoys, and E. Tardos. A constant-factor approximation algorithm for the k-median problem. *ACM Symp. on Theory of Computing (STOC)*, 1999.
8. M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2001.
9. S. Finbow, A. King, G. MacGillivray, and R. Rizzi. The firefighter problem for graphs of maximum degree three. *Discrete Mathematics*, 307(16):2094–2105, 2007.
10. I. L. Goertz and V. Nagarajan. Locating depots for capacitated vehicle routing. *Proceedings, International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2011.
11. T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.
12. S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. *Proceedings of SIGMOD*, 1998.
13. S. Har-Peled and S. Mazumdar. Coresets for k-means and k-median clustering and their applications. *ACM Symp. on Theory of Computing (STOC)*, 2004.
14. D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985.
15. S. Im and B. Moseley. Fast and better distributed mapreduce algorithms for k-center clustering. *Proceedings, ACM Symposium on Parallelism in Algorithms and Architectures*, 2015.
16. K. Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001.
17. T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu. A local search approximation algorithm for k-means clustering. In *Proceedings of the 18th Annual Symposium on Computational Geometry (SoCG'02)*, 2002.
18. A. King and G. MacGillivray. The firefighter problem for cubic graphs. *Discrete Mathematics*, 310(3):614–621, 2010.
19. A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1 +)$-approximation algorithm for k-means clustering in any dimensions. *Proceedings, IEEE Symposium on Foundations of Computer Science (FOCS)*, 2004.
20. G. Laporte. Location routing problems. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 163–198. 1998.
21. S. Li and O. Svensson. Approximating k-median via pseudo-approximation. *ACM Symp. on Theory of Computing (STOC)*, 2013.

22 G. Malkomes, M. J. Kusner, W. Chen, K. Q. Weinberger, and B. Moseley. Fast distributed k-center clustering with outliers on massive data. *Advances in Neur. Inf. Proc. Sys. (NIPS)*, 2015.

23 R. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. *Proceedings, International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2008.

24 H. Min, V. Jayaraman, and R. Srivastava. Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research*, 108:1–15, 1998.

k-Center Clustering Under Perturbation Resilience*†

Maria-Florina Balcan[1], Nika Haghtalab[2], and Colin White[3]

1 **Carnegie Mellon University, Pittsburgh, USA**
ninamf@cs.cmu.edu

2 **Carnegie Mellon University, Pittsburgh, USA**
nhaghtal@cs.cmu.edu

3 **Carnegie Mellon University, Pittsburgh, USA**
crwhite@cs.cmu.edu

Abstract

The k-center problem is a canonical and long-studied facility location and clustering problem with many applications in both its symmetric and asymmetric forms. Both versions of the problem have tight approximation factors on worst case instances: a 2-approximation for symmetric k-center and an O(log*(k))-approximation for the asymmetric version. Therefore to improve on these ratios, one must go beyond the worst case.

In this work, we take this approach and provide strong positive results both for the asymmetric and symmetric k-center problems under a very natural input stability (promise) condition called alpha-perturbation resilience [Bilu Linial, 2012], which states that the optimal solution does not change under any alpha-factor perturbation to the input distances. We show that by assuming 2-perturbation resilience, the exact solution for the asymmetric k-center problem can be found in polynomial time. To our knowledge, this is the first problem that is hard to approximate to any constant factor in the worst case, yet can be optimally solved in polynomial time under perturbation resilience for a constant value of alpha. Furthermore, we prove our result is tight by showing symmetric k-center under (2-epsilon)-perturbation resilience is hard unless NP=RP. This is the first tight result for any problem under perturbation resilience, i.e., this is the first time the exact value of alpha for which the problem switches from being NP-hard to efficiently computable has been found.

Our results illustrate a surprising relationship between symmetric and asymmetric k-center instances under perturbation resilience. Unlike approximation ratio, for which symmetric k-center is easily solved to a factor of 2 but asymmetric k-center cannot be approximated to any constant factor, both symmetric and asymmetric k-center can be solved optimally under resilience to 2-perturbations.

1998 ACM Subject Classification I.5.3 Clustering

Keywords and phrases k-center, clustering, perturbation resilience

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.68

* Full version of the paper available at http://arxiv.org/abs/1505.03924.

† This work was supported in part by grants NSF-CCF 1535967, NSF CCF-1422910, NSF CCF-145117, a Sloan Research Fellowship, a Microsoft Research Faculty Fellowship, a Google Research Award, an IBM Ph.D. fellowship, and a National Defense Science & Engineering Graduate (NDSEG) fellowship.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 68; pp. 68:1–68:14

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

1 Introduction

Overview: Traditionally, the theory of algorithms has focused on the analysis of worst-case instances. While this approach has led to many elegant algorithms and strong lower bounds, it tends to be overly pessimistic of an algorithm's performance on the most typical instances of a problem. A recent line of work in the algorithms community, the so called *beyond worst case analysis* of algorithms, considers the question of designing algorithms for instances that satisfy some natural structural properties and has given rise to strong positive results [4, 5, 6, 18, 20, 21, 24]. One of the most appealing properties that has been proposed in this space is the stability of the solution to small changes in the input. Bilu and Linial [10] formalized this property in the notion of α-*perturbation resilience*, which states that the optimal solution does not change under any α-factor perturbation to the input distances.

A large body of work has sought to exploit the power of perturbation resilience in problems such as center-based clustering [5, 8, 10, 22], finding Nash equilibria in game theoretic problems [7], and the traveling salesman problem [23]. These works are focused on providing positive results for exactly solving the corresponding optimization problem under perturbation resilient instances, for example, $1+\sqrt{2}$-perturbation resilience for center based clustering, and $O(\sqrt{\log n}\log\log n)$-perturbation resilience for max-cut. In this paper we continue this line of work and provide a tight result for the canonical and long-studied k-center clustering problem, thereby completely quantifying the power of perturbation resilience for this problem. We show that $\alpha = 2$ is the moment where the problem switches from NP-hard to efficiently computable – specifically, we show that by assuming 2-perturbation resilience, the exact solution for the k-center problem can be found in polynomial time; we also show that k-center under $(2-\epsilon)$-perturbation resilience cannot be solved in polynomial time unless $NP = RP$. Our results apply to both symmetric and asymmetric k-center, illustrating a surprising relationship between symmetric and asymmetric k-center instances under perturbation resilience. Unlike approximation ratio, for which symmetric k-center is easily solved to a factor of 2 but asymmetric k-center cannot be approximated to any constant factor, both symmetric and asymmetric k-center can be solved optimally under resilience to 2-perturbations. Overall, this is the first tight result quantifying the power of perturbation resilience for a canonical combinatorial optimization problem.

The k-center problem is a canonical and long-studied clustering problem with many applications to facility location, data clustering, image classification, and information retrieval [11, 12, 13, 14, 16, 25]. For example, it can be used to solve the problem of placing k fire stations spaced throughout a city to minimize the maximum time for a fire truck to reach any location, given the pairwise travel times between important locations in the city. In the symmetric k-center problem the distances are assumed to be symmetric, while in the asymmetric k-center problem they are not; however in both cases they satisfy the triangle inequality. Formally, given a set of n points S, a distance function $d : S \times S \to R^+$ satisfying the triangle inequality (and symmetry in the symmetric case), and an integer k, our goal is to find k centers $\{c_1, \ldots, c_k\}$ to minimize $\max_{p\in S} \min_i d(c_i, p)$.

Both forms of k-center admit tight approximation bounds. For symmetric k-center, several 2- approximation algorithms have been found starting in the mid 1980s (e.g., [16, 19]). This is the best possible approximation factor by a simple reduction from set cover. On the other hand, the asymmetric k-center problem is a prototypical problem where the best known approximation is superconstant and is matched by a lower bound. For the asymmetric k-center problem, an $O(\log^*(n))$-approximation algorithm was found by Vishwanathan [25], and later improved to $O(\log^*(k))$ by Archer [1]. This approximation ratio was shown to be

asymptotically tight by the work of Chuzhoy et al. [14], which built upon a sequence of papers establishing the hardness of approximating d-uniform hypergraph covering (culminating in [15]).

Perturbation resilience has a natural interpretation for both symmetirc and asymmetric k-center: it can be viewed as a stability condition in the presence of uncertainties involved in measurements. For example, small fluctuations in the travel time between a fire station and locations in the city, which are caused by different levels of traffic at different times of day, should not drastically affect the optimal placement of fire stations. Furthermore, perturbation resilience can be viewed as a condition on an instance under which the optimal solution satisfies a form of privacy. For instance, if the actions of no individuals (such as how they drive to work, or the amount of network traffic they are using in a network application) can affect the overall state of the problem drastically then the individual's actions cannot be detected by looking at the optimal solution.

There is a large body of work on instances satisfying perturbation resilience and other natural notions of stability on problems ranging from clustering to data privacy to social networks to topic modeling [2, 3, 17, 18, 20, 21, 24]. For discussion of related work, see the full version of the paper.

Our Results: In this work we consider both symmetric and asymmetric k-center under perturbation resilience and give tight results for both forms. In addition, we consider more robust and weaker variants of perturbation resilience, and give strong results for these problems as well. A summary of our results and techniques used to achieve them are as follows:

1. *Efficient algorithm for symmetric and asymmetric k-center under 2-perturbation resilience.* This directly improves over the result of Balcan and Liang [8] for symmetric k-center under $1+\sqrt{2}$-perturbation resilience. We show that *any* α-approximation algorithm returns the optimal solution for an α-perturbation resilient instance, thus showing there exists an optimal algorithm for symmetric k-center under 2-perturbation resilience. For the asymmetric result, we first construct a "symmetrized set" by only considering points that demonstrate a rough symmetry. Then we prove strong structural results about the symmetrized set which motivates a novel algorithm for detecting clusters locally.
2. *Hardness of symmetric k-center under $(2-\epsilon)$-perturbation resilience.* This shows that our perturbation-resilience results are tight for both symmetric and asymmetric k-center. For this hardness result, we use a reduction from a variant of perfect dominating set. To show that this variant is itself hard, we construct a chain of parsimonious reductions (reductions which conserve the number of solutions) from 3-dimensional matching to perfect dominating set.
3. *Efficient algorithms for symmetric and asymmetric k-center under $(3,\epsilon)$-perturbation resilience.* A clustering instance satisfies (α,ϵ)-perturbation resilience if $\leq \epsilon n$ points switch clusters under any α-perturbation. We assume the optimal clusters are of size $> 2\epsilon n$ (the problem is NP-hard without this assumption). We show that if any single point p is close to an optimal cluster other than its own, then $k-1$ centers achieve the optimal score under a carefully constructed 3-perturbation. Any other point we add to the set of centers must create a clustering that is ϵ-close to $\mathcal{OPT}$, and we show all of these sets cannot simultaneously be consistent with one another, thus causing a contradiction. A key concept in our analysis is defining the notion of a *cluster-capturing center*, which allows us to reason about which points can capture a cluster when its center is removed.
4. *Efficient algorithm for any center-based clustering objective under weak center proximity.* Weak center proximity asks that each point be closer to its own center than to any point

from any other cluster, but note that it allows a cluster center to be closer to points from different clusters than to its own. Thus it is not at all obvious whether efficient optimal clustering is possible in such a setting. We present a novel linkage-based algorithm that is able to do so. It works by iteratively running single linkage as a subroutine until all clusters are balanced, and then removing all but the very last link.

The novelty of our results are manifold. First, our work is the first to provide a tight perturbation resilience result, thereby painting the complete picture for k-center under perturbation resilience. Second, this is the first result where a problem is not approximable to any constant in the worst-case, but can be optimally solved under resilience to small constant perturbations. Third, we are the first to consider an asymmetric problem under stability. Our results here illustrate a stark contrast between worst-case analysis and analysis of algorithms under stability. Unlike approximation ratio, for which symmetric k-center is easily solved to a factor of 2 but asymmetric k-center cannot be approximated to any constant factor, both symmetric and asymmetric k-center can be solved optimally under the same constant level of resilience.

2 Preliminaries

We define a clustering instance as (S, d), where S is a set of n points and $d : S \times S \to \mathbb{R}_{\geq 0}$ is a distance function. In the k-center problem, the goal is to find a set of points $\vec{p} = \{p_1, \ldots, p_k\} \subseteq S$ called *centers* such that the maximum distance from any point to its closest center is minimized. More formally, in the k-center problem, given a Voronoi partition $\mathcal{P} = \{P_1, \ldots, P_k\}$ induced by a set of centers $\vec{p} = \{p_1, \ldots, p_k\}$ (where for all $1 \leq i \leq k$, $p_i \in P_i$), we refer to $\mathcal{P}$ as a clusering, and define its cost by $\Phi(\mathcal{P}) = \max_{i \in [k]} \max_{v \in P_i} d(p_i, v)$. We indicate by $\mathcal{OPT}$ the clustering $\{C_1, \ldots, \mathcal{C}_k\}$ with minimum cost, we denote the optimal centers $\{c_1, \ldots, c_k\}$, and we denote the optimal cost $\Phi(\mathcal{OPT})$ by r^*, the maximum cluster radius.

We study the k-center clustering of instance (S, d) under two types of distance functions, *symmetric* and *asymmetric*. A symmetric distance function is a metric. An asymmetric distance function satisfies all the properties of a metric space, except for symmetry. That is, it may be the case that for some $p, q \in S$, $d(p, q) \neq d(q, p)$. Note that the k-center objective function for asymmetric instances is the same as the symmetric case, the maximum distance *from the center to the points*, where the order now matters.

We consider *perturbation resilience*, a notion of stability introduced by Bilu & Linial [10]. Perturbation resilience implies that the optimal clustering does not change under small perturbations of the distance measure. Formally, d' is called an α-perturbation of distance function d, if for all $p, q \in S$, $d(p, q) \leq d'(p, q) \leq \alpha d(p, q)$.[1] Perturbation resilience is defined formally as follows.

▶ **Definition 1.** A clustering instance (S, d) satisfies *α-perturbation resilience* for k-center, if for any α-perturbation d' of d, the optimal k-center clustering under d' is unique and equal to $\mathcal{OPT}$.

Note that the optimal centers may change, but the Voronoi partition $C_1, \ldots, C_k$ induced by them must stay the same. We do *not* assume that d' satisfies the triangle inequality.[2] We

[1] WLOG, we only consider perturbations in which the distances increases because we can scale the distances to simulate decreasing distances.

[2] This is well-justified, as the data may be gathered from heuristics or an average of measurements.

also consider a more robust variant of α-perturbation resilience, called (α, ϵ)-perturbation resilience, that allows a small change in the optimal clustering when distances are perturbed. To this end, we say that two clusterings $\mathcal{C}$ and $\mathcal{C}'$ are ϵ-*close*, if only an ϵ-fraction of the input points are clustered differently in the two clusterings, i.e., $\min_\sigma \sum_{i=1}^{k} |C_i \setminus C'_{\sigma(i)}| \leq \epsilon n$, where σ is a permutation on $[k]$. Formally,

▶ **Definition 2.** A clustering instance (S, d) satisfies (α, ϵ)-*perturbation resilience* for k-center, if for any α-perturbation d' of d, any optimal k-center clustering $\mathcal{C}'$ under d' is ϵ-close to $\mathcal{OPT}$.

We use ϵ-*far* to denote two clusters which are not ϵ-close. We also discuss the strictly stronger notion of approximation stability [6], which requires *any* α-approximation (not just a Voronoi partition) to be ϵ-close to $\mathcal{OPT}$. This is formally defined in Section 3.3. In Section 5, we define *center-based* objectives [8], a more general class of clustering functions which includes objective functions such as k-center, k-median, and k-means. Throughout this work, we use $B_r(c)$ to denote a ball of radius r centered at point r. Also for a point p and a set D, $d(p, D)$ denotes the distance from p to the farthest point in D.

3 2-perturbation resilience

In this section, we provide efficient algorithms for finding $\mathcal{OPT}$ for symmetric and asymmetric instances of k-center under 2-perturbation resilience. Our result directly improves on the result of Balcan and Liang for symmetric k-center under $(1 + \sqrt{2})$-perturbation resilience [8]. We also show that it is NP-hard to recover $\mathcal{OPT}$ even in the symmetric k-center instance under $(2 - \epsilon)$-approximation stability. As an immediate consequence, our results are tight for both perturbation resilience and approximation stability, for symmetric and asymmetric k-center instances. This is the first problem for which the exact value of perturbation resilience is found ($\alpha = 2$), where the problem switches from efficiently computable to NP-hard.

In the remainder of this section, first we show that any α-approximation algorithm returns the optimal solution for α-perturbation resilient instances. An immediate consequence is an algorithm for symmetric k-center under 2-perturbation resilience. Then we provide a novel algorithm for asymmetric k-center under 2-perturbation resilience.

3.1 Approximation algorithms under perturbation resilience

The following lemma allows us to reason about a specific type of α-perturbation we construct. This lemma will be important throughout the analysis in this section and in Section 4.

▶ **Lemma 3.** *For all $\alpha \geq 1$, given an α-perturbation d' of d with the following property: for all p, q, if $d(p, q) \geq r^*$ then $d'(p, q) \geq \alpha r^*$. Then the optimal cost under d' is αr^*.*

Proof. Clearly the optimal cost under d' cannot be greater than αr^*, since d' is an α-perturbation. Suppose there exists a set of centers $c'_1, \ldots, c'_k$ under d' that achieves a cost $< \alpha r^*$. Then for all i and all $p \in C'_i$, $d'(c'_i, p) < \alpha r^*$. But then by assumption, $d(c'_i, p) < r^*$. This implies that $c'_1, \ldots, c'_k$ achieve an optimal cost $< r^*$ under d, which is a contradiction. ◀

The following theorem will imply that any α-approximation algorithm for k-center will return the optimal solution on clustering instances that are α-perturbation resilient.

▶ **Theorem 4.** *Given a clustering instance (S, d) satisfying α-perturbation resilience for asymmetric k-center. Given a set C of k centers which is an α-approximation, i.e., $\forall p \in S$, $\exists c \in C$ s.t. $d(c, p) \leq \alpha r^*$. Then the Voronoi partition induced by C is the optimal clustering.*

Proof. For a point $p \in S$, let $c(p) := \text{argmin}_{c \in C} d(c, p)$, the closest center in C to p. The idea is to construct an α-perturbation in which C is the optimal solution by increasing all distances except between p and $c(p)$, for all p. Then the theorem will follow by using the definition of perturbation resilience.

By assumption, $\forall p \in S$, $d(c(p), p) \leq \alpha r^*$. Create a perturbation d' as follows. Increase all distances by a factor of α, except for all $p \in S$, set $d'(c(p), p) = \min(\alpha d(c(p), p), \alpha r^*)$ (recall in Definition 1, the perturbation need not satisfy the triangle inequality). Then no distances were increased by more than a factor of α. And since we had that $d(c(p), p) \leq \alpha r^*$, no distances decrease either. Therefore, d' is an α-perturbation of d. By Lemma 3, the optimal cost for d' is αr^*. Also, C achieves cost $\leq \alpha r^*$ by construction, so C is an optimal set of centers under d'. Then by α-perturbation resilience, the Voronoi partition induced by C under d' is the optimal clustering.

Finally, we show the Voronoi partition of C under d is the same as the Voronoi partition of C under d'. Given $p \in S$ whose closest point in C is $c(p)$ under d, then under d', all distances from p to $C \setminus \{c(p)\}$ increased by exactly α, and $d(p, c(p))$ increased by $\leq \alpha$. Therefore, the closest point in C to p under d' is still $c(p)$. ◀

An immediate consequence is that we have exact algorithms for symmetric k-center under 2-perturbation resilience, and asymmetric k-center under $O(\log^*(k))$-perturbation resilience. Now we show it is possible to substantially improve the latter result.

3.2 Asymmetric k-center algorithm

One of the challenges involved in dealing with asymmetric k-center instances is the fact that even though for all $p \in C_i$, $d(c_i, p) \leq r^*$, $d(p, c_i)$ might be arbitrarily large. Such points for which $d(p, c_i) \gg r^*$ pose a challenge to the structure of the clusters, as they can be very close to points or even centers of other clusters. To deal with this challenge, we first define a set of "good" points, A, such that $A = \{p \mid \forall q, d(q, p) \leq r^* \implies d(p, q) \leq r^*\}$. Intuitively speaking, these points behave similarly to a set of points with symmetric distances up to a distance r^*. To explore this, we define a desirable property of A with respect to the optimal clustering.

▶ **Definition 5.** A is said to *respect the structure of* $\mathcal{OPT}$ if (1) $c_i \in A$ for all $i \in [k]$, and (2) for all $p \in S \setminus A$, if $A(p) := \arg\min_{q \in A} d(q, p) \in C_i$, then $p \in C_i$.

For all i, define $C'_i = C_i \cap A$ (which is in fact the optimal clustering of A, although we do not need to prove this). Satisfying Definition 5 implies that if we can optimally cluster A, then we can optimally cluster the entire instance (formalized in Theorem 8). Thus our goal is to show that A does indeed respect the structure of $\mathcal{OPT}$, and to show how to return $C'_1, \ldots, C'_k$.

Intuitively, A is similar to a symmetric 2-perturbation resilient clustering instance. However, some structure is no longer there, for instance, a point p may be at distance $\leq 2r^*$ from every point in a different cluster, which is not true for 2-perturbation resilient instances. This implies we cannot simply run a 2-approximation algorithm on the set A, as we did in the previous section. However, we show that the remaining structural properties are sufficient to optimally cluster A. To this end, we define two properties and show how they lead to an algorithm that returns $C'_1, \ldots, C'_k$, and help us prove that A respects the structure of $\mathcal{OPT}$.

The first of these properties requires each point to be closer to its center than any point in another cluster. That is, *Property (1): For all $p \in C'_i$ and $q \in C'_j$, $i \neq j$, $d(c_i, p) < d(q, p)$.*

The second property requires that any point within distance r^* of a cluster center belongs to that cluster. That is, *Property (2): For all $i \neq j$ and $q \in C_j$, $d(q, c_i) > r^*$.*[3]

Let us illustrate how these properties allow us to optimally cluster A.[4] Consider a ball of radius r^* around a center c_i. By Property 2, such a ball exactly captures C'_i. Furthermore, by Property 1, any point in this ball is closer to the center than to points outside of the ball. Is this true for a ball of radius r^* around a general point p? Not necessarily. If this ball contains a point $q \in C'_j$ from a different cluster, then q will be closer to a point outside the ball than to p (namely, c_j, which is guaranteed to be outside of the ball by Property 2). This allows us to determine that the center of such a ball must not be an optimal center.

This structure motivates our Algorithm 1 for asymmetric k-center under 2-perturbation resilience. At a high level, we start by constructing the set A (which can be done easily in polynomial time). Then we create the set of all balls of radius r^* around all points in A (if r^* is not known, we can use a guess-and-check wrapper). Next, we prune this set by throwing out any ball that contains a point farther from its center than to a point outside the ball. We also throw out any ball that is a subset of another one. Our claim is that the remaining balls are exactly $C'_1, \ldots, C'_k$. Finally, we add the points in $S \setminus A$ to their closest point in A.

Algorithm 1 Asymmetric k-center algorithm under 2-PR

Input: Asymmetric k-center instance (S, d), distance r^* (or try all possible candidates).

1. Build set $A = \{p \mid \forall q, d(q, p) \leq r^* \implies d(p, q) \leq r^*\}$
2. $\forall c \in A$, construct $G_c = B_{r^*}(c)$ (the ball of radius r^* around c).
3. $\forall G_c$, if $\exists p \in G_c$, $q \notin G_c$ s.t. $d(q, p) < d(c, p)$, then throw out G_c.
4. $\forall p, q$ s.t. $G_p \subseteq G_q$, throw out G_p.
5. $\forall p \notin A$, add p to G_q, where $q = \arg\min_{s \in A} d(s, p)$.

Output: Output the sets $G_1, \ldots, G_k$.

▶ **Lemma 6.** *Properties 1 & 2 hold for asymmetric k-center instances under 2-perturbation resilience.*

Proof sketch. For Property 2, assume that there exists c_i and $q \in C_j$, $i \neq j$, such that $d(q, c_i) \leq r^*$. We construct a 2-perturbation in which q becomes the center for C_i. Increase all distances by a factor of 2, except for the distances from q to C_i, which we increase until they reach $2r^*$. By Lemma 3, this 2-perturbation achieves a cost of $2r^*$. However, q is distance $2r^*$ to C_i, so it must replace c_i as an optimal center. Then q and c_j are no longer in the same cluster, causing a contradiction.

The first property was shown to hold for symmetric instances by Awasthi et al. and the same proof can be used for asymmetric instances. This proof appears in the full version. ◀

▶ **Lemma 7.** *A respects the structure of $\mathcal{OPT}$.*

We defer this proof to the full version of the paper.

[3] Property (1) first appeared in the work of Awasthi et al. [5], for symmetric clustering instances. A weaker variation of Property (2) was introduced by Balcan and Liang [8], which showed that in $1 + \sqrt{2}$-perturbation resilient instances for any cluster C_i with radius r_i, $B_{r_i}(c_i) = C_i$. Our Property (2) shows that this is true *for a universal radius, r^**, even for 2-perturbation resilient instances, *and* even for asymmetric instances.

[4] Other algorithms work, such as single linkage with dynamic programming at the end to find the minimum cost pruning of k clusters. However, our algorithm is able to recognize optimal clusters *locally* (without a complete view of the point set).

▶ **Theorem 8.** *Algorithm 1 returns the exact solution for asymmetric k-center under 2-perturbation resilience.*

Proof. First we must show that after step 4, the remaining sets are exactly $C'_1, \ldots, C'_k = C_1 \cap A, \ldots, C_k \cap A$. We prove this in three steps: the sets G_{c_i} correspond to C'_i, these sets are not thrown out in steps 3 and 4, and all other sets are thrown out in steps 3 and 4. Because of Lemma 6, we can use Properties 1 and 2.

For all i, $G_{c_i} = C'_i$: From Lemma 7, all centers are in A, so G_{c_i} will be created in step 2. For all $p \in C_i$, $d(c_i, p) \leq r^*$. For all $q \notin C'_i$, then by Property 2, $d(q, c_i) > r^*$ (and since $c_i, q \in A$, $d(c_i, q) > r^*$ as well). For all i, G_{c_i} is not thrown out in step 3: Given $s \in G_{c_i}$ and $t \notin G_{c_i}$. Then $s \in C'_i$ and $t \in C'_j$ for $j \neq i$. If $d(t, s) < d(c_i, s)$, then we get a contradiction from Property 1. For all non-centers p, G_p is thrown out in step 3 or 4: From the previous paragraph, $G_{c_i} = C'_i$. If $G_p \subseteq G_{c_i}$, then G_p will be thrown out in step 4 (if $G_p = G_{c_i}$, it does not matter which set we keep, so WLOG say that we keep G_{c_i}). Then if G_p is not thrown out in step 4, $\exists s \in G_p \cap C'_j$, $j \neq i$. If $s = c_j$, then $d(p, c_j) \leq r^*$ and we get a contradiction from Property 2. So, we can assume s is a non-center (and that $c_j \notin G_p$). But $d(c_j, s) < d(p, s)$ from Property 1, and therefore G_p will be thrown out in step 3. Thus, the remaining sets after step 4 are exactly $C'_1, \ldots, C'_k$.

Finally, by Lemma 7, for each $p \in C_i \setminus A$, $A(p) \in C_i$, so p will be added to G_{c_i}. Therefore, the final output is $C_1, \ldots, C_k$. ◀

3.3 Hardness for k-center under $(2 - \epsilon)$-approximation stability

In this section, we consider approximation stability, introduced by Balcan et al. [6], which is strictly stronger than perturbation resilience. We show that if symmetric k-center under $(2-\epsilon)$-approximation stability can be solved in polynomial time, then $NP = RP$, even under the condition that the optimal clusters are all $\geq \frac{n}{2k}$. Because approximation stability is stronger than perturbation resilience, this result implies k-center under $(2 - \epsilon)$-perturbation resilience is hard as well. Similarly, symmetric k-center is a special case of asymmetric k-center, so we get the same hardness results for asymmetric k-center. This proves that Theorem 8 is tight.

Approximation stability requires constant approximations to the optimal cost to differ from $\mathcal{OPT}$ by at most an ϵ-fraction of the points.

▶ **Definition 9.** A clustering instance (S, d) satisfies (α, ϵ)-*approximation stability* for k-center, if for any partition $\mathcal{C}'$ with objective value r' (not necessarily a Voronoi partition), if $r' \leq \alpha r^*$, then $\mathcal{C}'$ is ϵ-close to $\mathcal{OPT}$.

It is not hard to see that (α, ϵ)-approximation stability implies (α, ϵ)-perturbation resilience, as the optimal clustering under any α-perturbation costs at most αr^* under the original distance function, d. So, a violating instance of (α, ϵ)-perturbation resilience induces a partition which costs $\leq \alpha r^*$ and is ϵ-far from $\mathcal{OPT}$, and therefore is not (α, ϵ)-approximation stable.

▶ **Theorem 10.** *There is no polytime algorithm for finding the optimal k-center clustering under $(2 - \epsilon)$-approximation stability, even when assuming all optimal clusters are size $\geq \frac{n}{2k}$, unless $NP = RP$.*

We show a reduction from a special case of Dominating Set which we call Unambiguous-Balanced-Perfect Dominating Set. A reduction from Perfect Dominating Set (Dominating Set with the additional constraint that for all dominating sets of size $\leq k$, each vertex is hit by

exactly one dominator) to the problem of clustering under $(2-\epsilon)$-center proximity was shown in [9] (α-center proximity is the property that for all $p \in C_i$ and $j \neq i$, $\alpha d(c_i, p) < d(c_j, p)$, and it follows from α-perturbation resilience). Our contribution is to show that Perfect Dominating Set remains hard under two additional conditions. First, in the case of a YES instance, each dominator must hit at least $\frac{n}{2k}$ vertices (which translates to clusters having size at least $\frac{n}{2k}$ as well). Second, we are promised that there is at most one dominating set of size $\leq k$ (which is required for establishing approximation stability for the resulting clustering instance).

4 Robust perturbation resilience

In this section, we consider (α, ϵ)-perturbation resilience. We show that under $(3, \epsilon)$-perturbation resilience, there is an algorithm that recovers $\mathcal{OPT}$ for symmetric k-center, and an algorithm that returns a solution that is ϵ-close to $\mathcal{OPT}$ for asymmetric k-center. For both of these results, we assume a lower bound on the size of the optimal clusters, $|C_i| > 2\epsilon n$ for all $i \in [k]$. We show the lower bound on cluster sizes is necessary; in its absence, the problem becomes NP-hard for all values of $\alpha \geq 1$ and $\epsilon > 0$. The theorems in this section require a careful reasoning about sets of centers under different perturbations that cannot all simultaneously be valid.

4.1 Symmetric k-center

We show that for any $(3, \epsilon)$-perturbation resilient k-center instance such that $|C_i| > 2\epsilon n$ for all $i \in [k]$, $\mathcal{OPT}$ can be found by simply thresholding the input graph using distance r^* and outputting the connected components. A nice feature of our result is that the Single Linkage algorithm, a fast algorithm widely used in practice, is sufficient to optimally cluster these instances.

▶ **Theorem 11.** *Given a $(3, \epsilon)$-perturbation resilient k-center instance (S, d) where all optimal clusters are $> \max(2\epsilon n, 3)$. Then the optimal clusters in $\mathcal{OPT}$ are exactly the connected components of the threshold graph G_{r^*} of the input distances.*

Proof idea. Since each optimal cluster center is distance r^* from all points in its cluster, it suffices to show that any two points in different clusters are at least r^* apart from each other. Assume on the contrary that there exist $p \in C_i$ and $q \in C_j$, $i \neq j$, such that $d(p, q) \leq r^*$. First we find a set of $k+2$ points and a 3-perturbation d', such that every size k subset of the points are optimal centers under d'. Then we show how this leads to a contradiction under $(3, \epsilon)$-perturbation resilience.

From our assumption, p is distance $\leq 3r^*$ from every point in $C_i \cup C_j$ (by the triangle inequality). Under a 3-perturbation in which all distances are blown up by a factor of 3 except $d(p, C_i \cup C_j)$, then replacing c_i and c_j with p would still give us a set of $k-1$ centers that achieve the optimal score. But, *would this contradict $(3, \epsilon)$-perturbation resilience?* Indeed, not! Perturbation resilience requires exactly k *distinct* centers.[5] The key challenge is to pick a final "dummy" center to guarantee that the Voronoi partition is ϵ-far from $\mathcal{OPT}$. The dummy center might "accidentally" be the closest center for almost all points in C_i or C_j. Even worse, it might be the case that the new center sets off a chain reaction in which it

[5] This distinction is well-motivated; if for some application, the best k-center solution is to put two centers at the same location, then we could achieve the exact same solution with $k-1$ centers. That implies we should have been running k'-center for $k' = k-1$ instead of k.

becomes center to a cluster C_x, and c_x becomes center to C_j, which would also result in a partition that is not ϵ-far from $\mathcal{OPT}$.

To deal with the chain reactions, we crucially introduce the notion of a *cluster capturing center* (CCC). c_x is a CCC for C_y, if for all but ϵn points $p \in C_y$, $d(c_x, p) \leq r^*$ and for all $i \neq x, y$, $d(c_x, p) < d(c_i, p)$. Intuitively, a CCC exists if and only if c_x is a valid center for C_y when c_y is taken out of the set of optimal centers (i.e., a chain reaction will occur). We argue that if a CCC does not exist then every dummy center we pick must be close to either C_i or C_j, since there are no chain reactions. If there does exist a CCC c_x for C_y, then we cannot reason about what happens to the dummy centers under our d'. However, we can define a new d'' by increasing all distances except $d(c_x, C_y)$, which allows us to take c_y out of the set of optimal centers, and then any dummy center must be close to C_x or C_y. There are no chain reactions because *we already know c_x is the best center for C_y* among the original optimal centers. Thus, whether or not there exists a CCC, we can find $k+2$ points close to the entire dataset by picking points from both C_i and C_j (resp. C_x and C_y).

Because of the assumption that all clusters are size $> 2\epsilon n$, for every 3-perturbation there must be a bijection between clusters and centers, where the center is closest to the majority of points in the corresponding cluster. We show that all size k subsets of the $k+2$ points cannot simultaneously admit bijections that are consistent with one another.

Note that Theorem 10 implies $(2-\delta, \epsilon)$- perturbation resilient k-center is hard for $\delta > 0$, even when the optimal clusters are large. Therefore, the value of α we achieve is within one of optimal. ◀

4.2 Lower bound on cluster sizes

Before moving to the asymmetric case, we show that the lower bound on the cluster sizes in Theorem 11 is necessary. Without this lower bound, clustering becomes hard, even assuming (α, ϵ)-perturbation resilience for any α and ϵ. This reduction follows from k-center (the details appear in the full version).

▶ **Theorem 12.** *For all $\alpha \geq 1$ and $\epsilon > 0$, finding the optimal solution for k-center under (α, ϵ)-perturbation resilience is NP-hard.*

4.3 Asymmetric k-center

In the asymmetric case, we consider the definition of the symmetric set A from Section 3, $A = \{p \mid \forall q, d(q,p) \leq r^* \implies d(p,q) \leq r^*\}$. We might first ask whether A respects the structure of $\mathcal{OPT}$, as it did under 2-perturbation resilience. Namely, whether *Condition 1:* all centers are in A, and *Condition 2:* $\arg\min_{q \in A} d(q,p) \in C_i \implies p \in C_i$ hold. This is not the case for either condition. We explore to what degree these conditions are violated.

We call a center c_i "bad" if it is not in the set A, i.e., $\exists q \notin C_i$ and $d(q, c_i) \leq r^*$. When a bad center c_i exists, we can take it out of the set of optimal centers, and we can pick an arbitrary dummy center which must be close to C_i or a CCC for C_i. In our symmetric argument, we arrived at a contradiction by showing that two dummy centers which capture the same cluster, must be close by the triangle inequality. This logic breaks down for asymmetric distances. In the full version of the paper, we show an example of an instance with a bad center that satisfies (α, ϵ)-perturbation resilience. However, it turns out that *no* instance can have more than 6 bad centers under $(3, \epsilon)$-perturbation resilience, assuming all optimal clusters have size $> 2\epsilon n$. So Condition 1 is satisfied for all but a constant number of centers. However, Condition 2 may not be satisfied for up to ϵn points. Therefore, even if we fully cluster A, we will only get ϵ-close to $\mathcal{OPT}$.

Although up to 6 clusters may have no intersection with A, each point that does belong to A is distance r^* from its center and distance $2r^*$ from its entire cluster. This motivates the following algorithm. First, we run a symmetric k-center 2-approximation algorithm on A, for $k-6 \leq k' \leq k$. For instance, iteratively pick an unmarked point, and mark all points distance $2r^*$ away from it [19]. This gives us a 2-approximation for the centers in A, and thus a 3-approximation for S minus the clusters with no centers in A. Then we brute force search for the remaining ≤ 6 centers to find a 3-approximation for S. Under $(3,\epsilon)$-perturbation resilience, this 3-approximation must be ϵ-close to $\mathcal{OPT}$. We formally state the algorithm and theorem below, and we defer the proof to the full version of this paper. The main technical challenge is in proving that no instance can have more than 6 bad centers.

Algorithm 2 $(3,\epsilon)$-Perturbation Resilient Asymmetric k-center

Input: Asymmetric k-center instance (S,d), r^* (or try all possible candidates).

1. Build set $A = \{p \mid \forall q, d(q,p) \leq r^* \implies d(p,q) \leq r^*\}$.
2. Create the threshold graph G with vertices A, and threshold distance r^*. Define a new symmetric k-center instance with A, using the lengths of the paths in the threshold graph.
3. Run a symmetric k-center 2-approximation algorithm on the symmetrized instance. Start with $k' = k-6$, and increase k' by 1 until the algorithm returns a solution with radius $\leq 2r^*$.
4. Brute force over all size $k-x$ subsets of C and all size x subsets of S for $x \leq 6$, to find a set of size k which is $3r^*$ from all points in S. Denote this set by C'.

Output: Output the Voronoi tiling $G_1, \ldots, G_k$ using C' as the centers.

▶ **Theorem 13.** *Algorithm 2 runs in polytime and outputs a clustering that is ϵ-close to $\mathcal{OPT}$, for $(3,\epsilon)$-perturbation resilient asymmetric k-center instances s.t. all optimal clusters are size $> 2\epsilon n$.*

5 Weak center proximity

In this section, we consider any center-based objective, not just k-center. A clustering objective function is *center-based* if the solution can be defined by choosing a set of centers $\{c_1, c_2, \ldots, c_k\} \subseteq S$, and partitioning S into k clusters $\mathcal{OPT} = \{C_1, C_2, \ldots, C_k\}$ by assigning each point to its closest center. Furthermore:

1. The objective value of a given clustering is a weighted sum or maximum of the individual cluster scores.
2. Given a proposed single cluster, its score can be computed in polynomial time.

k-median, k-means, and k-center are all center-based objectives.

Here, we show a novel algorithm that finds the optimal clustering in instances that satisfy two simple properties: each point is closer to its center than to any point in a different cluster, and we can recognize optimal clusters as soon as they are formed. Formally, we define these properties as:

1. **Weak Center Proximity:** For all $p \in C_i$ and $q \in C_j$, $d(c_i,p) < d(p,q)$.
2. **Cluster Verifiability:** There exists a polytime computable function $f : 2^S \to \mathbb{R}$ that for $B \subseteq S$, if there is $i \in [k]$ such that $B \subset C_i$, then $f(B) < 0$, and if $B \supseteq C_i$, then $f(B) \geq 0$.

Examples of cluster verifiable instances include any instance where all the optimal clusters are the same size ($f(B) = |B| - \frac{n}{k}$), or where all the optimal clusters have the same k-median/k-means cost ($f(B) = \Phi(B) - \Phi(\mathcal{OPT})$).

For any center-based objective, weak center proximity is a consequence of 2-perturbation resilience (i.e., Lemma 6), so, our algorithm relies on a much weaker assumption than α-perturbation resilience for $\alpha \geq 2$, when instances are cluster verifiable.

All existing algorithms and analysis for α-perturbation resilience require that for all $p \in C_i$ and $q \in C_j$, $d(c_i, p) < d(c_i, q)$. It is not at all obvious how one can even proceed without such a property, as in its absence, clusters can 'overlap'. That is, for a cluster with center c_i and radius r, we can not assume that $B_r(c_i)$ only includes points from C_i. Our challenge is then in showing that even in absence of this property, there is still enough structure imposed by the weak center proximity and cluster verifiability to find the optimal clustering efficiently.

Our Algorithm 3 is a novel linkage based procedure. Given a clustering instance (S, d), we will start with a graph $G = (S, E)$ where $E = \emptyset$. In each round, we do single linkage on the components in G, except we do not merge two components if both are supersets of optimal clusters (indicated by $f(B) \geq 0$). Put the single linkage edges from this round in a set A. This will continue until every component is a superset of an optimal cluster. Then we throw away the set A except for the *very last edge* that was added. We will prove this last edge is never between two points from different clusters, so we add that single edge to E and then recur. Here, we present a proof sketch of our main theorem. The details can be found in the full version.

Algorithm 3 Clustering under weak center proximity and cluster verifiability

Input: Clustering instance (S, d), function f, and $k \leq |S|$.

Set $G = (S, E)$ and $E = \emptyset$. While there are more than k components in G, repeat (1) and (2):

1. Set $A = \emptyset$. While there exists a component B in $G' = (S, E \cup A)$ such that $f(B) < 0$, add (p, q) to A, where $d(p, q)$ is minimized such that p and q are in different components in G' and at least one of these components B has $f(B) < 0$.
2. Take the last edge e that was added to A, and put $e \in E$.

Output: Output the components of G.

▶ **Theorem 14.** *Given a center-based clustering instance satisfying weak center proximity and cluster verifiability, Algorithm 3 outputs $\mathcal{OPT}$ in polynomial time.*

Proof Sketch. It suffices to show that step (b) never adds an edge between two points from different clusters. We proceed by induction. Assume it is true up to iteration t of the first while loop. Now assume towards contradiction that in round t, the last edge added to A is between two points $p \in C_i$ and $q \in C_j$, $i \neq j$. WLOG, for the component in G' that includes p, called P', we have $f(P') < 0$, otherwise the merge would not have happened. Furthermore, $c_i \in P'$ by weak center proximity. Then $f(P') < 0$ implies that $C_i \setminus P'$ is nonempty, so call it P. The component(s) in G corresponding to P are strict subsets of C_i, therefore, $f(P) < 0$. So they must merge to another component, and by weak center proximity, the closest component is P', but this contradicts our assumption that (p, q) was the last edge added to A. ◀

6 Conclusions

Our work pushes the understanding of (promise) stability conditions farther in three ways. We are the first to design computationally efficient algorithms to find the optimal clustering under α-perturbation resilience with a constant value of α for a problem that is hard to

approximate to any constant factor in the worst case, thereby demonstrating the power of perturbation resilience. Furthermore, we demonstrate the limits of this power by showing the first tight results in this space for both perturbation resilience and approximation stability. Finally, we show a surprising relation between symmetric and asymmetric instances, in that they are equivalent under resilience to 2-perturbations, which is in stark contrast to their widely differing tight approximation factors.

References

1 Aaron Archer. Two o (log* k)-approximation algorithms for the asymmetric k-center problem. In *Integer Programming and Combinatorial Optimization*, pages 1–14. Springer, 2001.

2 Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models – going beyond SVD. In *53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, 2012.

3 Pranjal Awasthi, Afonso S. Bandeira, Moses Charikar, Ravishankar Krishnaswamy, Soledad Villar, and Rachel Ward. Relax, no need to round: Integrality of clustering formulations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 191–200, 2015. doi:10.1145/2688073.2688116.

4 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a ptas for k-median and k-means clustering. In *51st Annual IEEE Symposium on Foundations of Computer Science*, pages 309–318, 2010.

5 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1):49–54, 2012.

6 Maria-Florina Balcan, Avrim Blum, and Anupam Gupta. Approximate clustering without the approximation. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1068–1077, 2009.

7 Maria-Florina Balcan and Mark Braverman. Approximate nash equilibria under stability conditions. Technical report, 2010.

8 Maria-Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. In *Automata, Languages, and Programming*, pages 63–74. Springer, 2012.

9 Shalev Ben-David and Lev Reyzin. Data stability in clustering: A closer look. In *Algorithmic Learning Theory*, pages 184–198. Springer, 2012.

10 Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(05):643–660, 2012.

11 Fazli Can. Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems (TOIS)*, 11(2):143–164, 1993.

12 Fazli Can and ND Drochak. Incremental clustering for dynamic document databases. In *Proceedings of the 1990 Symposium on Applied Computing*, pages 61–67, 1990.

13 Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 626–635, 1997.

14 Julia Chuzhoy, Sudipto Guha, Eran Halperin, Sanjeev Khanna, Guy Kortsarz, Robert Krauthgamer, and Joseph Seffi Naor. Asymmetric k-center is log* n-hard to approximate. *Journal of the ACM (JACM)*, 52(4):538–551, 2005.

15 Irit Dinur, Venkatesan Guruswami, Subhash Khot, and Oded Regev. A new multilayered pcp and the hardness of hypergraph vertex cover. *SIAM Journal on Computing*, 34(5):1129–1146, 2005.

16 Martin E Dyer and Alan M Frieze. A simple heuristic for the p-centre problem. *Operations Research Letters*, 3(6):285–288, 1985.

17 Rishi Gupta, Tim Roughgarden, and C Seshadhri. Decompositions of triangle-dense graphs. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pages 471–482. ACM, 2014.

18 Moritz Hardt and Aaron Roth. Beyond worst-case analysis in private singular vector computation. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 331–340, 2013.

19 Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985.

20 Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *51st Annual IEEE Symposium on Foundations of Computer Science*, pages 299–308, 2010.

21 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time (1+ ε)-approximation algorithm for geometric k-means clustering in any dimensions. In *Proceedings-Annual Symposium on Foundations of Computer Science*, pages 454–462. IEEE, 2004.

22 Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Bilu-linial stable instances of max cut and minimum multiway cut. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 890–906. SIAM, 2014.

23 Matúš Mihalák, Marcel Schöngens, Rastislav Šrámek, and Peter Widmayer. On the complexity of the metric tsp under stability considerations. In *SOFSEM 2011: Theory and Practice of Computer Science*, pages 382–393. Springer, 2011.

24 Tim Roughgarden. Beyond worst-case analysis, 2014. URL: `http://theory.stanford.edu/~tim/f14/f14.html`.

25 Sundar Vishwanathan. An o(log*n) approximation algorithm for the asymmetric p-center problem. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–5, 1996. URL: `http://dl.acm.org/citation.cfm?id=313852.313861`.

Approximation Algorithms for Clustering Problems with Lower Bounds and Outliers*

Sara Ahmadian[1] **and Chaitanya Swamy**†[2]

1 **Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON, Canada**
sahmadian@math.uwaterloo.ca

2 **Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON, Canada**
sahmadian@math.uwaterloo.ca

Abstract

We consider clustering problems with *non-uniform lower bounds and outliers*, and obtain the *first approximation guarantees* for these problems. We have a set $\mathcal{F}$ of facilities with lower bounds $\{L_i\}_{i\in\mathcal{F}}$ and a set $\mathcal{D}$ of clients located in a common metric space $\{c(i,j)\}_{i,j\in\mathcal{F}\cup\mathcal{D}}$, and bounds k, m. A feasible solution is a pair $\big(S\subseteq\mathcal{F}, \sigma:\mathcal{D}\mapsto S\cup\{\mathsf{out}\}\big)$, where σ specifies the client assignments, such that $|S|\leq k$, $|\sigma^{-1}(i)|\geq L_i$ for all $i\in S$, and $|\sigma^{-1}(\mathsf{out})|\leq m$. In the *lower-bounded min-sum-of-radii with outliers* (LBkSRO) problem, the objective is to minimize $\sum_{i\in S}\max_{j\in\sigma^{-1}(i)}c(i,j)$, and in the *lower-bounded k-supplier with outliers* (LBkSupO) problem, the objective is to minimize $\max_{i\in S}\max_{j\in\sigma^{-1}(i)}c(i,j)$.

We obtain an approximation factor of 12.365 for LBkSRO, which improves to 3.83 for the non-outlier version (i.e., $m=0$). These also constitute the *first* approximation bounds for the min-sum-of-radii objective when we consider lower bounds and outliers *separately*. We apply the primal-dual method to the relaxation where we Lagrangify the $|S|\leq k$ constraint. The chief technical contribution and novelty of our algorithm is that, departing from the standard paradigm used for such constrained problems, we obtain an $O(1)$-approximation *despite the fact that we do not obtain a Lagrangian-multiplier-preserving algorithm for the Lagrangian relaxation.* We believe that our ideas have broader applicability to other clustering problems with outliers as well.

We obtain approximation factors of 5 and 3 respectively for LBkSupO and its non-outlier version. These are the *first* approximation results for k-supplier with *non-uniform* lower bounds.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.1.6 Optimization, G.2 Discrete Mathematics

Keywords and phrases Approximation algorithms, facililty-location problems, primal-dual method, Lagrangian relaxation, k-center problems, minimizing sum of radii

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.69

1 Introduction

Clustering is an ubiquitous problem arising in applications in various fields such as data mining, machine learning, image processing, and bioinformatics. Many of these problems involve finding a set S of at most k "cluster centers", and an assignment σ mapping an underlying set $\mathcal{D}$ of data points located in some metric space $\{c(i,j)\}$ to S, to minimize

* A full version of the paper is available at http://arxiv.org/abs/1608.01700.

† This work was supported in part by the second author's NSERC grant 327620-09 and NSERC Discovery Accelerator Supplement Award.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 69; pp. 69:1–69:15

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

some objective function; examples include the *k-center* (minimize $\max_{j\in\mathcal{D}} c(\sigma(j),j)$) [20, 21], *k-median* (minimize $\sum_{j\in\mathcal{D}} c(\sigma(j),j)$) [9, 22, 25, 6], and *min-sum-of-radii* (minimize $\sum_{i\in S}\max_{j:\sigma(j)=i} c(i,j)$) [15, 11] problems. Viewed from this perspective, clustering problems can often be viewed as *facility-location* problems, wherein an underlying set of clients that require service need to be assigned to facilities that provide service in a cost-effective fashion. Both clustering and facility-location problems have been extensively studied in the Computer Science and Operations Research literature; see, e.g., [27, 29] in addition to the above references.

We consider clustering problems with (non-uniform) *lower-bound requirements* on the cluster sizes, and where a bounded number of points may be designated as *outliers* and left unclustered. One motivation for considering lower bounds comes from an *anonymity* consideration. In order to achieve data privacy, [28] proposed an anonymization problem where we seek to perturb (in a specific way) some of (the attributes of) the data points and then cluster them so that every cluster has at least L identical perturbed data points, thus making it difficult to identify the original data from the clustering. As noted in [2, 1], this anonymization problem can be abstracted as a lower-bounded clustering problem where the clustering objective captures the cost of perturbing data. Another motivation comes from a facility-location perspective, where (as in the case of *lower-bounded facility location*), the lower bounds model that it is infeasible or unprofitable to use services unless they satisfy a certain minimum demand (see, e.g., [26]). Allowing outliers enables one to handle a common woe in clustering problems, namely that data points that are quite dissimilar from any other data point can often disproportionately (and undesirably) degrade the quality of *any* clustering of the *entire* data set; instead, the outlier-version allows one to designate such data points as outliers and focus on the data points of interest.

Formally, adopting the facility-location terminology, our setup is as follows. We have a set $\mathcal{F}$ of facilities with lower bounds $\{L_i\}_{i\in\mathcal{F}}$ and a set $\mathcal{D}$ of clients located in a common metric space $\{c(i,j)\}_{i,j\in\mathcal{F}\cup\mathcal{D}}$, and bounds k, m. A feasible solution chooses a set $S\subseteq\mathcal{F}$ of at most k facilities, and assigns each client j to a facility $\sigma(j)\in S$, or designates j as an outlier by setting $\sigma(j)=\mathsf{out}$ so that $|\sigma^{-1}(i)|\geq L_i$ for all $i\in S$, and $|\sigma^{-1}(\mathsf{out})|\leq m$. We consider two clustering objectives: minimize $\sum_{i\in S}\max_{j:\sigma(j)=i} c(i,j)$, which yields the *lower-bounded min-sum-of-radii with outliers* (LBkSRO) problem, and minimize $\max_{i\in S}\max_{j:\sigma(j)=i} c(i,j)$, which yields the *lower-bounded k-supplier with outliers* (LBkSupO) problem. We refer to the non-outlier versions of the above problems (i.e., where $m=0$) as LBkSR and LBkSup respectively.

Our contributions. We obtain the *first* results for clustering problems with *non-uniform lower bounds and outliers*. We develop various techniques for tackling these problems using which we obtain *constant-factor approximation guarantees* for LBkSRO and LBkSupO. Note that we need to ensure that none of the *hard* constraints involved here – at most k clusters, non-uniform lower bounds, and at most m outliers – are violated, which is somewhat challenging.

We obtain an approximation factor of 12.365 for LBkSRO (Theorem 7, Section 2.2), which improves to 3.83 for the non-outlier version LBkSR (Theorem 6, Section 2.1). These also constitute the *first* approximation results for the min-sum-of-radii objective when we consider: (a) lower bounds (even uniform bounds) but no outliers (LBkSR); and (b) outliers but no lower bounds. Previously, an $O(1)$-approximation was known only in the setting where there are *no lower bounds and no outliers* (i.e., $L_i=0$ for all i, $m=0$) [11].

For the k-supplier objective (Section 3), we obtain an approximation factor of 5 for LBkSupO (Theorem 16), and 3 for LBkSup (Theorem 15). These are the *first* approximation

results for the k-supplier problem with non-uniform lower bounds. Previously, [1] obtained approximation factors of 4 and 2 respectively for LBkSupO and LBkSup for the special case of *uniform* lower bounds and when $\mathcal{F} = \mathcal{D}$ (often called the k-center version). Complementing our approximation bounds, we prove a factor-3 hardness of approximation for LBkSup (Theorem 17), which shows that our approximation factor of 3 is optimal for LBkSup.

Our techniques. Our main technical contribution is an $O(1)$-approximation algorithm for LBkSRO (Section 2.2). Whereas for the non-outlier version LBkSR (Section 2.1), one can follow an approach similar to that in [11] for the min-sum-of-radii problem without lower bounds or outliers, the presence of outliers creates substantial difficulties whose resolution requires various novel ingredients. As in [11], we view LBkSRO as a *k-ball-selection* (k-BS) problem of picking k suitable balls (see Section 2) and consider its LP-relaxation (P_2). Let OPT denote its optimal value. Following the Jain-Vazirani (JV) template for k-median [22], we move to the version where we may pick any number of balls but incur a fixed cost of z for each ball we pick. The dual LP (D_2) has α_j dual variables for the clients, which "pay" for (i, r) pairs (where (i, r) denotes the ball $\{j \in \mathcal{D} : c(i, j) \leq r\}$). For LB$k$SR (where $m = 0$), as observed in [11], it is easy to adapt the JV primal-dual algorithm for facility location to handle this fixed-cost version of k-BS: we raise the α_js of uncovered clients until all clients are covered by some fully-paid (i, r) pair (see PDAlg). This yields a so-called *Lagrangian-multiplier-preserving* (LMP) 3-approximation algorithm: if F is the primal solution constructed, then $3\sum_j \alpha_j$ can pay for $cost(F) + 3|F|z$; hence, by varying z, one can find two solutions F_1, F_2 for nearby values of z, and combine them to extract a low-cost k-BS-solution.

The presence of outliers in LBkSRO significantly complicates things. The natural adaptation of the primal-dual algorithm is to now stop when at least $|\mathcal{D}| - m$ clients are covered by fully-paid (i, r) pairs. But now, the dual objective involves a $-m \cdot \gamma$ term, where $\gamma = \max_j \alpha_j$, which potentially cancels the dual contribution of (some) clients that pay for the last fully-paid (i, r) pair, say f. Consequently, we *do not obtain an LMP-approximation*: if F is the primal solution we construct, we can only say that (roughly) $3(\sum_j \alpha_j - m \cdot \gamma)$ pays for $cost(F \setminus f) + 3|F \setminus f|z$ (see Theorem 8 (ii)). In particular, this means that *even if the primal-dual algorithm returns a solution with k pairs, its cost need not be bounded*, an artifact that never arises in LBkSR (or k-median). This in turn means that by combining the two solutions F_1, F_2 found for $z_1, z_2 \approx z_1$, we only obtain a solution of cost $O(OPT + z_1)$ (see Theorem 10).

Dealing with the case where $z_1 = \Omega(OPT)$ is technically the most involved portion of our algorithm (Section 2.2.2). We argue that in this case the solutions F_1, F_2 (may be assumed to) have a very specific structure: $|F_1| = k + 1$, and every F_2-ball intersects at most one F_1-ball, and vice versa. We utilize this structure to show that either we can find a good solution in a suitable neighborhood of F_1 and F_2, or F_2 itself must be a good solution.

We remark that the above difficulties (i.e., the inability to pay for the last "facility" and the ensuing complications) also arise in the *k-median problem with outliers*. We believe that our ideas also have implications for this problem and should yield a much-improved approximation ratio for this problem. (The current guarantee is a large (unspecified) constant [12].)

For the k-supplier problem, LBkSupO, we leverage the notion of skeletons and pre-skeletons defined by [14] in the context of *capacitated k-supplier with outliers*, wherein facilities have capacities instead of lower bounds limiting the number of clients that can be assigned to them. Roughly speaking, a skeleton $F \subseteq \mathcal{F}$ ensures there is a low-cost solution (F, σ). A pre-skeleton satisfies some of the properties of a skeleton. We show that if F is a pre-skeleton,

then either F is a skeleton or $F \cup \{i\}$ is a pre-skeleton for some facility i. This allows one to find a sequence of facility-sets such that at least one of them is a skeleton. For a given set F, one can check if F admits a low-cost assignment σ, so this yields an $O(1)$-approximation algorithm.

Related work. There is a vast literature on clustering and facility-location (FL) problems (see, e.g., [27, 29]); we limit ourselves to work that is relevant to LBkSRO and LBkSupO.

The only prior work on clustering problems to incorporate both lower bounds *and* outliers is by Aggarwal et al. [1]. They obtain approximation ratios of 4 and 2 respectively for LBkSupO and LBkSup with *uniform* lower bounds, which they consider as a means of achieving anonymity. They also consider an alternate *cellular clustering* (CellC) objective and devise an $O(1)$-approximation algorithm for lower-bounded CellC with uniform lower bounds, and mention that this can be extended to an $O(1)$-approximation for lower-bounded CellC with outliers.

More work has been directed towards clustering problems involving outliers *or* lower bounds (but not both). Charikar et al. [10] consider (among other problems) the outlier-versions of the uncapacitated FL, k-supplier and k-median problems. They devise constant-factor approximations for the first two problems, and a bicriteria approximation for the k-median problem with outliers. They also proved a factor-3 approximation hardness result for k-supplier with outliers. This nicely complements our factor-3 hardness result for k-supplier with lower bounds but no outliers. Chen [12] obtained the first true approximation for k-median with outliers via a sophisticated combination of the primal-dual algorithm for k-median and local search that yields a large (unspecified) $O(1)$-approximation. Cygan and Kociumaka [14] consider the *capacitated k-supplier with outliers* problem, and devise a 25-approximation algorithm. We leverage some of their ideas in developing our algorithm for LBkSupO.

Lower-bounded clustering and FL problems remain largely unexplored and ill-understood. Besides LBkSup (which has also been studied in Euclidean spaces [16]) another such FL problem that has been studied is *lower-bounded facility location* (LBFL) [23, 19], wherein we seek to open facilities (which have lower bounds) and assign each client j to an open facility $\sigma(j)$ so as to minimize $\sum_{j \in \mathcal{D}} c(\sigma(j), j)$. Svitkina [30] obtained the first true approximation for LBFL, achieving an $O(1)$-approximation; the $O(1)$-factor was subsequently improved by [3]. Both results apply to LBFL with uniform lower bounds, and can be adapted to yield $O(1)$-approximations to the k-median variant (where we may open at most k facilities).

Doddi et al. [15] introduced the min-sum-of-diameters objective, which is closely related to the min-sum-of-radii objective (the former is at most twice the latter). Charikar and Panigrahi [11] devised the first (and current-best) $O(1)$-approximation algorithms for these problems, obtaining approximation ratios of 3.53 and 7.06 for the radii and diameter problems respectively. Various other results are known for specific metric spaces and when $\mathcal{F} = \mathcal{D}$, such as Euclidean spaces [18, 7] and metrics with bounded aspect ratios [17, 5].

The k-supplier and k-center (i.e., k-supplier with $\mathcal{F} = \mathcal{D}$) objectives have a rich history of study. Hochbaum and Shmoys [20, 21] obtained optimal approximation ratios of 3 and 2 for these problems respectively. Capacitated versions of k-center and k-supplier have also been studied: [24] devised a 6-approximation for uniform capacities, [13] obtained the first $O(1)$-approximation for non-uniform capacities, and this $O(1)$-factor was improved to 9 in [4].

Finally, our algorithm for LBkSRO leverages the template based on Lagrangian relaxation and the primal-dual method to emerge from the work of [22, 8] for the k-median problem.

2 Minimizing sum of radii with lower bounds and outliers

Recall that in the *lower-bounded min-sum-of-radii with outliers* (LBkSRO) problem, we have a facility-set $\mathcal{F}$ and client-set $\mathcal{D}$ located in a metric space $\{c(i,j)\}_{i,j\in\mathcal{F}\cup\mathcal{D}}$, lower bounds $\{L_i\}_{i\in\mathcal{F}}$, and bounds k and m. A feasible solution is a pair $\big(S\subseteq\mathcal{F}, \sigma:\mathcal{D}\mapsto S\cup\{\mathsf{out}\}\big)$, where $\sigma(j)\in S$ indicates that j is assigned to facility $\sigma(j)$, and $\sigma(j)=\mathsf{out}$ designates j as an outlier, such that $|S|\leq k$, $|\sigma^{-1}(i)|\geq L_i$ for all $i\in S$, and $|\sigma^{-1}(\mathsf{out})|\leq m$. The cost $cost(S,\sigma)$ of such a solution is $\sum_{i\in S} r_i$, where $r_i := \max_{j\in\sigma^{-1}(i)} c(i,j)$ denotes the *radius* of facility i; the goal is to find a solution of minimum cost. We use LBkSR to denote the non-outlier version where $m=0$.

It will be convenient to consider a relaxation of LBkSRO that we call the *k-ball-selection* (k-BS) problem, which focuses on selecting at most k balls centered at facilities of minimum total radius. More precisely, let $B(i,r) := \{j\in\mathcal{D} : c(i,j)\leq r\}$ denote the ball of clients centered at i with radius r. Let $c_{\max} = \max_{i\in\mathcal{F},j\in\mathcal{D}} c(i,j)$. Let $\mathcal{L}_i := \{(i,r) : |B(i,r)|\geq L_i\}$, and $\mathcal{L} := \bigcup_{i\in\mathcal{F}}\mathcal{L}_i$. The goal in k-BS is to find a set $F\subseteq\mathcal{L}$ with $|F|\leq k$ and $\big|\mathcal{D}\setminus\bigcup_{(i,r)\in F} B(i,r)\big|\leq m$ so that $cost(F) := \sum_{(i,r)\in F} r$ is minimized. (When formulating the LP-relaxation of the k-BS-problem, we equivalently view $\mathcal{L}$ as containing only pairs of the form $(i,c(i,j))$ for some client j, which makes $\mathcal{L}$ finite.) It is easy to see that any LBkSRO-solution yields a k-BS-solution of no greater cost. The key advantage of working with k-BS is that we do not explicitly consider the lower bounds (they are folded into the $\mathcal{L}_i$s) and we do not require the balls $B(i,r)$ for $(i,r)\in F$ to be disjoint. While a k-BS-solution F need not directly translate to a feasible LBkSRO-solution, one can show that it does yield a feasible LBkSRO-solution of cost at most $2\cdot cost(F)$. We prove a stronger version of this statement in Lemma 1. In the following two sections, we utilize this relaxation to devise the *first* constant-factor approximation algorithms for for LBkSR and LBkSRO. To our knowledge, our algorithm is also the first $O(1)$-approximation algorithm for the outlier version of the min-sum-of-radii problem *without* lower bounds.

We consider an LP-relaxation for the k-BS-problem, and to round a fractional k-BS-solution to a good integral solution, we need to preclude radii that are much larger than those used by an optimal solution. We therefore "guess" the t facilities in the optimal solution with the largest radii, and their radii, where $t\geq 1$ is some constant. That is, we enumerate over all $O\big((|\mathcal{F}|+|\mathcal{D}|)^{2t}\big)$ choices $F^O=\{(i_1,r_1),\ldots,(i_t,r_t)\}$ of t (i,r) pairs from $\mathcal{L}$. For each such selection, we set $\mathcal{D}' = \mathcal{D}\setminus\bigcup_{(i,r)\in F^O} B(i,r)$, $\mathcal{L}' = \{(i,r)\in\mathcal{L} : r\leq \min_{p=1,\ldots,t} r_p\}$ and $k' = k-|F^O|$, and run our k-BS-algorithm on the modified k-BS-instance $(\mathcal{F},\mathcal{D}',c,\mathcal{L}',k',m)$ to obtain a k-BS-solution F. We translate $F\cup F^O$ to an LBkSRO-solution, and return the best of these solutions. The following lemma, and the procedure described therein, is repeatedly used to bound the cost of translating $F\cup F^O$ to a feasible LBkSRO-solution. We call pairs $(i,r),(i',r')\in\mathcal{F}\times\mathbb{R}_{\geq 0}$ *non-intersecting*, if $c(i,i')>r+r'$, and *intersecting* otherwise. Note that $B(i,r)\cap B(i',r')=\emptyset$ if (i,r) and (i',r') are non-intersecting. For a set $P\subseteq\mathcal{F}\times\mathbb{R}_{\geq 0}$ of pairs, define $\mu(P) := \{i\in\mathcal{F} : \exists r \text{ s.t. } (i,r)\in P\}$.

▶ **Lemma 1.** *Let $F^O\subseteq\mathcal{L}$, and $\mathcal{D}',\mathcal{L}',k'$ be as defined above. Let $F\subseteq\mathcal{L}$ be a k-BS-solution for the k-BS-instance $(\mathcal{F},\mathcal{D}',c,\mathcal{L}',k',m)$. Suppose for each $i\in\mu(F)$, we have a radius $r'_i\leq\max_{r:(i,r)\in F} r$ such that the pairs in $U := \bigcup_{i\in\mu(F)}(i,r'_i)$ are non-intersecting and $U\subseteq\mathcal{L}'$. Then there exists a feasible LBkSRO-solution (S,σ) with $cost(S,\sigma)\leq cost(F)+\sum_{(i,r)\in F^O} 2r$.*

2.1 Approximation algorithm for LBkSR

We now present our algorithm for the non-outlier version, LBkSR, which introduces many of the ideas underlying our algorithm for LBkSRO (Section 2.2). Let O^* be the cost of an optimal

solution to the given LBkSR instance. For each selection $(i_1, r_1), \ldots, (i_t, r_t)$ of t pairs, we do the following. We set $\mathcal{D}' = \mathcal{D} \setminus \bigcup_{p=1}^{t} B(i_p, r_p)$, $\mathcal{L}' = \{(i,r) \in \mathcal{L} : r \leq R^* := \min_{p=1,\ldots,t} r_p\}$, $k' = k - t$, and consider the k-BS-problem of picking a min-cost set of at most k' pairs from $\mathcal{L}'$ whose corresponding balls cover $\mathcal{D}'$ (but our algorithm k-BSAlg will return pairs from $\mathcal{L}$). Consider the following natural LP-relaxation (P_1) of this problem, and its dual (D_1).

$$\begin{array}{lll} \min & \sum_{(i,r)\in\mathcal{L}'} r \cdot y_{i,r} & (\mathrm{P}_1) \\ \text{s.t.} & \sum_{(i,r)\in\mathcal{L}': j\in B(i,r)} y_{i,r} \geq 1 & \forall j \in \mathcal{D}' \\ & \sum_{(i,r)\in\mathcal{L}'} y_{i,r} \leq k' \quad (1) & \\ & y \geq 0. & \end{array} \qquad \begin{array}{lll} \max & \sum_{j\in\mathcal{D}'} \alpha_j - k' \cdot z & (\mathrm{D}_1) \\ \text{s.t.} & \sum_{j\in B(i,r)\cap\mathcal{D}'} \alpha_j - z \leq r & \forall (i,r) \in \mathcal{L}' \\ & \alpha, z \geq 0. & \end{array} \tag{2}$$

Let OPT denote the common optimal value of (P_1) and (D_1). As in the JV-algorithm for k-median, we Lagrangify constraint (1) and consider the unconstrained problem where we do not bound the number of pairs we may pick, but we incur a fixed cost z for each pair (i,r) that we pick (in addition to r). It is easy to adapt the JV primal-dual algorithm for facility location [22] to devise a simple *Lagrangian-multiplier-preserving* (LMP) 3-approximation algorithm for this problem (see PDAlg and Theorem 3). We use this LMP algorithm within a binary-search procedure for z to obtain two solutions F_1 and F_2 with $|F_2| \leq k' < |F_1|$, and show that these can be "combined" to extract a k-BS-solution F of cost at most $3.83 \cdot OPT + O(R^*)$. This combination step is more involved than in k-median. The main idea here is to use the F_2 solution as a guide to merge some F_1-pairs. We cluster the F_1 pairs around the F_2-pairs and setup a *covering-knapsack problem* whose solution determines for each F_2-pair (i,r), whether to "merge" the F_1-pairs clustered around (i,r) or select all these F_1-pairs (see step B2). Finally, we add back the pairs $(i_1, r_1), \ldots (i_t, r_t)$ selected earlier and apply Lemma 1 to obtain an LBkSR-solution. As required by Lemma 1, to aid in this translation, our k-BS-algorithm returns, along with F, a suitable radius $\mathsf{rad}(i)$ for every facility $i \in \mu(F)$. This yields a $(3.83 + \epsilon)$-approximation algorithm (Theorem 6).

While our approach is similar to the one in [11] for the min-sum-of-radii problem *without* lower bounds (although our combination step is notably simpler), an important distinction that arises is the following. In the absence of lower bounds, the ball-selection problem k-BS is *equivalent* to the min-sum-of-radii problem, but (as noted earlier) this is no longer the case when we have lower bounds since in k-BS we do not insist that the balls we pick be disjoint. Moving from overlapping balls in a k-BS-solution to an LBkSR-solution incurs, in general, a factor-2 blowup in the cost, but we avoid this blowup by exploiting the structure of the k-BS-solution obtained and carefully merging in the pairs $(i_1, r_1), \ldots, (i_t, r_t)$ (see Lemma 1). It is interesting that our approximation factor is quite close to the approximation factor (of 3.53) achieved in [11] for the min-sum-of-radii problem without lower bounds.

We now describe our algorithm in detail and analyze it. We describe a slightly simpler $(6.183 + \epsilon)$-approximation algorithm below (Theorem 2). We sketch the ideas behind the improved approximation ratio at the end of this section and defer the details to the full version.

▶ Algorithm 1.
Input: An LBkSR-instance $\mathcal{I} = \big(\mathcal{F}, \mathcal{D}, \{L_i\}, \{c(i,j)\}, k\big)$, parameter $\epsilon > 0$.
Output: A feasible solution (S, σ).

A1. Let $t = \min\big\{k, \lceil \frac{1}{\epsilon} \rceil\big\}$. For each set $F^O \subseteq \mathcal{L}$ with $|F^O| \leq t$, do the following.

A1.1. Set $\mathcal{D}' = \mathcal{D} \setminus \bigcup_{(i,r)\in F^O} B(i,r)$, $\mathcal{L}' = \{(i',r') \in \mathcal{L} : r \leq R^* = \min_{(i,r)\in F^O} r\}$, $k' = k - |F^O|$.

A1.2. If (P_1) is infeasible, then reject this guess and move to the next set F^O. If $\mathcal{D}' \neq \emptyset$, run k-BSAlg$(\mathcal{D}', \mathcal{L}', k', \epsilon)$ to obtain $\big(F, \{\mathsf{rad}(i)\}_{i\in F}\big)$; else set $(F, \mathsf{rad}) = (\emptyset, \emptyset)$.

A1.3. Apply the procedure in Lemma 1 taking $r'_i = \mathsf{rad}(i)$ for all $i \in \mu(F)$ to obtain (S, σ).

A1. Among all the solutions (S, σ) found in step A1, return the one with smallest cost.

▶ Algorithm k-**BSAlg**$(\mathcal{D}', \mathcal{L}', k', \epsilon)$.
Output: $F \subseteq \mathcal{L}$ with $|F| \leq k'$, a radius $\mathsf{rad}(i)$ for all $i \in \mu(F)$.

B1. Binary search for z.

B1.1. Set $z_1 = 0$ and $z_2 = 2k' c_{\max}$. For $p = 1, 2$, let $(F_p, \{\mathsf{rad}_p(i)\}, \alpha^p) \leftarrow$ PDAlg$(\mathcal{D}', \mathcal{L}', z_p)$, and let $k_p = |F_p|$. If $k_1 \leq k'$, stop and return $\big(F_1, \{\mathsf{rad}_1(i)\}\big)$. We prove in Theorem 3 that $k_2 \leq k'$; if $k_2 = k'$, stop and return $\big(F_2, \{\mathsf{rad}_2(i)\}\big)$.

B1.2. Repeat the following until $z_2 - z_1 \leq \delta_z = \frac{\epsilon OPT}{3n}$, where $n = |\mathcal{F}| + |\mathcal{D}|$. Set $z = \frac{z_1+z_2}{2}$. Let $(F, \{\mathsf{rad}(i)\}, \alpha) \leftarrow$ PDAlg$(\mathcal{D}', \mathcal{L}', z)$. If $|F| = k'$, stop and return $\big(F, \{\mathsf{rad}(i)\}\big)$; if $|F| > k'$, update $z_1 \leftarrow z$ and $(F_1, \mathsf{rad}_1, \alpha^1) \leftarrow (F, \mathsf{rad}, \alpha)$, else update $z_2 \leftarrow z$ and $(F_2, \mathsf{rad}_2, \alpha^2) \leftarrow (F, \mathsf{rad}, \alpha)$.

B2. Combining F_1 and F_2. Let $\pi : F_1 \mapsto F_2$ be any map such that (i',r') and $\pi(i',r')$ intersect $\forall (i',r') \in F_1$. (This exists since every $j \in \mathcal{D}'$ is covered by $B(i,r)$ for some $(i,r) \in F_2$.) Define star $\mathcal{S}_{i,r} = \pi^{-1}(i,r)$ for all $(i,r) \in F_2$ (see Fig. 1). Solve the following *covering-knapsack LP*.

$$\min \quad \sum_{(i,r)\in F_2} \Big(x_{i,r}\big(2r + \textstyle\sum_{(i',r')\in\mathcal{S}_{i,r}} 2r'\big) + (1 - x_{i,r}) \sum_{(i',r')\in\mathcal{S}_{i,r}} r' \Big) \tag{C-P}$$

$$\text{s.t.} \quad \sum_{(i,r)\in F_2} \big(x_{i,r} + |\mathcal{S}_{i,r}|(1 - x_{i,r}) \big) \leq k', \qquad 0 \leq x_{i,r} \leq 1 \quad \forall (i,r) \in F_2.$$

Let x^* be an extreme-point optimal solution to (C-P). The variable $x_{(i,r)}$ has the following interpretation. If $x^*_{i,r} = 0$, then we select all pairs in $\mathcal{S}_{i,r}$. Otherwise, if $\mathcal{S}_{i,r} \neq \emptyset$, we pick a pair in $(i',r') \in \mathcal{S}_{i,r}$, and include $(i', 2r + r' + \max_{(i'',r'')\in\mathcal{S}_{i,r}\setminus\{(i',r')\}} 2r'')$ in our solution. Notice that by expanding the radius of i' to $2r + r' + \max_{(i'',r'')\in\mathcal{S}_{i,r}\setminus\{(i',r')\}} 2r''$, we cover all the clients in $\bigcup_{(i'',r'')\in\mathcal{S}_{i,r}} B(i'',r'')$. Let F' be the resulting set of pairs.

B3. If $cost(F_2) \leq cost(F)$, return (F_2, rad_2), else return $\big(F', \{\mathsf{rad}_1(i)\}_{i\in\mu(F')}\big)$.

▶ Algorithm **PDAlg**$(\mathcal{D}', \mathcal{L}', z)$.
Output: $F \subseteq \mathcal{L}$, radius $\mathsf{rad}(i)$ for all $i \in \mu(F)$, dual solution α.

P1. Dual-ascent phase. Start with $\alpha_j = 0$ for all $j \in \mathcal{D}'$, $\mathcal{D}'$ as the set of *active clients*, and the set T of *tight pairs* initialized to $\emptyset$. We repeat the following until all clients become inactive: we raise the α_js of all active clients uniformly until constraint (2) becomes tight for some (i,r); we add (i,r) to T and mark all active clients in $B(i,r)$ as inactive.

P2. Pruning phase. Let T_I be a maximal subset of non-intersecting pairs in T picked by a greedy algorithm that scans pairs in T in non-increasing order of radius. Note that for each $i \in \mu(T_I)$, there is exactly one pair $(i,r) \in T_I$. We set $\mathsf{rad}(i) = r$, and $r_i = \max\{c(i,j) : j \in B(i',r'),\ (i',r') \in T,\ r' \leq r,\ (i',r')$ intersects (i,r) $((i',r')$ could be $(i,r))\}$. Let $F = \{(i, r_i)\}_{i\in\mu(T_I)}$. Return F, $\{\mathsf{rad}(i)\}_{i\in\mu(T_I)}$, and α.

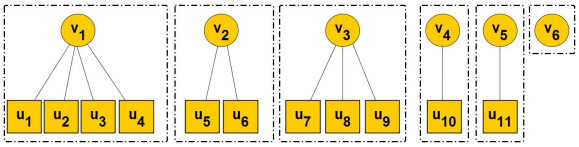


Figure 1 An example of stars formed by F_1 and F_2 where $F_1 = \{u_1, u_2, \ldots, u_{11}\}$ and $F_2 = \{v_1, v_2, \ldots, v_6\}$ depicted by squares and circles, respectively.

Analysis. We prove the following result.

▶ **Theorem 2.** *For any $\epsilon > 0$, Algorithm 1 returns a feasible* LBkSR*-solution of cost at most $\big(6.1821 + O(\epsilon)\big)O^*$ in time $n^{O(1/\epsilon)}$.*

We first prove that PDAlg is an LMP 3-approximation algorithm, i.e., its output (F, α) satisfies $cost(F) + 3|F|z \leq 3\sum_{j\in\mathcal{D}'} \alpha_j$. (Theorem 3). Utilizing this, we analyze k-BSAlg, in particular, the output of the combination step B2, and argue that k-BSAlg returns a feasible solution of cost at most $\big(6.183 + O(\epsilon)\big) \cdot OPT + O(R^*)$ (Theorem 5). For the right choice of F^O, combining this with Lemma 1 yields Theorem 2.

▶ **Theorem 3.** *Suppose* PDAlg$(\mathcal{D}', \mathcal{L}', z)$ *returns* $(F, \{\mathsf{rad}(i)\}, \alpha)$. *Then*

(i) *the balls corresponding to F cover $\mathcal{D}'$;*
(ii) $cost(F) + 3|F|z \leq 3\sum_{j\in\mathcal{D}'} \alpha_j \leq 3(OPT + k'z)$;
(iii) $\big\{(i, \mathsf{rad}(i))\big\}_{i\in\mu(F)} \subseteq \mathcal{L}'$, *is a set of non-intersecting pairs,* $\mathsf{rad}(i) \leq r_i \leq 3R^*\ \forall i \in \mu(F)$;
(iv) *if $|F| \geq k'$ then $cost(F) \leq 3 \cdot OPT$; if $|F| > k'$, then $z \leq OPT$. (Hence, $k_2 \leq k'$ in step B1.1.)*

Let $\big(F, \{\mathsf{rad}(i)\}\big) = k$-BSAlg$(\mathcal{D}', \mathcal{L}', k', \epsilon)$. If k-BSAlg terminates in step B1, then $cost(F) \leq 3 \cdot OPT$ due to part (ii) of Theorem 3, so assume otherwise. Let $a, b \geq 0$ be such that $ak_1 + bk_2 = k'$, $a + b = 1$. Let $C_1 = cost(F_1)$ and $C_2 = cost(F_2)$. Recall that $(F_1, \mathsf{rad}_1, \alpha^1)$ and $(F_2, \mathsf{rad}_2, \alpha^2)$ are the outputs of PDAlg for z_1 and z_2 respectively.

▶ **Claim 4.** *We have $aC_1 + bC_2 \leq (3 + \epsilon)OPT$.*

▶ **Theorem 5.** *k-*BSAlg$(\mathcal{D}', \mathcal{L}', k', \epsilon)$ *returns a feasible solution $\big(F, \{\mathsf{rad}(i)\}\big)$ with $cost(F) \leq \big(6.183 + O(\epsilon)\big) \cdot OPT + O(R^*)$ where $\big\{(i, \mathsf{rad}(i))\big\}_{i\in\mu(F)} \subseteq \mathcal{L}'$ is a set of non-intersecting pairs.*

Proof. The radii $\{\mathsf{rad}(i)\}_{i\in\mu(F)}$ are simply radii obtained from some execution of PDAlg, so $\big\{(i, \mathsf{rad}(i))\big\}_{i\in\mu(F)} \subseteq \mathcal{L}'$ and comprises non-intersecting pairs. If k-BSAlg terminates in step B1, we have a better bound on $cost(F)$. If not, and we return F_2, the cost incurred is C_2.

Otherwise, we return the solution F' found in step B2. Since (C-P) has only one constraint in addition to the bound constraints $0 \leq x_{i,r} \leq 1$, the extreme-point optimal solution x^* has at most one fractional component, and if it has a fractional component, then $\sum_{(i,r)\in F_2} \big(x^*_{i,r} + |\mathcal{S}_{i,r}|(1 - x^*_{i,r})\big) = k'$. For any $(i, r) \in F_2$ with $x^*_{i,r} \in \{0, 1\}$, the number of pairs we include is exactly $x^*_{i,r} + |\mathcal{S}_{i,r}|(1 - x^*_{i,r})$, and the total cost of these pairs is at most the contribution to the objective function of (C-P) from the $x^*_{i,r}$ and $(1 - x^*_{i,r})$ terms. If x^* has a fractional component $(i', r') \in F_2$, then $x^*_{i',r'} + |\mathcal{S}_{i',r'}|(1 - x^*_{i',r'})$ is a *positive* integer. Since we include at most one pair for (i', r'), this implies that $|F'| \leq k'$. The cost of the pair we include is at most $15R^*$, since all $(i, r) \in F_1 \cup F_2$ satisfy $r \leq 3R^*$. Therefore, $cost(F') \leq OPT_{\text{C-P}} + 15R^*$. Also, $OPT_{\text{C-P}} \leq 2bC_2 + (2b + a)C_1 = 2bC_2 + (1 + b)C_1$, since setting $x_{i,r} = b$ for all $(i, r) \in F_2$ yields a feasible solution to (C-P) of this cost.

So when we terminate in step B3, we return a solution F with $cost(F) \leq \min\{C_2, 2bC_2 + (1+b)C_1 + 15R^*\}$. We show that $\min\{C_2, 2bC_2 + (1+b)C_1\} \leq 2.0607(aC_1 + bC_2)$ for all $a, b \geq 0$ with $a + b = 1$. Combining this with Claim 4 yields the bound in the theorem. ◀

Proof. Proof of Theorem 2 It suffices to show that when the selection $F^O = \{(i_1, r_1), \ldots (i_t, r_t)\}$ in step A1 corresponds to the t facilities in an optimal solution with largest radii, we obtain the desired approximation bound. In this case, we have $R^* \leq \frac{O^*}{t} \leq \epsilon O^*$ and $OPT \leq O^* - \sum_{p=1}^{t} r_p$. Combining Theorem 5 and Lemma 1 then yields the theorem. ◀

Improved approximation ratio. The improved approximation ratio comes from a better way of combining F_1 and F_2 in step B2. We observe that the dual solutions α^1 and α^2 are component-wise close to each other (we can control the closeness by controlling δ_z). Thus, we may essentially assume that if $T_{1,I}$, $T_{2,I}$ denote the tight pairs yielding F_1, F_2 respectively, then every pair in $T_{1,I}$ intersects some pair in $T_{2,I}$, because we can augment $T_{2,I}$ to include non-intersecting pairs of $T_{1,I}$. This yields dividends when we combine solutions as in step B2, because we can now ensure that if $\pi(i', r') = (i, r)$, then the pairs of $T_{2,I}$ and $T_{1,I}$ yielding (i, r) and (i', r') respectively intersect, which yields an improved bound on $c_{i,i'}$. This yields an improved approximation of 3.83 for the combination step, and hence for the entire algorithm.

▶ **Theorem 6.** *For any $\epsilon > 0$, our algorithm returns a feasible* LBkSR*-solution of cost at most $(3.83 + O(\epsilon))O^*$ in time $n^{O(1/\epsilon)}$.*

2.2 Approximation algorithm for LBkSRO

We now build upon the ideas in Section 2.1 to devise an $O(1)$-approximation algorithm for the outlier version LBkSR. The high-level approach is similar to the one in Section 2.1. We again "guess" the t (i, r) pairs F^O corresponding to the facilities with largest radii in an optimal solution, and consider the modified k-BS-instance $(\mathcal{D}', \mathcal{L}', k', m)$ (where $\mathcal{D}', \mathcal{L}', k'$ are defined as before). If the LP-relaxation below, (P_2), for the k-BS-problem is infeasible, we move on to the next guess. Otherwise, we design a primal-dual algorithm for the Lagrangian relaxation of the k-BS-problem where we are allowed to pick any number of pairs from $\mathcal{L}'$ (leaving at most m uncovered clients) incurring a fixed cost of z for each pair picked, utilize this to obtain two solutions F_1 and F_2, and combine these to extract a low-cost solution. However, the presence of outliers introduces various difficulties both in the primal-dual algorithm and in the combination step. Consider the following LP-relaxation of the k-BS-problem and its dual.

$$
\begin{array}{lll}
\min & \displaystyle\sum_{(i,r)\in\mathcal{L}'} r \cdot y_{i,r} & (P_2) \\
\text{s.t.} & \displaystyle\sum_{(i,r)\in\mathcal{L}': j\in B(i,r)} y_{i,r} + w_j \geq 1 & \forall j \in \mathcal{D}' \\
& \displaystyle\sum_{(i,r)\in\mathcal{L}'} y_{i,r} \leq k', \quad \sum_{j\in\mathcal{D}'} w_j \leq m & \\
& y, w \geq 0. &
\end{array}
\qquad
\begin{array}{lll}
\max & \displaystyle\sum_{j\in\mathcal{D}'} \alpha_j - k' \cdot z - m \cdot \gamma & (D_2) \\
\text{s.t.} & \displaystyle\sum_{j\in B(i,r)\cap\mathcal{D}'} \alpha_j - z \leq r & \forall (i,r) \in \mathcal{L}' \\
& \alpha_j \leq \gamma & \forall j \in \mathcal{D}' \\
& \alpha, z, \gamma \geq 0. &
\end{array}
\tag{3}
$$

Let OPT denote the optimal value of (P_2). The natural modification of the earlier primal-dual algorithm PDAlg is to now stop the dual-ascent process when the number of active clients is at most m and set $\gamma = \max_{j\in\mathcal{D}'} \alpha_j$. This introduces the significant complication

that one may not be able to pay for the $r+z$-cost of non-intersecting tight pairs selected in the pruning phase by the dual objective value $\sum_{j\in\mathcal{D}'}\alpha_j - m\cdot\gamma$, since clients with $\alpha_j=\gamma$ may be needed to pay for the $r+z$-cost of the last tight pair $f=(i_f,r_f)$ but their contribution gets canceled by the $-m\cdot\gamma$ term. This issue affects us in various guises. First, we no longer obtain an LMP-approximation for the unconstrained problem since we have to account for the $(r+z)$-cost of f separately. Second, unlike Claim 4, given solutions F_1 and F_2 obtained via binary search for $z_1, z_2 \approx z_1$ respectively with $|F_2|\le k'\le|F_1|$, we now only obtain a fractional k-BS-solution of cost $O(OPT+z_1)$. While one can modify the covering-knapsack-LP based procedure in step B2 of k-BSAlg to combine F_1, F_2, this only yields a good solution when $z_1=O(OPT)$. The chief technical difficulty is that z_1 may however be much larger than OPT. Overcoming this obstacle requires various novel ideas and is the key technical contribution of our algorithm. We design a second combination procedure that is guaranteed to return a good solution when $z_1=\Omega(OPT)$. This requires establishing certain structural properties for F_1 and F_2, using which we argue that one can find a good solution in the neighborhood of F_1 and F_2.

We now detail the changes to the primal-dual algorithm and k-BSAlg in Section 2.1, and analyze them to prove the following theorem.

▶ **Theorem 7.** *There exists a $\big(12.365+O(\epsilon)\big)$-approximation algorithm for* LBkSRO *that runs in time $n^{O(1/\epsilon)}$ for any $\epsilon>0$.*

Modified primal-dual algorithm PDAlg$^{\circ}(\mathcal{D}',\mathcal{L}',z)$. This is quite similar to PDAlg (and we again return pairs from $\mathcal{L}$). We stop the dual-ascent process when there are at most m active clients. We set $\gamma=\max_{j\in\mathcal{D}'}\alpha_j$. Let $f=(i_f,r_f)$ be the last tight pair added to the tight-pair set T, and $B_f=B(i_f,r_f)$. We sometimes abuse notation and use (i,r) to also denote the singleton set $\{(i,r)\}$. For a set P of (i,r) pairs, define $\mathsf{uncov}(P):=\mathcal{D}'\setminus\bigcup_{(i,r)\in P}B(i,r)$. Note that $|\mathsf{uncov}(T\setminus f)|>m\ge|\mathsf{uncov}(T)|$. Let Out be a set of m clients such that $\mathsf{uncov}(T)\subseteq Out\subseteq\mathsf{uncov}(T\setminus f)$. Note that $\alpha_j=\gamma$ for all $j\in Out$.

The pruning phase is similar to before, but we only use f if necessary. Let T_I be a maximal subset of non-intersecting pairs picked by greedily scanning pairs in $T\setminus f$ in non-increasing order of radius. For $i\in\mu(T_I)$, set $\mathsf{rad}(i)$ to be the unique r such that $(i,r)\in T_I$, and let r_i be the smallest radius ρ such that $B(i,\rho)\supseteq B(i',r')$ for every $(i',r')\in T\setminus f$ such that $r'\le\mathsf{rad}(i)$ and (i',r') intersects $(i,\mathsf{rad}(i))$. Let $F'=\{(i,r_i)\}_{i\in\mu(T_I)}$. If $\mathsf{uncov}(F')\le m$, set $F=F'$. If $\mathsf{uncov}(F')>m$ and $\exists i\in\mu(F')$ such that $c(i,i_f)\le 2R^*$, then increase r_i so that $B(i,r_i)\supseteq B_f$ and let F be this updated F'. Otherwise, set $F=F\cup f$ and $r_{i_f}=\mathsf{rad}(i_f)=r_f$. We return $(F,f,Out,\{rad(i)\}_{i\in\mu(F)},\alpha,\gamma)$.

▶ **Theorem 8.** *Let $(F,f,Out,\{\mathsf{rad}(i)\},\alpha,\gamma)=\mathsf{PDAlg}^{\circ}(\mathcal{D}',\mathcal{L}',z)$. Then:*
- **(i)** $\mathsf{uncov}(F)\le m$;
- **(ii)** $cost(F\setminus f)+3|F\setminus f|z-3R^*\le 3(\sum_{j\in\mathcal{D}'}\alpha_j-m\gamma)\le 3(OPT+k'z)$;
- **(iii)** $\big\{(i,\mathsf{rad}(i))\big\}_{i\in\mu(F)}\subseteq\mathcal{L}'$, *is a set of non-intersecting pairs,* $\mathsf{rad}(i)\le r_i\le 3R^*\ \forall i\in\mu(F)$;
- **(iv)** *if $|F\setminus f|\ge k'$ then $cost(F)\le 3\cdot OPT+4R^*$, and if $|F\setminus f|>k'$ then $z\le OPT$.*

Modified algorithm k-BSAlg$^{\circ}(\mathcal{D}',\mathcal{L}',k',\epsilon)$. We again use binary search to find solutions F_1, F_2 and extract a low-cost solution from these. The only changes to step B1 are as follows. We start with $z_1=0$ and $z_2=2nk'c_{\max}$; for this z_2, one can argue PDAlg$^{\circ}$ returns at most k' pairs. We stop when $z_2-z_1\le\delta_z:=\frac{\epsilon OPT}{3n2^n}$. We *do not stop* even if PDAlg$^{\circ}$ returns a solution $(F,\ldots)$ with $|F|=k'$ for some $z=\frac{z_1+z_2}{2}$, since *Theorem 8 is not strong enough to bound cost(F) even when this happens!* If $|F|>k'$, we update $z_1\leftarrow z$ and the F_1-solution;

otherwise, we update $z_2 \leftarrow z$ and the F_2-solution. Thus, we maintain that $k_1 = |F_1| > k'$, and $k_2 = |F_2| \leq k'$.

The main change is in the way solutions F_1, F_2 are combined. We adapt step B2 to handle outliers (procedure $\mathcal{A}$ in Section 2.2.1), but the key extra ingredient is that we devise an alternate combination procedure $\mathcal{B}$ (Section 2.2.2) that returns a low-cost solution when $z_1 = \Omega(OPT)$. We return the better of the solutions output by the two procedures. Combining Theorem 9 with Lemma 1 (for the right selection of t (i,r) pairs) yields Theorem 7.

▶ **Theorem 9.** *k-BSAlg$^\circ(\mathcal{D}', \mathcal{L}', k', \epsilon)$ returns a solution (F, rad) with $cost(F) \leq \big(12.365 + O(\epsilon)\big) \cdot OPT + O(R^*)$ where $\big\{(i, \mathsf{rad}(i))\big\}_{i \in \mu(F)} \subseteq \mathcal{L}'$ comprises non-intersecting pairs.*

2.2.1 Combination subroutine $\mathcal{A}\Big((F_1, \mathsf{rad}_1), (F_2, \mathsf{rad}_2)\Big)$

As in step B2, we cluster the F_1-pairs around F_2-pairs in stars. However, unlike before, some $(i', r') \in F_1$ may remain *unclustered* and and we may not pick (i', r') or some pair close to it. Since we do not cover all clients covered by F_1, we need to cover a suitable number of clients from $\mathsf{uncov}(F_1)$. We again setup an LP to obtain a suitable collection of pairs, which is now a 2-dimensional covering knapsack LP, and use the structure of an extreme-point optimal solution to extract from it a good collection of pairs.

▶ **Theorem 10.** *We can obtain a solution $\big(F, \{\mathsf{rad}(i)\}_{i \in \mu(F)}\big)$ to the k-BS-problem with $cost(F) \leq \big(6.1821 + O(\epsilon)\big)(OPT + z_1) + O(R^*)$ where $\big\{(i, \mathsf{rad}(i))\big\}_{i \in \mu(F)} \subseteq \mathcal{L}'$ is a set of non-intersecting pairs.*

2.2.2 Subroutine $\mathcal{B}\Big((F_1, f_1, Out_1, \mathsf{rad}_1, \alpha^1, \gamma^1), (F_2, f_2, Out_2, \mathsf{rad}_2, \alpha^2, \gamma^2)\Big)$

Subroutine $\mathcal{A}$ in the previous section yields a low-cost solution only if $z_1 = O(OPT)$. We complement subroutine $\mathcal{A}$ by now describing a procedure that returns a good solution when z_1 is large. We assume in this section that $z_1 > (1+\epsilon)OPT$. Then $|F_1 \setminus f_1| \leq k'$ (otherwise $z \leq OPT$ by part (iv) of Theorem 8), so $|F_1 \setminus f_1| \leq k' < |F_1|$, which means that $k_1 = k' + 1$ and $f_1 \in F_1$. Hence, $\alpha^1_j = \gamma^1$ for all $j \in B_{f_1} \cap \mathcal{D}'$. We utilize the following *continuity lemma*, which is essentially Lemma 6.6 in [11]; we include a proof in the full version of the paper.

▶ **Lemma 11.** *Let $(F_p, \ldots, \alpha^p, \gamma^p) = \mathsf{PDAlg}^\circ(\mathcal{D}', \mathcal{L}', z_p)$ for $p = 1, 2$, where $0 \leq z_2 - z_1 \leq \delta_z$. Then, $\|\alpha^1_j - \alpha^2_j\|_\infty \leq 2^n\delta_z$ and $|\gamma^1 - \gamma^2| \leq 2^n\delta_z$. Thus, if (3) is tight for some $(i,r) \in \mathcal{L}'$ in one execution, then $\sum_{j \in B(i,r) \cap \mathcal{D}'} \alpha^p_j \geq r + z - 2^n\delta_z$ for $p = 1, 2$.*

First, we take care of some simple cases. If there exists $(i,r) \in F_1 \setminus f_1$ such that $\big|\mathsf{uncov}\big(F_1 \setminus \{f_1, (i,r)\} \cup (i, r + 12R^*)\big)\big| \leq m$, then set $F = F_1 \setminus \{f_1, (i,r)\} \cup (i, r + 12R^*)$. We have $cost(F) = cost(F_1 \setminus f_1) + 12R^* \leq 3 \cdot OPT + 15R^*$ (by part (ii) of Theorem 8). If there exist pairs $(i,r), (i',r') \in F_1$ such that $c(i,i') \leq 12R^*$, take r'' to be the minimum $\rho \geq r$ such that $B(i',r') \subseteq B(i,\rho)$ and set $F = F_1 \setminus \{(i,r), (i',r')\} \cup (i, r'')$. We have $cost(F) \leq cost(F_1 \setminus f_1) + 13R^* \leq 3 \cdot OPT + 16R^*$. In both cases, we return $\big(F, \{\mathsf{rad}_1(i)\}_{i \in \mu(F)}\big)$.

So we assume in the sequel that neither of the above apply. In particular, all pairs in F_1 are well-separated. Let $AT = \{(i,r) \in \mathcal{L}' : \sum_{j \in B(i,r) \cap \mathcal{D}'} \alpha^1_j \geq r + z_1 - 2^n\delta_z\}$ and $AD = \{j \in \mathcal{D}' : \alpha^1_j \geq \gamma^1 - 2^n\delta_z\}$. By Lemma 11, AT includes the tight pairs of $\mathsf{PDAlg}^\circ(\mathcal{D}', \mathcal{L}', z_p)$ for both $p = 1, 2$, and $Out_1 \cup Out_2 \subseteq AD$. Since the tight pairs T_2 used for building solution F_2 are almost tight in $(\alpha^1, \gamma^1, z_1)$, we swap them in and swap out pairs from F_1 one by one while maintaining a feasible solution. Either at some point, we will

be able to remove f, which will give us a solution of size k', or we will obtain a bound on $cost(F_2)$. The following lemma is our main tool for bounding the cost of the solution returned.

▶ **Lemma 12.** *Let $F \subseteq \mathcal{L}'$, and $T_F = \{(i, r'_i)\}_{i \in \mu(F)}$ where $r'_i \leq r$ for each $(i, r) \in F$. Suppose $T_F \subseteq AT$ and consists of non-intersecting pairs. If $|F| \geq k'$ and $|AD \setminus \bigcup_{(i,r)\in F} B(i, r))| \geq m$ then $cost(T_F) \leq (1 + \epsilon)OPT$. Moreover, if $|F| > k'$ then $z_1 \leq (1 + \epsilon)OPT$.*

Define a mapping $\psi : F_2 \to F_1 \setminus f_1$ as follows. Note that any $(i, r) \in F_2$ may intersect with at most one F_1-pair: if it intersects $(i', r'), (i'', r'') \in F_1$, then we have $c(i', i'') \leq 12R^*$. First, for each $(i, r) \in F_2$ that intersects with some $(i', r') \in F_1$, we set $\psi(i, r) = (i', r')$. Let $M \subseteq F_2$ be the F_2-pairs mapped by ψ this way. For every $(i, r) \in F_2 \setminus M$, we arbitrarily match (i, r) with a *distinct* $(i', r') \in F_1 \setminus \psi(M)$. We claim that ψ is in fact a one-one function.

▶ **Lemma 13.** *Every $(i, r) \in F_1 \setminus f_1$ intersects with at most one F_2-pair.*

Let F'_2 be the pairs $(i, r) \in F_2$ such that if $(i', r') = \psi(i, r)$, then $r' < r$. Let $P = F'_2 \cap M$ and $Q = F'_2 \setminus M$. For every $(i', r') \in \psi(Q)$ and $j \in B(i', r')$, we have $j \in \mathsf{uncov}(F_2) \subseteq AD$ (else (i', r') would lie in $\psi(M)$). Starting with $F = F_1 \setminus f_1$, we iterate over $(i, r) \in F'_2$ and do the following. Let $(i', r') = \psi(i, r)$. If $(i, r) \in P$, we update $F \leftarrow F \setminus (i', r') \cup (i, r + 2r')$ (so $B(i, r + 2r') \supseteq B(i', r')$), else we update $F \leftarrow F \setminus (i', r') \cup (i, r)$. Let $T_F = \{(i, \mathsf{rad}_1(i))\}_{(i,r)\in F\cap F_1} \cup \{(i, \mathsf{rad}_2(i))\}_{(i,r)\in F\setminus F_1}$. Note that $|F| = k'$ and $\mathsf{uncov}(F) \subseteq AD$ at all times. Also, since (i, r) intersects only (i', r'), which we remove when (i, r) is added, we maintain that T_F is a collection of non-intersecting pairs and a subset of $AT \subseteq \mathcal{L}'$. This process continues until $|\mathsf{uncov}(F)| \leq m$, or when all pairs of F'_2 are swapped in. In the former case, we argue that $cost(F)$ is small and return $\big(F, \{\mathsf{rad}_1(i)\}_{(i,r)\in F\cap F_1} \cup \{\mathsf{rad}_2(i)\}_{(i,r)\in F\setminus F_1}\big)$. In the latter case, we show that $cost(F'_2)$, and hence $cost(F_2)$ is small, and return (F_2, rad_2).

▶ **Lemma 14.**

(i) *If the algorithm stops with $|\mathsf{uncov}(F)| \leq m$, $cost(F) \leq (9 + 3\epsilon)OPT + 18R^*$.*

(ii) *If case (i) does not apply, then $cost(F_2) \leq (3 + 3\epsilon)OPT + 9R^*$.*

(iii) *The pairs corresponding to the radii returned are non-intersecting, and a subset of $\mathcal{L}'$.*

3 Minimizing the maximum radius with lower bounds and outliers

The *lower-bounded k-supplier with outliers* (LBkSupO) problem is the min max-radius version of LBkSRO. The input and the set of feasible solutions are the same as in LBkSRO: the input is an instance $\mathcal{I} = \big(\mathcal{F}, \mathcal{D}, \{c(i, j)\}, \{L_i\}, k, m\big)$, and a feasible solution is $\big(S \subseteq \mathcal{F}, \sigma : \mathcal{D} \mapsto S \cup \{\mathsf{out}\}\big)$ with $|S| \leq k$, $|\sigma^{-1}(i)| \geq L_i$ for all $i \in S$, and $|\sigma^{-1}(\mathsf{out})| \leq m$. The cost of (S, σ) is now $\max_{i\in S} \max_{j\in\sigma^{-1}(i)} c(i, j)$. The case $m = 0$ is called the *lower-bounded k-supplier* (LBkSup) problem, and the setting where $\mathcal{D} = \mathcal{F}$ is often called the *k-center* version.

Let τ^* denote the optimal value; note that there are only polynomially many choices for τ^*. As is common in the study of min-max problems, we reduce the problem to a "graphical" instance, where given some value τ, we try to find a solution of cost $O(\tau)$ or deduce that $\tau^* > \tau$. We construct a bipartite unweighted graph $G_\tau = \big(V_\tau = \mathcal{D} \cup \mathcal{F}_\tau, E_\tau\big)$, where $\mathcal{F}_\tau = \{i \in \mathcal{F} : |B(i, \tau)| \geq L_i\}$, and $E_\tau = \{ij : c(i, j) \leq \tau, i \in \mathcal{F}_\tau, j \in \mathcal{D}\}$. Let $dist_\tau(i, j)$ denote the shortest-path distance in G_τ between i and j, so $c(i, j) \leq dist_\tau(i, j) \cdot \tau$. We say that an assignment $\sigma : \mathcal{D} \mapsto \mathcal{F}_\tau \cup \{\mathsf{out}\}$ is a *distance-α assignment* if $dist_\tau(j, \sigma(j)) \leq \alpha$ for every client j with $\sigma(j) \neq \mathsf{out}$. We call such an assignment feasible, if it yields a feasible LBkSupO-solution, and we say that G_τ is feasible if it admits a feasible distance-1 assignment. It is not hard to see that given $F \subseteq \mathcal{F}_\tau$, the problem of finding a feasible

distance-α-assignment $\sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}$ in G_τ (if one exists) can be solved by creating a network-flow instance with lower bounds and capacities.

Observe that an optimal solution yields a feasible distance-1 assignment in G_{τ^*}. We devise an algorithm that for every τ, either finds a feasible distance-α assignment in G_τ for some constant α, or detects that G_τ is not feasible. This yields an α-approximation algorithm since the smallest τ for which the algorithm returns a feasible LBkSupO-solution must be at most τ^*. We obtain Theorems 15 and 16 via this template, and complement these via a hardness result (Theorem 17) showing that our approximation factor for LBkSup is tight.

▶ **Theorem 15.** *There is a* 3*-approximation algorithm for* LBkSup.

▶ **Theorem 16.** *There is a* 5*-approximation algorithm for* LBkSupO.

▶ **Theorem 17.** *It is NP-hard to approximate* LBkSup *within a factor better than* 3.

Finding a distance-5 assignment for LBkSupO. The goal is to find a set $F \subseteq \mathcal{F}_\tau$ of at most k centers that are close to the centers in $F^* \subseteq \mathcal{F}_\tau$ for some feasible distance-1 assignment $\sigma^* : \mathcal{D} \mapsto F^* \cup \{\mathsf{out}\}$ in G_τ. If centers in F do not share a neighbor in G_τ, then clients in $N(i)$ can be assigned to i for each $i \in F$ to satisfy the lower bounds.

▶ **Definition 18** ([14])**.** Given the graph G_τ, a set $F \subseteq \mathcal{F}$ is called a *skeleton* if it satisfies the following properties.

(a) (*Separation property*) For $i, i' \in F$, $i \neq i'$, we have $dist_\tau(i, i') \geq 6$;

(b) There exists a feasible distance-1 assignment $\sigma^* : \mathcal{D} \mapsto F^* \cup \{\mathsf{out}\}$ in G_τ such that

- (*Covering property*) For all $i^* \in F^*$, $dist_\tau(i^*, F) \leq 4$, where $dist_\tau(i^*, F) = \min_{i \in F} dist_\tau(i^*, i)$.
- (*Injection property*) There exists $f : F \mapsto F^*$ such that $dist_\tau(i, f(i)) \leq 2$ for all $i \in F$.

If F satisfies the separation and injection properties, it is called a *pre-skeleton*.

▶ **Lemma 19.** *Let F be a pre-skeleton in G_τ. Define $U = \{i \in \mathcal{F}_\tau : dist_\tau(i, F) \geq 6\}$ and let $i = \arg\max_{i' \in U} |N(i')|$. Then, either F is a skeleton, or $F \cup \{i\}$ is a pre-skeleton.*

Suppose $F \subseteq \mathcal{F}_\tau$ is a skeleton and satisfies the properties with respect to a feasible distance-1 assignment (F^*, σ^*). The separation property ensures that the neighbor sets of any two locations $i, i' \in F$ are disjoint. The covering property ensures that F^* is at distance at most 4 from F, so there are at least $|\mathcal{D}| - m$ clients at distance at most 5 from F. Finally, the injection and separation properties together ensure that $|F| \leq k$. Thus, if F is a skeleton, then we can obtain a feasible distance-5 assignment $\sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}$.

If G_τ is feasible, then $\emptyset$ is a pre-skeleton. A skeleton can have size at most k. So using Lemma 19, we can find a sequence $\mathcal{F}'$ of at most $k+1$ subsets of $\mathcal{F}_\tau$ by starting with $\emptyset$ and repeatedly applying Lemma 19 until we either have a set of size k or the set U in Lemma 19 is empty. By Lemma 19, if G_τ is feasible then one of these sets must be a skeleton. So for each $F \in \mathcal{F}'$, we check if there exists a feasible distance-5 assignment $\sigma : \mathcal{D} \mapsto F \cup \{\mathsf{out}\}$, and if so, return (F, σ). Otherwise we return that G_τ is not feasible.

References

1 Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Samir Khuller, Rina Panigrahy, Dilys Thomas, and An Zhu. Achieving anonymity via clustering. *ACM Transactions on Algorithms (TALG)*, 6(3):49, 2010.

2 Gagan Aggarwal, Tomás Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. *Journal of Privacy Technology (JOPT)*, 2005.

3 Sara Ahmadian and Chaitanya Swamy. Improved approximation guarantees for lower-bounded facility location. In *Proceedings of the 10th International Workshop on Approximation and Online Algorithms*, pages 257–271. Springer, 2012.

4 Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated k-center. *Mathematical Programming*, 154(1-2):29–53, 2015.

5 Babak Behsaz and Mohammad R Salavatipour. On minimum sum of radii and diameters clustering. *Algorithmica*, 73(1):143–165, 2015.

6 Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 737–756. SIAM, 2015.

7 Vasilis Capoyleas, Günter Rote, and Gerhard Woeginger. Geometric clusterings. *Journal of Algorithms*, 12(2):341–356, 1991.

8 Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

9 Moses Charikar, Sudipto Guha, Éva Tardos, and David B Shmoys. A constant-factor approximation algorithm for the k-median problem. In *Proceedings of the 31st annual ACM Symposium on Theory of Computing*, pages 1–10. ACM, 1999.

10 Moses Charikar, Samir Khuller, David Mount, and Giri Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 642–651. SIAM, 2001.

11 Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, pages 1–10. ACM, 2001.

12 Ke Chen. A constant factor approximation algorithm for k-median clustering with outliers. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 826–835. SIAM, 2008.

13 Marek Cygan, MohammadTaghi Hajiaghayi, and Samir Khuller. Lp rounding for k-centers with non-uniform hard capacities. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science*, pages 273–282. IEEE, 2012.

14 Marek Cygan and Tomasz Kociumaka. Constant factor approximation for capacitated k-center with outliers. In *Proc. 31st International Symposium on Theoretical Aspects of Computer Science (STACS 2014)*, volume 25 of *LIPIcs – Leibniz International Proceedings in Informatics*, pages 251–262. Schloss Dagstuhl –Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.STACS.2014.251.

15 Srinivas R Doddi, Madhav V Marathe, SS Ravi, David Scot Taylor, and Peter Widmayer. Approximation algorithms for clustering to minimize the sum of diameters. In *Proceedings of the 11th Scandinavian Workshop on Algorithm Theory*, pages 237–250, 2000.

16 Alina Ene, Sariel Har-Peled, and Benjamin Raichel. Fast clustering with lower bounds: No customer too far, no shop too small. *arXiv preprint arXiv:1304.7318*, 2013.

17 Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A Pirwani, and Kasturi Varadarajan. On metric clustering to minimize the sum of radii. *Algorithmica*, 57(3):484–498, 2010.

18 Matt Gibson, Gaurav Kanade, Erik Krohn, Imran A Pirwani, and Kasturi Varadarajan. On clustering to minimize the sum of radii. *SIAM Journal on Computing*, 41(1):47–60, 2012.

19 Sudipto Guha, Adam Meyerson, and Kamesh Munagala. Hierarchical placement and network design problems. In *Proceedings of 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612. IEEE, 2000.

20 Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

21 Dorit S Hochbaum and David B Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM*, 33(3):533–550, 1986.

22 Kamal Jain and Vijay V Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *Journal of the ACM*, 48(2):274–296, 2001.

23 David R. Karger and Maria Minkoff. Building steiner trees with incomplete global knowledge. In *Proceedings of 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623. IEEE, 2000.

24 Samir Khuller and Yoram J Sussmann. The capacitated k-center problem. *SIAM Journal on Discrete Mathematics*, 13(3):403–418, 2000.

25 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, pages 901–910. ACM, 2013.

26 Andrew Lim, Fan Wang, and Zhou Xu. A transportation problem with minimum quantity commitment. *Transportation Science*, 40(1):117–129, 2006.

27 Pitu B Mirchandani and Richard L Francis. *Discrete location theory*. Wiley-Interscience, 1990.

28 Pierangela Samarati. Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

29 David B Shmoys. The design and analysis of approximation algorithms. *Trends in Optimization: American Mathematical Society Short Course, January 5-6, 2004, Phoeniz, Arizona*, 61:85, 2004.

30 Zoya Svitkina. Lower-bounded facility location. *ACM Transactions on Algorithms (TALG)*, 6(4):69, 2010.

A Duality Based 2-Approximation Algorithm for Maximum Agreement Forest*†

Frans Schalekamp‡1, Anke van Zuylen§2, and Suzanne van der Ster3

1 **School of Operations Research & Information Engineering, Cornell University, Ithaca, NY, USA**
fms9@cornell.edu

2 **Department of Mathematics, College of William & Mary, Williamsburg, VA,USA**
anke@wm.edu

3 **Institut für Informatik, Technische Universität München, München, Germany**
ster@in.tum.de

Abstract

We give a 2-approximation algorithm for the Maximum Agreement Forest problem on two rooted binary trees. This NP-hard problem has been studied extensively in the past two decades, since it can be used to compute the Subtree Prune-and-Regraft (SPR) distance between two phylogenetic trees. Our result improves on the very recent 2.5-approximation algorithm due to Shi, Feng, You and Wang (2015). Our algorithm is the first approximation algorithm for this problem that uses LP duality in its analysis.

1998 ACM Subject Classification F.2.2 Analysis of algorithms and problem complexity

Keywords and phrases Maximum agreement forest, phylogenetic tree, SPR distance, subtree prune-and-regraft distance, computational biology

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.70

1 Introduction

Evolutionary relationships are often modeled by a rooted tree, where the leaves are a set of species, and internal nodes are (putative) common ancestors of the leaves below the internal node. Such phylogenetic trees date back to Darwin [5], who used them in his notebook to elucidate his thoughts on evolution.

The topology of phylogenetic trees can be based on different sources of data, e.g., morphological data, behavioral data, genetic data, etc., which can lead to different phylogenetic trees on the same set of species. Different measures have been proposed to measure the similarity of (or distance between) different phylogenetic trees on the same set of species (or individuals). Using the size of a maximum subtree common to both input trees as a similarity measure was proposed by Gordon [8]. The problem of finding such a subtree is now known as the Maximum Agreement Subtree Problem, and has been studied extensively.

* Full version is available at http://arxiv.org/abs/1511.06000.

† This work was initiated when the authors were visitors of Leen Stougie at the Tinbergen Institute.

‡ FS was supported in part by the Simon Prize for Excellence in the Teaching of Mathematics at William & Mary.

§ AvZ was supported in part by a William & Mary Summer Research Award, NSF Prime Award: HRD-1107147, Women in Scientific Education (WISE) and by a grant from the Simons Foundation (#359525, Anke Van Zuylen).

Steel and Warnow [14] are the first to give a polynomial-time algorithm for this problem. Their approach is refined to an $O(n^{1.5} \log n)$-time algorithm by Farach and Thorup [6], who subsequently show their algorithm is optimal, unless unweighted bipartite matching can be solved in near linear time [7].

There exist non-tree-like evolutionary processes that preclude the existence of a phylogenetic tree, so-called reticulation events (such as hybridization, recombination and horizontal gene transfer). In this context, a particularly meaningful measure of comparing phylogenetic trees is the SPR-distance measure (where SPR is short for Subtree Prune-and-Regraft): this measure provides a lower bound on a certain type of these non-tree evolutionary events. The problem of finding the exact value of this measure for a set of species motivated the formulation of the Maximum Agreement Forest Problem (MAF) by Hein, Jian, Wang and Zhang [9]. Since the introduction by Hein et al., MAF has been extensively studied, including several variants, such as the problem where the input consists of more than two trees, whether the input trees are rooted or unrooted, binary or non-binary. In our paper, we concentrate on MAF on two rooted binary trees.

For ease of defining the solutions to MAF on two rooted binary trees, we think of the input trees as being directed, where all edges are directed away from the root. Given two rooted binary trees on the same leaf set $\mathcal{L}$, the MAF problem asks to find a minimum set of edges to delete from the two trees, so that the directed trees in the resulting two forests can be paired up into pairs of "isomorphic" trees. Two trees are isomorphic if they contain the same nodes from $\mathcal{L}$ and recursively removing nodes of out-degree zero that are not in $\mathcal{L}$ and contracting two arcs incident to a node of in-degree and out-degree one, results in the same binary tree. An alternative (but equivalent) definition will be given in Section 2.

The problem of finding a MAF on two rooted binary trees has been extensively studied, although unfortunately some of the published results later turned out to have subtle errors. First of all, Allen and Steel [1] point out that the claim by Hein et al. that solving MAF on two rooted directed trees computes the SPR-distance between the trees is incorrect. Bordewich and Semple [4] show how to redefine MAF for rooted directed trees so that it is indeed the case that the optimal objective value of MAF is equal to the SPR-distance. In the paper in which they introduce MAF, Hein et al. [9] proved NP-hardness and they give an approximation algorithm for the problem, the approximation guarantee of which turned out to be slightly higher than what was claimed. Bordewich and Semple [4] show that, for their corrected definition of MAF, NP-hardness continues to hold. Other approximation algorithms followed [11, 2, 3]. The current best approximation ratio for MAF is 2.5, due to Shi, Feng, You and Wang [13], and Rodrigues [10] has shown that MAF is MAXSNP-hard. In addition, there is a body of work on other approaches, such as Fixed-Parameter Tractable (FPT) algorithms (e.g., [16, 15]) and Integer Programming [17, 18].

In this paper we give an improved approximation algorithm for MAF with an analysis based on linear programming duality. Our 2-approximation algorithm differs from previous works in two aspects. First of all, in terms of bounding the optimal value, we construct a feasible dual solution, rather than arguing more locally about the objective of the optimal solution. Secondly, our algorithm itself also takes a more global approach, whereas the algorithms in previous works mainly consider local substructures of at most four leaves. In particular, we identify a minimal subtree[1] of one of the two input trees for which the leaf set is incompatible, i.e., when deleting all other leaves from both trees, the remaining two trees

[1] By subtree of a rooted tree, we mean a tree containing the leaves that are descendents of some particular node in the rooted tree (including this node itself), and all edges between them.

are not isomorphic (such a minimal subtree can be found efficiently). We then use "local" operations which repeatedly look at two "sibling" leaves in the minimal incompatible subtree, and perform similar operations as those suggested by previous authors.

Preliminary tests were conducted using a proof-of-concept implementation of our algorithm in Java. Our preliminary results indicate that our algorithm finds a dual solution that in 44% of the 1000 runs is equal to the optimal dual solution, and in 37% of the runs is 1 less than the optimal solution. The observed average approximation ratio is about 1.92; following our algorithm with a simple greedy search algorithm decreases this to less than 1.28.

The remainder of our paper is organized as follows. In Section 2, we formally define the problem. In Section 3, we give a 3-approximation algorithm for MAF that introduces the dual linear program that is used throughout the remainder of the paper and gives a flavor of the arguments used to prove the approximation ratio of two. In Section 4, we give an overview of the 2-approximation algorithm and the key ideas in its analysis. In Section 5 we give more details on the randomly generated instances that we used to obtain our preliminary experimental results, and we conclude in Section 6 with some directions for future research.

The full version of this paper [12] contains further details on the algorithm and a complete analysis that shows that its approximation ratio is 2.

2 Preliminaries

The input to the Maximum Agreement Forest problem (MAF) consists of two rooted binary trees T_1 and T_2, where the leaves in each tree are labeled with the same label set $\mathcal{L}$. Each leaf has exactly one label, and each label in $\mathcal{L}$ is assigned to exactly one leaf in T_1, and one leaf in T_2. For ease of exposition, we sometimes think of the edges in the trees as being directed, so that there is a directed path from the root to each of the leaves.

We call the non-leaf nodes the internal nodes of the trees, and we let V denote the set of all nodes (internal nodes and leaves) in $T_1 \cup T_2$. Given a tree containing u and v, we let $\text{lca}(u, v)$ denote the lowest (closest to the leaves, furthest from the root) common ancestor of u and v. We let $\text{lca}_1(u, v)$ and $\text{lca}_2(u, v)$ denote $\text{lca}(u, v)$ in tree T_1, respectively, T_2. We extend this notation to $\text{lca}(U)$ which will denote the lowest common ancestor of a set of leaves U. For three leaves u, v, w and a rooted tree T, we use the notation $uv|w$ in T to denote that $\text{lca}(u, v)$ is a descendent of $\text{lca}(\{u, v, w\})$. A triplet $\{u, v, w\}$ of labeled leaves is *consistent* if $uv|w$ in $T_1 \Leftrightarrow uv|w$ in T_2. The triplet is called *inconsistent* otherwise. We call a set of leaves $L \subseteq \mathcal{L}$ a *compatible set*, if it does not contain an inconsistent triplet.

For a compatible set $L \subseteq \mathcal{L}$, define $V[L] := \{v \in V$: there exists a pair of leaves u, u' in L so that v is on the path between u and u' in T_1 or $T_2\}$. Then, a partitioning $L_1, L_2, \ldots, L_p$ of $\mathcal{L}$ corresponds to a feasible solution to MAF with objective value $p - 1$, if the sets $L_1, L_2, \ldots, L_p$ are compatible, and the sets $V[L_j]$ for $j = 1, \ldots, p$ are node disjoint. Using this definition, we can write the following Integer Linear Program[2] for MAF: Let $\mathcal{C}$ be the collection of all compatible sets of leaves, and introduce a binary variable x_L for every compatible set $L \in \mathcal{C}$, where the variable takes value 1 if the optimal solution to MAF has a tree with leaf set L. The constraints ensure that each leaf $v \in \mathcal{L}$ is in some tree in the optimal forest, and each internal node $v \in V \setminus \mathcal{L}$ is in at most one tree in the optimal forest. The objective encodes the fact that we need to delete $\sum_{L \in \mathcal{C}} x_L - 1$ edges from each of T_1

[2] This ILP was obtained in discussions with Neil Olver and Leen Stougie.

and T_2 to obtain forests with $\sum_{L\in\mathcal{C}} x_L$ trees.

$$\begin{array}{lll} \text{minimize} & \sum_{L\in\mathcal{C}} x_L - 1, & \\ \text{s.t.} & \sum_{L:v\in L} x_L = 1 & \forall v \in \mathcal{L}, \\ & \sum_{L:v\in V[L]} x_L \leq 1 & \forall v \in V \setminus \mathcal{L}, \\ & x_L \in \{0,1\} & \forall L \in \mathcal{C}. \end{array}$$

Remark

The definition of MAF we use is *not* the definition that is now standard in the literature, but any (approximation) algorithm for our version can be used to get the same (approximation) result for the standard formulation: The standard formulation was introduced by Bordewich and Semple [4] in order to ensure that the objective value of MAF is equal to the rooted SPR distance. They note that for this to hold, we need the additional requirement that the two forests must also agree on the tree containing the original root; in other words, the original roots of T_1 and T_2 should be contained in a tree with the same (compatible) subset of leaves. An easy reduction shows that we can solve this problem using our definition of MAF: given two rooted binary trees for which we want to compute the SPR distance, we can simply add one new label ρ, and for each of the two input trees, we add a new root which has an edge to the original root and an edge to a new node with label ρ.[3] A solution to "our" MAF problem on this modified input defines a solution to Bordewich and Semple's problem on the original input with the same objective value and vice versa.

3 Duality Based 3-Approximation Algorithm

3.1 Algorithm

The algorithm we describe in this section is a variant of the algorithm of Rodrigues et al. [11] (see also Whidden and Zeh [16]). The algorithm maintains two forests, T_1' and T_2' on the same leaf set $\mathcal{L}'$, and iteratively deletes edges from these forests. At the start, T_1' is set equal to T_1, T_2' to T_2 and $\mathcal{L}'$ to $\mathcal{L}$. The leaves in $\mathcal{L}'$ are called the *active* leaves. The algorithm will ensure that the leaves that are not active, will have been resolved in one of the two following ways: (1) they are part of a tree that contains only inactive leaves in both T_1' and T_2'; these two trees then have the same leaf set, which is compatible, and they will be part of the final solution; or (2) an inactive leaf is *merged* with another leaf which is active, and in the final solution this inactive leaf will be in the same tree as the leaf it was merged with.

A tree is called active if it contains a leaf in $\mathcal{L}'$, and the tree is called inactive otherwise. An invariant of the algorithm is that there is a single active tree in T_1'.

We define the parent of a set of active leaves W in a tree of a forest, denoted by $p(W)$, as the lowest node in the tree that is a common ancestor of W and at least one other active node. (That is, $p(W)$ is undefined if there are no other active leaves in the tree that contains the leaves in W.) Note that the parent of a node is defined with respect to the current state of the algorithm, and not with respect to the initial input tree. If $W = \{u\}$ is a singleton, we will also use the notation $p(u) = p(\{u\})$. For a given tree or forest T_i', for $i \in \{1,2\}$, we use the notation $p_i(W)$ to denote $p(W)$ in T_i'.

[3] This is essentially the formulation that is common in the literature, except that in order to ensure that *only leaves have labels*, we give the label ρ to a new leaf that is an immediate descendent of the new root in both trees, rather than to the new root itself.

The operation in the algorithm that deletes edges from forest T_i' is *cut off a subset of leaves* W in T_i'. The edge that is deleted by this operation is the edge directly below $p_i(W)$ towards W (provided $p_i(W)$ is defined). Note that this means that the algorithm maintains the property that each internal node has a path to at least one leaf in $\mathcal{L}$. This ensures that the number of trees with leaves in $\mathcal{L}$ in T_i' is equal to the number of edges cut in T_i' plus 1. It also ensures that the only leaves (nodes with outdegree 0) are the nodes in $\mathcal{L}$.

We will call two leaves u and v *a sibling pair* or *siblings* in a forest, if they belong to the same tree in the forest, and they are the only two leaves in the subtree rooted at the lowest common ancestor $\text{lca}(u, v)$. Similarly, u and v are an *active* sibling pair in a forest, if they belong to the same tree in the forest, and are the only two *active* leaves in the subtree rooted at the lowest common ancestor $\text{lca}(u, v)$ (an equivalent definition is that $p(u) = p(v)$ in the forest).

If leaves u and v are an active sibling pair in both T_1' and T_2', we *merge* one of the leaves (say u) with the other (v). This means that from that point on v represents the subtree containing both u and v, instead of just the leaf v itself. This is accomplished by just making u inactive. Note that this merge operation can be performed recursively, where one or both of the leaves involved in the operation can already be leaves that represent subtrees. It is not hard to see that the subtree that is represented by an active leaf v is one of the two subtrees rooted at the child of $p(v)$, namely the subtree that contains v.

If leaves u and v are not active siblings in T_2' (and they are active siblings in T_1'), we can choose to *cut off an active subtree* between leaves u and v. To describe this operation, let $W_1, W_2, \ldots, W_k$ be the active leaves of the active trees that would be created by deleting the path between u and v (both the nodes and the edges) in T_2'. Note that $p_2(W_\ell)$ is on the path between u and v for all $\ell \in \{1, 2, \ldots, k\}$, because u and v are not active siblings. Cutting off an active subtree between leaves u and v now means cutting off any such a set W_ℓ.

The algorithm is given in Algorithm 1. The boxed expressions refer to updates of the dual solution which will be discussed in Section 3.2.2. These expressions are only necessary for the analysis of the algorithm.

▶ **Theorem 1.** *Algorithm 1 is a 3-approximation algorithm for the Maximum Agreement Forest problem.*

The proof of this theorem is given in the next subsection. It is clear the algorithm can be implemented to run in polynomial time. In Section 3.2.1, we show that the algorithm returns an agreement forest and we show that the number of edges deleted from T_2 by the algorithm can be upper bounded by three times the objective value of a feasible solution to the dual of a linear programming (LP) relaxation of MAF.

3.2 Analysis of the 3-Approximation Algorithm

3.2.1 Correctness

We need to show that the algorithm outputs an agreement forest. The trees of T_1' and T_2' each give a partitioning of $\mathcal{L}$, and clearly any internal node v belongs to $V[L]$ for at most one set in the partitioning. It remains to show that the two forests give the *same* partitioning of $\mathcal{L}$ and that each set in the partitioning is compatible.

The algorithm ends with all trees in T_1' and T_2' being inactive, and the algorithm maintains that the set of leaves represented by an active leaf u (i.e., the leaves that were merged with u (recursively), and u itself) form the leaf set of a subtree in both T_1' and T_2'. To be precise, it is the subtree rooted at one of the children of $p(u)$, namely the subtree that contains u.

1 $\boxed{y_v \leftarrow 0 \text{ for all } v \in V.}$
2 **while** *there exist at least 2 active leaves* **do**
3 Find an active sibling pair u, v in the active tree in T_1'.
4 **if** *u or v is the only active leaf in its tree in T_2'* **then**
5 Cut off this node (say u) in T_1' as well and make it inactive. $\boxed{y_u \leftarrow 1.}$
6 **else**
7 **if** *u and v are active siblings in T_2'* **then**
8 Merge u and v (i.e., make u inactive to "merge" it with v).
9 **else**
10 **if** *u and v are in the same tree in T_2', and this tree contains an active leaf w such that $uv|w$ in T_2* **then**
11 Cut off an active subtree W between u and v in T_2'. $\boxed{\text{Decrease } y_{p_2(W)} \text{ by 1.}}$
12 **end if**
13 Cut off u and cut off v in T_1' and T_2' and make them inactive.
$\boxed{y_u \leftarrow 1,\ y_v \leftarrow 1, \text{ decrease } y_{\mathrm{lca}_1(u,v)} \text{ by 1.}}$
14 **end if**
15 **end if**
16 **end while**
17 Make the remaining leaf (say v) inactive. $\boxed{y_v \leftarrow 1.}$

Algorithm 1: A 3-Approximation for Maximum Agreement Forest. The boxed text refers to updates of the dual solution as discussed in Section 3.2.2.

Furthermore note that this leaf set is compatible. This is easily verified by induction on the number of merges.

When u is the only active leaf in its tree in both forests, then the trees containing u in the two forests are thus guaranteed to have the same, compatible, set of leaves. Now, an inactive tree is created exactly when both T_1' and T_2' have an active tree in which some u is the only active leaf (lines 5, 13 and 17), and thus the two forests indeed induce the same partition of $\mathcal{L}$ into compatible sets.

3.2.2 Approximation Ratio

In order to prove the claimed approximation ratio, we will construct a feasible dual solution to the dual of the relaxation of the ILP given in Section 2. The dual LP is given in Figure 1(a). The dual LP has an optimal solution in which $0 \leq y_v \leq 1$ for all $v \in \mathcal{L}$. The fact that $\{v\}$ is a compatible set implies that $y_v \leq 1$ must hold for every $v \in \mathcal{L}$. Furthermore, note that changing the equality constraints of the primal LP to $\geq$-inequalities does not change the optimal value, and hence we may assume $y_v \geq 0$ for $v \in \mathcal{L}$.

It will be convenient for our analysis to rewrite this dual by introducing additional variables for every (not necessarily compatible) set of labeled leaves. We will adopt the convention to use the letter A to denote a set of leaves that is not necessarily compatible, and the letter L to denote a set of leaves that is compatible (i.e., $L \in \mathcal{C}$). The dual LP can then be written as in Figure 1(b). Any solution to this new LP can be transformed into a solution to the original dual LP by, for each A such that $z_A > 0$, taking some leaf $v \in A$ and setting $y_v = y_v + z_A$ and $z_A = 0$. This is feasible because the left-hand side of the first family of inequalities will not increase for any compatible set L, and it will decrease for L such that $A \cap L \neq \emptyset$ and $v \notin L$. Conversely, a solution to the original dual LP is feasible for this new LP by setting $z_A = 0$, for every set of labeled leaves A.

$$\begin{array}{lll} \max & \sum_v y_v - 1, & \\ \text{s.t.} & \sum_{v \in V[L]} y_v \leq 1 & \forall L \in \mathcal{C}, \\ & y_v \leq 0 & \forall v \in V \setminus \mathcal{L}. \end{array}$$

(a) Dual LP

$$\begin{array}{lll} \max & \sum_v y_v + \sum_{A \subseteq \mathcal{L}} z_A - 1, & \\ \text{s.t.} & \sum_{v \in V[L]} y_v + \sum_{A : A \cap L \neq \emptyset} z_A \leq 1 & \forall L \in \mathcal{C}, \\ & y_v \leq 0 & \forall v \in V \setminus \mathcal{L}, \\ & y_v \geq 0 & \forall v \in \mathcal{L}, \\ & z_A \geq 0 & \forall A \subseteq \mathcal{L}. \end{array}$$

(b) Reformulated dual LP

Figure 1 The dual of the LP relaxation for the ILP given in Section 2. The reformulated dual LP will be referred to as (D).

We will refer to the left-hand side of the first family of constraints, i.e., $\sum_{v \in V[L]} y_v + \sum_{A : A \cap L \neq \emptyset} z_A$, as the *load* on set L.

▶ **Definition 2.** The dual solution associated with a forest T_2', obtained from T_2 by edge deletions, active leaf set $\mathcal{L}'$, and variables $y = \{y_v\}_{v \in V}$ is defined as (y, z) where $z_A = 1$ exactly when A is the active leaf set of a tree in T_2', and 0 otherwise.

We will sometimes use "the dual solution" to refer to the dual solution associated with $T_2', \mathcal{L}'$ and y when the forest, active leaf set and y-values are clear from the context.

▶ **Lemma 3.** *After every execution of the while-loop of Algorithm 1, the dual solution associated with T_2', $\mathcal{L}'$, and y is a feasible solution to (D) and the objective value of this solution is at least $\frac{1}{3}|E(T_2) \setminus E(T_2')|$.*

Proof. We prove the lemma by induction on the number of iterations. Initially, $z_{\mathcal{L}} = 1$, and all other variables are equal to 0. Clearly, this is a feasible solution with objective value 0.

Observe that the dual solution maintained by the algorithm satisfies that $y_u = 0$ while u is active. Therefore, if there is a single active leaf u in a tree in T_2', then making this leaf u inactive and setting $y_u = 1$ does not affect dual feasibility and the dual objective value, since making u inactive decreases $z_{\{u\}}$ from 1 to 0. Also note that merging two active leaves (and thus making one of the two leaves inactive), replaces the set of active leaves A in an active tree in T_2' by a smaller set $A' \subset A$, with $A' \neq \emptyset$. Hence, the dual solution changes from having $z_A = 1$ to having $z_{A'} = 1$, which clearly does not affect dual feasibility or the dual objective value. Hence, we only need to verify that the dual solution remains feasible and its objective increases sufficiently for operations of the algorithm that cut edges from T_2', i.e., lines 11 and 13.

In line 11, one edge is cut in T_2', $y_{p_2(W)}$ decreases by 1. Let A be the set of active leaves in the tree containing W in T_2' before cutting off W. z_A decreases by 1, $z_{A \setminus W}$ increases by 1, z_W increases by 1. The only sets L for which the left-hand side potentially increases are sets L so that $W \cap L \neq \emptyset$ and $(A \setminus W) \cap L \neq \emptyset$. However, $p_2(W) \in V[L]$ for such sets L, and since $y_{p_2(W)}$ is decreased by 1, the load is not increased for any compatible set L. The dual objective is unchanged, but will change in line 13 of the algorithm, as we will show next.

In line 13, let A_u be the set of active leaves in the tree in T_2' containing u at the start of line 13 in the algorithm, and A_v be the set of active leaves in the tree in T_2' containing v. Note that $A_u \setminus \{u, v\} \neq \emptyset$: if $v \notin A_u$, then this holds because otherwise we would execute line 5, and if $v \in A_u$, then this holds because u, v are not active siblings at the start of line 11, and if u, v became active siblings after executing line 11, then the condition for line 11 implies that there exists $w \in A_u$ such that $uv|w$ in T_2.

The fact that $A_u \setminus \{u, v\} \neq \emptyset$ (and, by symmetry $A_v \setminus \{u, v\} \neq \emptyset$) implies that the total value of $\sum_A z_A + y_u + y_v$ increases by 2. Since we also decrease $y_{\mathrm{lca}_1(u,v)}$ by 1 the total increase in the objective of the dual solution by line 13 is 1. Also, in lines 11 and 13, a total of at most three edges are cut in T_2'.

It remains to show that executing line 13 does not make the dual solution (y, z) infeasible. Note that for each active set A' with $z_{A'} = 1$ before cutting off u and v, there is exactly one unique active subset $A \subseteq A'$ with $z_A = 1$ after cutting off u and v. Therefore the total value of $\sum_{A:A\cap L\neq\emptyset} z_A$ does not increase after cutting off u and v for any $L \subseteq \mathcal{L}$.

For any L that includes one of u, v, and at least one other active leaf, it must be the case that $\mathrm{lca}_1(u, v) \in V[L]$, because all active leaves are in one tree in T_1', and u and v were active siblings in T_1' at the start. Hence the only compatible sets L for which the load on L potentially increases by 1 because of an increase in $\sum_{x\in L} y_x$ are sets L that include both u and v. We discern two cases.

Case 1: An active subtree W was cut off in line 11. In this case, the load on L was decreased by 1 in line 11, compensating for the increase in line 13: $V[L]$ contains all nodes on the path between u and v in T_2, and hence also $p_2(W)$. It cannot contain a leaf $x \in W$, because $\{u, v, x\}$ form an inconsistent triplet (because $uv|x$ in T_1).

Case 2: No active subtree W was cut off in line 11. In this case, the value of $\sum_{A:A\cap L\neq\emptyset} z_A$ is decreased by at least 1: If u and v are in the same tree in T_2' before cutting off u and v, then this tree contains no leaves x such that $uv|x$ in T_2 since otherwise an active subtree W would have been cut off. Hence, L does not contain any active leaf x in the active tree that remains after cutting off u and v in T_2', since any such leaf x does not have $uv|x$ in T_2 and therefore forms an inconsistent triplet with u and v. Since L does contain active leaves in the tree containing u and v in T_2' before cutting off u and v (namely, u and v themselves), the value of $\sum_{A:A\cap L\neq\emptyset} z_A$ indeed decreases by 1.

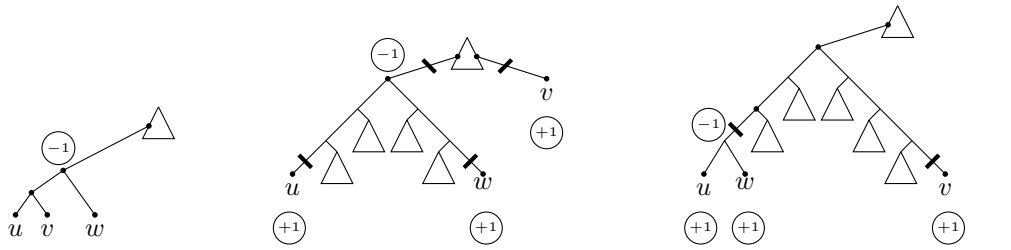
If u and v are not in the same tree in T_2' before cutting off u and v, then a similar argument holds. Since T_2' is obtained from T_2 by deleting edges, at least one of the two active trees containing u and v contains no leaves x such that $uv|x$ in T_2. Without loss of generality, suppose that this holds for the tree containing u. Then, L does not contain any active leaves in the active tree remaining after u is cut off, and hence $\sum_{A:A\cap L\neq\emptyset} z_A$ decreases by at least 1. ◀

By weak duality, we have that the objective value of any feasible solution to (D) provides a lower bound on the objective value of any feasible solution to the LP relaxation of our ILP for MAF, and hence also on the optimal value of the ILP itself. Theorem 1 thus follows from Lemma 3 and the correctness shown in Section 3.2.1.

4 Overview of the 2-Approximation Algorithm

In this section, we begin by giving an outline of the key ideas of our 2-approximation algorithm. We then give an overview of the complete algorithm that we call the "Red-Blue Algorithm".

One of the main ideas behind our 2-approximation is the consideration of the following two "essential" cases. The first "essential" case is the case where we have an active sibling pair u, v in T_1' that are (i) active siblings in T_2', or (ii) in different trees in T_2', or (iii) the tree in T_2' containing u, v does *not* contain an active leaf w such that $uv|w$ in T_2. Then, it is easy to verify, using the arguments in the proof of Lemma 3, that Algorithm 1 "works": it



(a) Subtree in T_1' **(b)** Case (I)(a): Subtree in T_2' **(c)** Case (I)(b): Subtree in T_2'

Figure 2 Dealing with an inconsistent triplet: Circled values denote the y-variables that are set by the algorithm, and bold diagonal lines denote edges that are cut (deleted from T_2'). Triangles denote subtrees with active leaves (that may be empty). Note that there is a distinction between edges that are incident to the root of a subtree represented by a triangle, and edges that are incident to some internal node of the subtree. The latter edges are connected to a dot on the triangle.

increases the dual objective value by at least half of the increase in $|E(T_2) \setminus E(T_2')|$. We will say such a sibling pair u, v is a "success".

The second "essential" case is the case where, in our current forest T_1', there is a subtree containing exactly three active leaves, say u, v, w, where $uv|w$ in T_1, and $\{u, v, w\}$ is an inconsistent triplet; assume without loss of generality that $uw|v$ in T_2, and that the first "essential" case does not apply; in particular, this implies that u, v are in the same tree in T_2'. It turns out that such an inconsistent triplet can be "processed" in a way that allows us to increase the objective value of the dual solution in such a way that it "pays for" half the increase in the number of edges cut from T_2'. There are a number of different cases to consider depending on whether all three leaves u, v, w are in the same tree in T_2' (case I) or not (case II), and whether the tree in T_2' containing w has an active leaf x such that $xw|u$ in T_2' (case (a)) or not (case (b)).

Figure 2 gives an illustration of T_1' and some possible configurations for T_2'. Consider for example case (I)(b). Since $\{u, v, w\}$ is an inconsistent triplet, it is not hard to see that any solution to MAF either has v as a singleton component, or either u or w must be a singleton component. Indeed, we can increase the dual objective by 1, by updating the y-values as indicated by the circled values in Figure 2 (a) and (b). The bold diagonal lines denote two edges that are cut (deleted from T_2'). Similar arguments can be made for the other cases.

Unfortunately, neither of the essential cases may be present in the forests T_1', T_2', and therefore the ideas given above may not be applicable. However, they do work if we generalize our notions. First, we generalize the notion of "active sibling pair in T_1'".

▶ **Definition 4.** A set of active leaves U is an *active sibling set* in T_1' if the leaves in U are the only active leaves in the subtree of T_1' rooted at $\text{lca}_1(U)$. U is a *compatible active sibling set* in T_1' if U is an active sibling set in T_1' that contains no inconsistent triplets.

Note that we will only use the term compatible active sibling set for T_1', and never for T_2'. We will therefore sometimes omit the reference to T_1', and simply talk about a "compatible active sibling set".

We similarly generalize the notion of a subtree in T_1' containing exactly three active leaves that form an inconsistent triplet.

▶ **Definition 5.** A set of active leaves $R \cup B$ is a *minimal incompatible active sibling set* in T_1' if $R \cup B$ is incompatible, R and B are compatible active sibling sets in T_1', and $p_1(R) = p_1(B)$.

The Red-Blue Algorithm now proceeds as follows: it begins by identifying a minimal incompatible active sibling set $R \cup B$ in T_1'. Such a set can be found by checking if the active leaf sets of the left and right subtrees of the root are compatible sets. If yes, then either all active leaves are compatible, or we have found a minimal incompatible active set. If not, then the active leaf set of one of the subtrees is incompatible, and we recurse on this subtree until we find a node in T_1' for which the active leaf sets of the left and right subtrees form a minimal incompatible set $R \cup B$. Note that we can assume $\text{lca}_2(R) = \text{lca}_2(R \cup B)$.

The algorithm will then "distill" R by repeatedly considering sibling pairs u, v in R, and executing operations similar to those in Algorithm 1, except that only one of u and v becomes inactive (and a bit more care has to be taken in certain cases). Procedure 1 gives the procedure ResolvePair the algorithm uses for handling a sibling pair u, v.

```
1  if u and v are in different trees in T_2' then
2      Relabel u and v if necessary so that lca_2(u,v) is not in the tree containing u in T_2'.
3      if the tree containing u in T_2' has other active leaves not in U then
4          FinalCut: Cut off u in T_1' and T_2' and make it inactive. [y_u ← 1.]
5      else
6          Cut off u in T_1' and make it inactive. [y_u ← 1.]
7      end if
8  else
9      if u and v are active siblings in T_2' then
10         Merge u and v (i.e., make u inactive to "merge" it with v).
11     else
12         Relabel u and v if necessary so that p_2(u) ≠ lca_2(u,v).
13         Cut off an active subtree W between u and v by cutting the edge below p_2(u) that is
               not on the path from u to v. [Decrease y_{p_2(u)} by 1.]
14         if u and v are now active siblings in T_2' then
15             Merge-After-Cut: Merge u and v (i.e., make u inactive to "merge" it with v).
                   [y_u ← 1.]
16         else
17             Cut off u in T_1' and T_2' and make u inactive. [y_u ← 1.]
18         end if
19     end if
20 end if
```

Procedure 1: ResolvePair(u, v)

Arguments similar to those in Section 3 show that ResolvePair maintains dual feasibility, *provided that we initially reduce* $y_{\text{lca}_1(R)}$ *by* 1. It is also not hard to verify that ResolvePair increases the dual objective by at least half the increase in the primal objective, and the only thing that is therefore needed to show that the algorithm is a 2-approximation is that we can "make up for" the initial decrease of the dual objective caused by decreasing $y_{\text{lca}_1(R)}$. Let us define the operation of "distilling" R as starting by reducing $y_{\text{lca}_1(R)}$ by 1, and then repeatedly finding a pair of active leaves u, v in R which are siblings in T_1' and executing ResolvePair(u, v) until only two active leaves $\hat{u}, \hat{v}$ in R remain. Since all other leaves in R are inactive, $\hat{u}$ and $\hat{v}$ form an active sibling pair in T_1'.

If pair $\hat{u}, \hat{v}$ is a "success" or if line 4 or 15 was executed at least once during the distilling of R, then there exists an operation that makes at least one of $\hat{u}, \hat{v}$ inactive and updates the dual, so that the total increase in the primal objective is at most twice the total increase in the dual objective caused by the processing of pairs in R. Procedure 2 gives the complete description of the procedure that, if successful, "resolves" set R (and will return "Success"):

▶ **Lemma 6.** *If* ResolveSet(R) *returns* Success *then at least one leaf in* R *became inactive, and the increase in the primal objective* $|E(T_2) \setminus E(T_2')|$ *caused by the procedure is at most twice the increase in the dual objective.*

Lemma 12 of the full version [12] contains a more precise formulation of this lemma.

```
21 Decrease y_{lca_1(R)} by 1.
22 while there exist at least three active leaves in R do
23 |  Find u, v in R that form an active sibling pair in T_1'.
24 |  ResolvePair(u, v).
25 end while
26 Let û, v̂ be the remaining active leaves in R.
27 if û and v̂ are active siblings in T_2' then
28 |  Merge û and v̂ (i.e., make û inactive to "merge" it with v̂). y_û ← 1.
29 |  Return Success.
30 else if (û and v̂ are in different trees in T_2') or (the tree containing û and v̂ does not contain
     an active leaf w such that ûv̂|w in T_2) then
31 |  Cut off û in T_2' (if û's tree contains at least one other active leaf) and in T_1' and make û
   |    inactive. y_û ← 1.
32 |  Cut off v̂ in T_2' (if v̂'s tree contains at least one other active leaf) and in T_1' and make v̂
   |    inactive. y_v̂ ← 1.
33 |  Return Success.
34 else if (At least one FinalCut or Merge-After-Cut was executed in some call to ResolvePair
     in the course of the current ResolveSet procedure) then
35 |  ResolvePair(û, v̂).
36 |  Cut off the last active leaf v̂ in U in T_2' and in T_1' and make v̂ inactive. y_v̂ ← 1.
37 |  Return Success.
38 else
39 |  Return Fail.
40 end if
```

Procedure 2: ResolveSet(R)

If Lemma 6 applies, we have made progress (since we have made at least one leaf inactive), and we will have paid for the increase in the primal objective $|E(T_2) \setminus E(T_2')|$ caused by the procedure by twice the increase in the dual objective.

Otherwise, the last active pair of leaves $\hat{u}, \hat{v}$ in R remain active, and we will have a "deficit" in the sense that the increase in the dual objective is at most half the increase in the primal objective *plus* 1. In this case, we similarly distill B by repeatedly calling ResolvePair(u, v) for pairs u, v in B that are active siblings in T_1' until only a single active leaf in B remains. However, we will show that in order to retain dual feasibility, we do not need to start the distilling of B by decreasing $y_{\text{lca}_1(B)}$ (which would give a total "deficit" of 2), but that we can "move" the initial decrease of $y_{\text{lca}_1(R)}$ to instead decrease $y_{\text{lca}_1(R \cup B)}$. Lemma 13 in the full version [12] shows that this indeed preserves dual feasibility.

Once R and B have both been "distilled", we are left with $\hat{u}, \hat{v}, \hat{w}$ that are an inconsistent triplet and form the active leaf set of a subtree in T_1'. We show how to deal with the triplet $\{\hat{u}, \hat{v}, \hat{w}\}$ (in ways similar to those in Figure 2) and we prove that in the entire processing of $R \cup B$, we have increased the dual objective by half of the number of edges we cut from T_2'.

Algorithm 2 gives an overview of the "Red-Blue Algorithm". It first calls a procedure Preprocess, which executes simple operations that do not affect the primal or dual objective:

merging two leaves if they are active siblings in both forests, and cutting off and deactivating a leaf in T_1' if it is the only active leaf in its tree in T_2'. At the end of an iteration, the Red-Blue algorithm needs to consider different cases for the final triplet. The description of these subroutines can be found in [12].

Set $T_1' \leftarrow T_1$, $T_2' \leftarrow T_2$, $\mathcal{L}' \leftarrow \mathcal{L}$. $y_u \leftarrow 0$ for all $u \in \mathcal{L}$.
PREPROCESS.
while $\mathcal{L}' \neq \emptyset$ **do**
  Find a minimal incompatible active sibling set $R \cup B$, with $\mathrm{lca}_2(R) = \mathrm{lca}_2(R \cup B)$.
  if RESOLVESET(R) *returns* FAIL **then**
    Decrease $y_{\mathrm{lca}_1(R \cup B)}$ by 1, and increase $y_{\mathrm{lca}_1(R)}$ by 1.
    while *there exist at least two active leaves in* B **do**
      Find u, v in B that form an active sibling pair in T_1'.
      RESOLVEPAIR(u, v).
    end while
    Let $\hat{r}_1, \hat{r}_2 \in R$ and $\hat{b} \in B$ be the remaining active leaves.
    Consider three different cases depending on whether $\hat{r}_1, \hat{r}_2$ and $\hat{b}$ are in one, two or three different trees in T_2' (see Section 6.1 and 6.2 in [12] for details).
  end if
  PREPROCESS.
end while

Algorithm 2: Red-Blue Algorithm for Maximum Agreement Forest

▶ **Theorem 7.** *The Red-Blue Algorithm is a 2-approximation algorithm for the Maximum Agreement Forest problem.*

5 Implementation Details

We implemented the Red-Blue approximation algorithm in Java, and tested it on instances with $|\mathcal{L}| = 2000$ leaves that were generated as follows: the number of leaves in the left subtree is set equal to a number between 1 and $|\mathcal{L}| - 1$ drawn uniformly at random, and a subset of this size is chosen uniformly at random from the label set. Then this procedure recurses until it arrives at a subtree with only 1 leaf – this will be the whole subtree.

After generating T_1 as described above, the tree T_2 was created by doing 50 random Subtree Prune-and-Regraft operations (where random means that the root of the subtree that is pruned was chosen uniformly at random, as well as the edge which is split into two edges, so that the new node created can be the parent of the pruned subtree, under the conditions that this is a valid SPR-operation). This construction allows us to deduce an upper bound of 50 on the optimal value. Our algorithm finds a dual solution that in 44% of the 1000 runs is equal to the optimal dual solution, and in 37% of the runs is 1 less than the optimal solution. The observed average approximation ratio is about 1.92. After running our algorithm, we run a simple greedy search algorithm which repeatedly looks for two trees in the agreement forest that can be merged (i.e., such that the resulting forest is still a feasible solution to MAF). The solution obtained after executing the greedy algorithm decreases the observed approximation ratio to less than 1.28. The code is available at `http://frans.us/MAF`.

6 Conclusion

Our algorithm and analysis raise a number of questions. First of all, although we believe that, conceptually, our algorithm is quite natural, the actual algorithm is complicated, and it would be interesting to find a simpler 2-approximation algorithm. Secondly, it is clear that our algorithm can be implemented in polynomial time, but the exact order of the running time is not clear. The bottleneck seems to be the finding of a minimal incompatible active sibling set, although it may be possible to implement the algorithm in a way that simultaneously processes sibling pairs as in ResolvePair, while it is looking for a minimal incompatible active sibling set.

Acknowledgements. We thank Neil Olver and Leen Stougie for fruitful discussions.

References

1 Benjamin L. Allen and Mike Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5(1):1–15, 2001.

2 Maria Luisa Bonet, Katherine St John, Ruchi Mahindru, and Nina Amenta. Approximating subtree distances between phylogenies. *Journal of Computational Biology*, 13(8):1419–1434, 2006.

3 Magnus Bordewich, Catherine McCartin, and Charles Semple. A 3-approximation algorithm for the subtree distance between phylogenies. *Journal of Discrete Algorithms*, 6(3):458–471, 2008. doi:10.1016/j.jda.2007.10.002.

4 Magnus Bordewich and Charles Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Ann. Comb.*, 8(4):409–423, 2004. doi:10.1007/s00026-004-0229-z.

5 Charles Darwin. *Notebook B: Transmutation of species (1837–1838)*. In: John van Wyhe: The Complete Work of Charles Darwin Online, 2002. URL: http://darwin-online.org.uk/.

6 Martin Farach and Mikkel Thorup. Optimal evolutionary tree comparison by sparse dynamic programming. In *FOCS'94: Proceedings of 35th Annual Symposium on Foundations of Computer Science*, pages 770–779. IEEE, 1994.

7 Martin Farach and Mikkel Thorup. Sparse dynamic programming for evolutionary-tree comparison. *SIAM Journal on Computing*, 26(1):210–230, 1997.

8 A. D. Gordon. A measure of the agreement between rankings. *Biometrika*, 66(1):7–15, 1979.

9 Jotun Hein, Tao Jiang, Lusheng Wang, and Kaizhong Zhang. On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71(1-3):153–169, 1996. doi:10.1016/S0166-218X(96)00062-5.

10 Estela M. Rodrigues. *Algoritmos para Comparação de Árvores Filogenéticas e o Problema dos Pontos de Recombinação*. PhD thesis, University of São Paulo, Brazil, 2003. Chapter 7, available at http://www.ime.usp.br/~estela/studies/tese-traducao-cp7.ps.gz.

11 Estela M. Rodrigues, Marie-France Sagot, and Yoshiko Wakabayashi. The maximum agreement forest problem: approximation algorithms and computational experiments. *Theoretical Computer Science*, 374(1-3):91–110, 2007. doi:10.1016/j.tcs.2006.12.011.

12 Frans Schalekamp, Anke van Zuylen, and Suzanne van der Ster. A duality based 2-approximation algorithm for maximum agreement forest. *CoRR*, abs/1511.06000, 2015. URL: http://arxiv.org/abs/1511.06000.

13 Feng Shi, Qilong Feng, Jie You, and Jianxin Wang. Improved approximation algorithm for maximum agreement forest of two rooted binary phylogenetic trees. *Journal of Combinatorial Optimization*, 2015. doi:10.1007/s10878-015-9921-7.

14 Mike Steel and Tandy Warnow. Kaikoura tree theorems: Computing the maximum agreement subtree. *Information Processing Letters*, 48(2):77–82, November 1993. doi:10.1016/0020-0190(93)90181-8.

15 Chris Whidden, Robert G. Beiko, and Norbert Zeh. Fixed-parameter algorithms for maximum agreement forests. *SIAM Journal on Computing*, 42(4):1431–1466, 2013. doi:10.1137/110845045.

16 Chris Whidden and Norbert Zeh. A unifying view on approximation and FPT of agreement forests. In *Algorithms in Bioinformatics*, volume 5724 of *Lecture Notes in Computer Science*, pages 390–402. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-04241-6_32.

17 Yufeng Wu. A practical method for exact computation of subtree prune and regraft distance. *Bioinformatics*, 25(2):190–196, 2009. doi:10.1093/bioinformatics/btn606.

18 Yufeng Wu and Jiayin Wang. Fast computation of the exact hybridization number of two phylogenetic trees. In *Bioinformatics Research and Applications*, volume 6053 of *Lecture Notes in Computer Science*, pages 203–214. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-13078-6_23.

Robust Assignments via Ear Decompositions and Randomized Rounding

David Adjiashvili[1], Viktor Bindewald[*2], and Dennis Michaels[3]

1 **Institute for Operations Research, ETH Zürich, Zürich, Switzerland**
addavid@ethz.ch
2 **Department of Mathematics, TU Dortmund University, Dortmund, Germany**
viktor.bindewald@math.tu-dortmund.de
3 **Department of Mathematics, TU Dortmund University, Dortmund, Germany**
dennis.michaels@math.tu-dortmund.de

Abstract

Many real-life planning problems require making a priori decisions before all parameters of the problem have been revealed. An important special case of such problem arises in scheduling and transshipment problems, where a set of jobs needs to be assigned to the available set of machines or personnel (resources), in a way that all jobs have assigned resources, and no two jobs share the same resource. In its nominal form, the resulting computational problem becomes the *assignment problem*.

This paper deals with the *Robust Assignment Problem* (RAP) which models situations in which certain assignments are *vulnerable* and may become unavailable after the solution has been chosen. The goal is to choose a minimum-cost collection of assignments (edges in the corresponding bipartite graph) so that if any vulnerable edge becomes unavailable, the remaining part of the solution contains an assignment of all jobs.

We develop algorithms and hardness results for RAP and establish several connections to well-known concepts from matching theory, robust optimization, LP-based techniques and combinations thereof.

1998 ACM Subject Classification G.2.2 Graph algorithms

Keywords and phrases robust optimization, matching theory, ear decomposition, randomized rounding, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.71

1 Introduction

The need for incorporating system reliability into decision making has sprung wide-spread interest in optimization models which incorporate data uncertainty in the last decades. The latter trend has lead to the development of several new theories including the popular field of *Robust Optimization*. In robust optimization the nominal optimization problem is equipped with a set of *scenarios*, representing various possible states of nature that may occur after the solution to the problem is chosen. The goal is to determine a solution that will perform well (in terms of feasibility, or cost) in the *worst case* realization of the state of nature.

The *Assignment Problem* is one of the most fundamental optimization problems arising in many reliability-sensitive systems. In its nominal form, the input consists of a set of

* Financial support through RTG 1855 of the second author by the German Research Foundation (DFG) is gratefully appreciated.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 71; pp. 71:1–71:14

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

n_T *tasks*, a set of n_R *resources* (with $n_T \leq n_R$), and *assignment costs* $c_{i,j}$ representing the cost associated with assigning resource i to task j. The set of allowed assignments can be represented by a bipartite graph $G := (R \dot\cup T, E)$ where each resource i corresponds to a node $r_i \in R$, each task j corresponds to a node $t_j \in T$, and the edge $\{r_i, t_j\}$ is present in E if the j-th task can be assigned to resource i. The goal is to find a *matching* $M \subseteq E$ of minimal cost that covers all nodes in T, i.e. a set of non-adjacent edges that is incident to every node in T. In the following, a subset M satisfying that property is called *an assignment*.

The Robust Assignment Problem (RAP) is the natural robust counterpart of the assignment problem and defined as follows[1]. An instance of RAP consists of an instance of the nominal assignment problem, i.e. of a bipartite graph $G = (R \dot\cup T, E)$ representing admissible assignments and a non-negative cost vector $c \in \mathbf{R}^E_{\geq 0}$, as well as a collection $F \subseteq E$ of *vulnerable edges*. Each $f \in F$ induces a failure scenario that leads to a deletion of f from G. The goal is to find a subset $X \subseteq E$ of minimal cost with the property that, for every vulnerable edge $f \in F$, the set $X \setminus \{f\}$ contains an assignment of G.

Intuitively, RAP asks to choose a redundant assignment, namely one that contains a feasible assignment, even when an arbitrary single vulnerable edge becomes unavailable. Therefore, the robustness paradigm considered in this paper falls into the topic of *redundancy-based robustness* – a well-motivated and widely studied approach (see e.g. [5, 23] for an overview of different robustness concepts). Some of the problems that fall into this category include the minimum k-edge connected spanning subgraph problem [12, 18] and the robust facility location problem [25, 31, 9]. More recently, Adjiashvili, Stiller and Zenklusen [2] introduced a robustness model called *bulk-robustness*, which combines the standard redundancy based robustness approach with a non-uniform failure model. In its general form, a bulk robust counterpart of a combinatorial optimization problem consists of an instance of the nominal problem, as well as a collection of scenarios, each comprising an arbitrary set of resources that may fail simultaneously. The goal is, as usual, to choose a minimum-cost set of resources that contains a feasible solution, even when the resources in any single failure scenario become unavailable. In the language of bulk-robustness, RAP is the bulk-robust assignment problem restricted to the case of where each scenario is composed of a single edge.

In the remainder of this section we provide a few motivating applications for RAP, establish some connections to related notions in matching theory and discuss results and technical contributions.

1.1 Motivation

The most natural applications of RAP, and redundancy-based robust optimization in general, emerge in situations where resources can not be easily made available on demand. In such applications, any resource that is intended for use at a certain point in time must be *reserved* at an earlier stage, and thus made available for potential deployment. Examples of such applications range from construction of robust power transmission networks [21] to supply chain management [32].

In a nutshell, redundancy-based models deal with the problem of choosing the optimal set of (potentially unreliable) resources to reserve, in order to guarantee that the available set of resources at the time of solution implementation, i.e. the reserved resources that did not fail, contains a feasible solution in every scenario. While we believe that RAP can be a

[1] Several other robust counterparts of the assignment problem have been considered in the literature under the same, or similar names. We review these models in Section 2.

useful model to incorporate robustness in any assignment model with up-front decisions of the latter type, we bring hereafter a few concrete applications.

Staff Training. Large companies often employ intensive training programs for their employees, designed to adapt the available pool of skills to their dynamic needs. For example, large software firms starting a new project involving new technologies, might need to train some employees to use these technologies. It is natural to incorporate the incurred training costs into the task assignment problem. The cost of assigning an employee to perform a given task in the project corresponds to the cost to train the employee to perform this task.

In a more realistic scenario, some employee to task assignments might become unavailable *even if the employee were trained to perform the task*. This type of vulnerability is very common, and can be caused, e.g. by employee dissatisfaction from his task assignment, or by unexpected inability to perform the task (due to injury or unavailability of equipment, etc.). RAP is a suitable model for deciding on robust training programs for the project, where skill sets of the employees allow for reassignments even if any single employee to task assignment becomes unavailable.

Continuity of Service. In industries such as health care and consulting it is often desirable to maintain very stable client to service provider relationships [7]. It is hence natural for a service provider to model the resulting resource allocation problem as an assignment problem, where an available pool of trained employees (nurses, consultants etc.) is matched to the set of customers. In the nominal variant, the company might want to minimize the cost incurred by the assignment, where the cost is computed as the total cost incurred by establishing all relationships in the assignment (establishing such relationships incurs significant costs).

It is however common that certain established relationships go off track in the course of a long interaction (e.g. due to customer or employee dissatisfaction). These relationships correspond to vulnerabilities of individual assignments. With RAP it is possible to account for such vulnerabilities by establishing a cost-effective set of relationships that, even if any nominal interaction becomes unavailable, the organization can quickly adjust the assignment to satisfy all clients.

1.2 Overview of results and techniques

This paper addresses the computational complexity of RAP. In particular, we present approximation algorithms and hardness of approximation results. We justify the study of approximation algorithms by showing that RAP is NP-hard even in very restricted variants. Due to space constraints we omit technical proofs as well as details on the complexity results.

The assignment problem has a well-known natural interpretation as a bipartite matching problem in the graph $G = (R \dot\cup T, E)$. It is hence also natural to view RAP as a robust version of the bipartite matching problem: find a minimum-cost set of edges in $M \subseteq E$ such that for every $f \in F$, the set $M \setminus \{f\}$ contains a matching incident to all nodes in T. Furthermore, if $|R| = |T|$, the problem becomes a robust variant of the *perfect* matching problem. We henceforth adopt this point of view, as it facilitates a clearer exposition of our results, and highlights an inherent connection between RAP and *matching-covered graphs*, a notion that we repeatedly use in our approximation algorithms.

The next statement shows that it suffices to consider RAP on balanced bipartite graphs, implying that we can state the feasibility condition for RAP using perfect matchings.

▶ **Proposition 1.** *Any* RAP *instance can be efficiently transformed to an equivalent weighted* RAP *instance with a balanced bipartite graph such that any α-approximation for the new instance can be used to efficiently construct an α-approximation of the original instance for all $\alpha \geq 1$.*

It is important to note that the transformation in Proposition 1 leads to an instance on a balanced bipartite graph that is equipped with a weighted cost function. Interpreting the resulting instance as an unweighted one may destroy the preservation of the transformation's approximation quality, thus Proposition 1 can not be used as a black box reduction for the unweighted case.

In the following, RAP refers to general instances of the robust assignment problem on balanced bipartite graphs with weighted cost function, while card-RAP is used for the unweighted version of RAP. We denote $n = n_T + n_R$ and $m = |E|$.

We remark that feasibility of a given RAP instance can be efficiently verified as one only needs to check, for each $f \in F$, whether the graph contains a perfect matching not using f. The latter can be done using any polynomial algorithms for finding maximum matchings in bipartite graphs.

1.2.1 Matching-Covered Graphs and Ear Decompositions

Our algorithmic results rely on a tight connection between RAP and *matching-covered graphs*, a well-known notion in matching theory. A graph $G = (V, E)$ is *matching-covered* if every edge $e \in E$ appears in some perfect matching of G.[2]

It turns out that inclusion-wise minimal solutions of any RAP instance are matching-covered as the following proposition states.

▶ **Proposition 2.** *A set $X \subseteq E$ is an inclusion-wise minimal feasible solution to* RAP *if and only if $(R \dot\cup T, X)$ is an inclusion-wise minimal non-empty graph with the properties of being matching-covered and that every isolated edge $e \in X$ is not vulnerable, i.e. $e \notin F$.*

Proposition 2 provides a very useful characterization of minimal solutions of RAP, as it allows us to use various results on matching-covered graphs in our algorithms for RAP. In particular, it allows us to identify feasible subgraphs and augment them to feasible solutions for the entire instance by adding structures that maintain the property of being matching-covered. One particularly useful tool is an *ear decomposition* of a bipartite matching-covered graph, i.e. a certain decomposition of G into edge-disjoint paths of odd length. In the following, we denote by $V[G]$ and $E[G]$ the set of nodes and edges of a graph G, respectively. For two graphs G and H we denote by $G + H$ their union $(V[G] \cup V[H], E[G] \cup E[H])$.

▶ **Definition 3** (Ear Decomposition of a Bipartite Graph)**.** Let H be a bipartite graph, and let H' be a subgraph of H. An *odd ear of H with respect to H'* is a path P in H with an odd number of edges and such that P and H' have exactly two nodes in common. Those two nodes form the end points of P, and belong to different parts of the bipartition.

A *bipartite ear decomposition* is a sequence $P_0, P_1, \ldots, P_q$ of paths in H, such that: (i) $P_0 = (\{v_1, v_2\}, \{\{v_1, v_2\}\})$ is a graph composed of a single edge; (ii) $H = P_0 + \cdots + P_q$; and (iii)µfor every $j = 1, \ldots, q$, the path P_j is an odd ear with respect to $H_{j-1} := P_0 + \cdots + P_{j-1}$.

[2] The notion of matching-covered graphs is originally introduced for connected graphs. In this paper we use this term also for disconnected graphs. Furthermore, note that some authors use synonymously the notion 1-*extendable* or in the bipartite case *elementary* (cf. [28]).

We exploit the following well-known connection between matching-covered bipartite graphs and bipartite ear decompositions.

▶ **Theorem 4** ([28, Thm. 4.1.6]). *A bipartite graph is matching-covered if and only if it has a bipartite ear decomposition.*

1.2.2 Results for card-RAP

Theorem 4 allows us to prove the following results for card-RAP. An instance of RAP is called *uniform* if every edge is vulnerable, i.e. if $F = E$.

▶ **Theorem 5.** card-RAP *admits a polynomial* 1.5*-approximation algorithm in the uniform case, and a* 3*-approximation algorithm in the general case.*

Our algorithm starts by producing an ear decomposition of the input graph. Then, it iteratively selects a certain subset of the edges to be part of the solution, by processing the ears in the decomposition in the order given by the decomposition, and omitting the edges corresponding to ears of length one.

We complement the latter algorithmic result by showing that card-RAP is NP-hard to approximate within some constant $\delta > 1$ even in the restricted case of a uniform scenario set, as stated in the following theorem.

▶ **Theorem 6.** *There exists a constant $\delta > 1$, such that there is no polynomial δ-approximation algorithm for the uniform* card-RAP*, unless* P=NP*.*

Theorems 5 and 6 imply that the true approximability thresholds for uniform card-RAP and card-RAP lie in the intervals $[\delta, 1.5]$ and $[\delta, 3]$, respectively.

To complete the complexity landscape of card-RAP we also consider the case of only two vulnerable edges. This special case comprises the simplest variant of card-RAP that is not equivalent to a standard matching problem[3]. In the following theorem we prove that already this special case is NP-hard, thus drawing a sharp threshold for polynomial solvability of card-RAP.

▶ **Theorem 7.** card-RAP *is* NP*-hard even when restricted to instances with two vulnerable edges, i.e. with $F = \{f_1, f_2\}$.*

To the best of our knowledge, this is the first example of an NP-hard robust counterpart of a polynomial optimization problem, with a constant number of vulnerable resources.[4] To prove Theorem 7 we first show NP-hardness of a problem of partitioning a graph into a cycle containing a given node and a matching whose union covers all nodes, so as to minimize the length of the cycle, a problem that might be interesting in its own right.

1.2.3 Results for RAP

Our main algorithmic result for RAP is a randomized $O(\log n)$-approximation for the general case, as stated hereafter.

[3] Observe that the case of a single vulnerable edge $F = \{f\}$ is solvable by reporting any minimum-cost perfect matching in the graph $(R \dot\cup T, E \setminus \{f\})$ as a solution.

[4] There are many examples of optimization problems that become NP-hard when the robust counterpart is allowed to contain a constant number of scenarios (see e.g. [26]). In all such examples, however, each scenario affects a *non-constant* number of resources.

▶ **Theorem 8.** *RAP admits a randomized polynomial $O(\log n)$-approximation algorithm.*

Our approximation algorithm for RAP builds upon our simple approximation algorithm for card-RAP in that it also iteratively constructs a solution maintaining the invariant that at any point in the algorithm, the edges selected so far form a matching-covered graph. It is however unclear how to arrive at the desired approximation for RAP relying only on properties of matching-covered graphs. We therefore combine the latter techniques with additional tools from linear programming (LP) theory and randomized rounding. Concretely, we start by solving an LP relaxation of RAP, derived from a natural integer linear programming (ILP) formulation of the problem. The obtained fractional solution is used to guide an iterative randomized procedure. In each iteration a fractional bipartite matching corresponding to part of the fractional solution is selected. A decomposition of this fractional matching into a convex combination of integral matchings is then used to randomly pick one matching, and a carefully selected subset of this matching is added to the current solution. To bound the quality it does not suffice to bound the number of iterations, or the expected number of times an edge is part of a candidate matching. Instead we use a discharging argument that assigns costs to nodes depending on the graph selected so far.

We complement our algorithmic results for RAP with hardness of approximation result with the same asymptotic bound, as stated hereafter.

▶ **Theorem 9.** *Provided that* NP $\not\subseteq$ DTIME$(n^{\log\log n})$*, the uniform* RAP *admits no* $c\log n$*-approximation for any* $c < 1$*. RAP unless P=NP.*

2 Related work

Redundancy-based robustness is a paradigm that motivates many well studied problems, including the minimum k-connected subgraph problem [18, 12, 30], survivable and robust network design problems [24, 10, 2, 1], robust clustering problems [25, 31], robust spanner problems [8, 15] and many more. All of the latter models bare a close resemblance to RAP: they assume resources to be vulnerable and ask to find a minimum-cost set of resources that contains a desired structure even in case any vulnerable resource, or set of resources, fails.

A relatively new approach to redundancy based robustness is the incorporation of non-uniform uncertainty sets [2, 1]. RAP is seen as a robust model of this type, as we allow both vulnerable and invulnerable edges in the same instance.

The study of robustness with respect to cost uncertainty was initially studied by Kouvelis and Yu [26], and Yu and Yang [33]. For a survey we refer to Aissi, Bazgan and Vanderpooten [4]. A closely related class of multi-budgeted problems has received considerable attention (see e.g. [19] and references therein). The latter works include variants of the related multi-objective matching problem.

Various variants of robust matching problems have been considered in the literature. Hassin and Rubinstein [22], and Fujita, Kobayashi and Makino [17] study the following notion of an α-robust matching. A perfect matching M in a weighted graph is α-robust (for $\alpha \in (0, 1]$), if for every $p \leq |M|$, the p heaviest edges of the matching have total weight at least α times the weight of a maximum weight matching of size p. Deineko and Woeginger [14] showed that the min-max-robust assignment problem with a fixed number of scenarios is equivalent to the *exact perfect matching problem*, a famous problem with unknown complexity status. In the case of a variable number of scenarios the min-max-robust problem is NP-hard, as was proved by Aissi, Bazgan and Vanderpooten [3]. Additional work on robust variants of the matching problem include models with recovery [16], models with node failures [27], and

the closely related matching interdiction problem [34]. Plesník [29] provided conditions under which an r-regular graph remains perfectly matchable after removing $r-1$ arbitrarily chosen edges. Brigham, Harary, Violin and Yellen [6] and Cheng, Lesniak, Lipman and Liptak [11] studied the minimum number of edges to be removed from a graph to arrive at a graph without a perfect matching.

3 Approximation Algorithms for RAP

3.1 $O(1)$-Approximation for card-RAP

Our algorithm relies on an ear decomposition of the underlying bipartite graph. Similar ideas using ear decompositions were successfully used to approximate various combinatorial optimization problems including, among others, the minimum edge connected subgraph problem and the path traveling salesman problem [12, 30]. Recall, that a balanced bipartite graph admits an ear decomposition (which is by no means unique) if and only if it is matching-covered.

As an initial pre-processing step, a given card-RAP instance consisting of $G = (R \dot\cup T, E)$ and F is transformed into a balanced instance. For this, we introduce a set D of $n_R - n_T$ dummy task nodes, and connect each such node to all nodes from R. Let E_D denote the set of newly introduced edges, and let $G_b = (R \dot\cup (T \cup D), E \cup E_D)$. In a second step we remove from G_b all *dispensable edges*, i.e. all edges not appearing in any perfect matching of G_b. This way, we obtain a graph, that is, by definition, matching-covered. Note that the second step can be implemented in polynomial time using any efficient algorithm for finding bipartite matchings. Moreover, we remark that omitting dispensable edges clearly does not change the underlying card-RAP instance, since such edges can be removed from any feasible solution without breaking feasibility. In the following, we allow some abuse of notation and call the new graph G_b as well. Next, we assume that G (and equivalently G_b) is feasible, i.e. there do not exist any isolated edges. If an isolated edge exists the algorithm terminates and reports that the instance is infeasible. Now let $G_b = P_0 + \cdots + P_q$ be any ear decomposition of G_b with the initial edge P_0 not covering a dummy node from D. We call an ear P_j *trivial* if it is not P_0 and if it consists of a single edge. The next lemma shows that a feasible solution to card-RAP can be obtained from the ear decomposition of G_b by skipping trivial ears.

▶ **Lemma 10.** *Let $J = \{j \in [q] \mid P_j$ is a trivial ear$\}$. Define $G'_b = P_0 + \sum_{i \in [q] \setminus J} P_i$, and $X := E[G'_b] \setminus E_D$. Then, the set X is a feasible solution to the* card-RAP *instance. Furthermore, $|X| \leq 3n_T$.*

Lemma 10 allows us to arrive at an approximation algorithm, summarized as Algorithm 1.

Algorithm 1 : $O(1)$-Approximation for card-RAP

Require: $G = (R \dot\cup T, E)$ and $F \subseteq E$.
Ensure: a feasible solution X to card-RAP on G and F
1: $X \leftarrow \emptyset$
2: Transform G into a balanced graph G_b and remove all dispensable edges
3: Compute an ear decomposition $G_b = P_0 + \ldots + P_q$
4: $X \leftarrow P_0 \cup \bigcup\{E[P_j] \mid P_j$ is not trivial, $j = 1, \ldots, q\}$
5: **return** X

Proof of Theorem 5. According to [13], an ear decomposition of a matching-covered graph can be computed in polynomial time. Furthermore, all other computations can also be

implemented efficiently, such that the running time of Algorithm 1 is polynomial. From Lemma 10, it follows that the set X returned by Algorithm 1 is feasible. For $F = E$, any feasible solution must have at least two edges incident to any node from the set T. Hence, $\mathrm{OPT} \geq 2n_T$. Since $|X| \leq 3n_T$, the approximation guarantee is indeed 1.5. If $F \subsetneq E$, then G can contain a perfect matching not including any edge from F. Thus, we can use $\mathrm{OPT} \geq n_T$ yielding an approximation factor of 3. ◀

3.2 $O(\log n)$-Approximation for RAP

In this section we provide a polynomial $O(\log n)$-approximation algorithm for RAP, thus proving Theorem 8. Again, we assume that the RAP instance is feasible. For a clean presentation, we first describe an algorithm for the uniform case $F = E$ and then explain how it can be extended to the non-uniform case.

Our algorithm is based on an LP-rounding procedure that works with a relaxation of the integer linear formulation of RAP, that is defined as follows. Let $G = (R \dot\cup T, E)$ be a balanced, bipartite graph and let $c \in \mathbf{R}^E_{\geq 0}$ be a non-negative cost vector. Moreover, let $P_G \subseteq \mathbf{R}^E$ denote the perfect matching polytope associated with G (i. e. P_G is the convex hull of all incidence vectors of perfect matchings in G). A standard ILP formulation of RAP contains the following variables: (i) $x^{-f} \in \{0,1\}^E$ representing a perfect matching in $G - f := (R \dot\cup T, E \setminus \{f\})$, for all $f \in F$, and (ii) $y \in \{0,1\}^E$ encoding a feasible solution to RAP. Then, RAP can be modeled as an ILP as follows.

$$
\begin{array}{lrcll}
\min & c^\top y & & & \\
\text{s.t.} & x^{-f} & \in & P_G \cap \{x \in \mathbf{R}^E \mid x_f = 0\}, & \text{for each } f \in F, \\
& y & \geq & x^{-f}, & \text{for each } f \in F, \\
& x^{-f} & \in & \{0,1\}^E, & \text{for each } f \in F, \\
& y & \in & \{0,1\}^E &
\end{array}
\tag{ILP}
$$

The LP-relaxation (LP) is obtained by relaxing all integrality constraints in (ILP). To keep notation short, we let $x \in \left(\mathbf{R}^E\right)^E$ be the vector with parts x^{-f}, $f \in F$. It is straightforward to verify that integer solutions to (ILP) coincide with feasible solutions to the RAP instance. In the following, we will denote, by $\chi^S \in \{0,1\}^E$, the incidence vector of a subset $S \subseteq E$.

Now, let (x, y) be a fractional solution to (LP). We describe hereafter a rounding procedure that yields an approximation for RAP. Consider some edge $f \in F$. Since x^{-f} is contained in $P_G \cap \{x \in \mathbf{R}^E \mid x_f = 0\}$, there exist positive scalars $\lambda_1^{-f}, \cdots, \lambda_k^{-f}$ with $\sum_{i\in[k]} \lambda_i^{-f} = 1$, and perfect matchings $M_1^{-f}, \cdots, M_k^{-f}$ in $G - f$ such that $x^{-f} = \sum_{i\in[k]} \lambda_i^{-f} \chi^{M_i^{-f}}$. By Caratheodory's theorem, there is a decomposition of the latter type with k bounded by $O(m) = O\left(n^2\right)$. Furthermore given x^{-f}, such a decomposition can be computed in polynomial time using polyhedral techniques [20, Thm. 6.5.11].

Our algorithm performs several iterations of randomized rounding based on the latter decomposition of fractional matchings. More precisely, at each iteration, an infeasible set $X \subseteq E$ of edges, that was chosen so far, is augmented with an additional set M of edges chosen randomly as follows. First, an arbitrary edge f is chosen from E among all edges not yet covered by X, i.e. among all $e' \in E$ such that the edge set X selected so far contain no perfect matching that does not include e'. Next, a decomposition of the vector x^{-f} as a convex combination of perfect matchings is computed, as above. This decomposition is then used to select a single perfect matching $\bar{M}$ from $\{M_1^{-f}, \cdots, M_k^{-f}\}$ *randomly*, where M_i^{-f} is chosen with probability λ_i^{-f} for all $i \in [k]$. Finally, the augmenting set $M \subseteq \bar{M}$ is chosen to contain all edges of $\bar{M}$ connecting *distinct connected components* of X. The edges of M

are added to X and the algorithm proceeds to the next iteration. The algorithm terminates when X is a feasible solution. A summary of the algorithm is presented as Algorithm 2.

To prove the correctness of the algorithm we resort again to properties of matching-covered graphs. Concretely, as a main ingredient of the proof of Lemma 12, which states a useful structural property of intermediate solutions in the algorithm, we use the following classic result.

Algorithm 2 : Randomized $O(\log n)$-Approximation for RAP

Require: $G = (R \dot\cup T, E)$ with $|R| = |T|$, and $c \in \mathbf{R}^E_{\geq 0}$.
Ensure: A feasible solution X to RAP on G with $F = E$ and cost vector c.
1: Solve (LP) to obtain an optimal solution (x, y)
2: $X \leftarrow \emptyset$
3: **while** X is infeasible **do**
4: Select an edge $f \in F$ such that $X \setminus \{f\}$ contains no perfect matching
5: Compute a decomposition of x^{-f} as $x^{-f} = \sum_{i=1}^{k} \lambda_i^{-f} \chi^{M_i^{-f}}$ and select one matching $\bar{M} \in \{M_i^{-f} \mid i \in [k]\}$ with $\Pr\left[\bar{M} = M_i^{-f}\right] = \lambda_i^{-f}$ for all $i \in [k]$
6: Add to X all edges from $\bar{M}$ that connect distinct connected components in $(R \dot\cup T, X)$
7: **end while**
8: **return** X

▶ **Theorem 11** ([28, Thm. 4.1.1., p. 122]). *A connected bipartite graph $H = (U \dot\cup W, E)$ with $|U \dot\cup W| \geq 4$ is matching-covered if and only if for any $u \in U$ and $w \in W$ the graph $H - \{u\} - \{w\}$ has a perfect matching.*

▶ **Lemma 12.** *Let X be a non-empty set of edges already selected in an arbitrary iteration of Algorithm 2. Then, the graph $G[X] := (R \dot\cup T, X)$ is matching-covered.*

Proof. As X is assumed to be non-empty, X contains at least one perfect matching of G. Thus, $G[X]$ does not have isolated nodes.

Now, let $S \subseteq R \dot\cup T$ be the nodes of some connected component of $G[X]$. It suffices to prove that the graph $(S, X[S])$, with $X[S] := \{e \in X \mid e = \{s_1, s_2\}, \text{ for some } s_1, s_2 \in S\}$, is matching-covered. For $|S| = 2$, $X[S]$ contains exactly one edge that belongs to a perfect matching in X. Thus, the claim is proved.

Next, assume that $|S| > 2$. To prove that $(S, X[S])$ is matching-covered, we proceed by induction on the number of iterations in Algorithm 2. In the first iteration, a perfect matching is added to X in Step 6, thus the claim holds in that case.

Now, let $X' \subseteq X$ be the set of edges selected until the beginning of the iteration preceding the current iteration. By the induction hypothesis, we can assume that every connected component of $(R \dot\cup T, X')$ is matching-covered.

To prove the claim we need to show that every edge $e \in X[S]$ is contained in some perfect matching of S. If $e \in X'$, this claim holds by the inductive hypothesis, and due to $X' \subseteq X$. In case that $e \notin X'$, we have that $e \in \bar{M}$, where $\bar{M}$ is selected in Step 5 in the current iteration. This means that e connects nodes from two distinct connected components of $(R \dot\cup T, X')$.

Now, pick any cycle $C \subseteq X$ in $G[X]$ containing e with a minimum number of edges from $\bar{M}$. Let $D_1, \cdots, D_l \subseteq R \dot\cup T$ be the components in $(R \dot\cup T, X')$ that have edges in C. From minimality of $|C \cap \bar{M}|$ it follows that C is a simple cycle (i.e. each node is contained in at most two of its edges) and that each component D_j, $j = 1, \cdots, l$ contains exactly two nodes incident to the cycle. This cycle can now be used to demonstrate the existence of the desired

perfect matching M' as follows. First, include in M' all edges in $C \cap \bar{M}$. Then, in every component D_j for $j = 1, \cdots, l$ pick a matching covering all nodes, except the two nodes incident to the cycle C. Due to Theorem 11, such a matching exists since each component D_j is matching-covered. The matching chosen so far covers exactly the nodes in $D_1 \cup \cdots \cup D_l$. Finally, pick any matching covering all other components of $(R \dot\cup T, X')$ that are not incident to C. This matching exists, since again, $(R \dot\cup T, X')$ is matching-covered. The result is a perfect matching in $G[X]$ containing e, which completes the proof. ◀

Lemma 12 guarantees that at every stage in the algorithm, the only edges not yet covered by the current solution X are the isolated edges of $G[X]$. Now, since at an iteration where an uncovered edge f is chosen in Step 4, the set M must contain two edges distinct from f, that are incident to the endpoints of f, the edge f is guaranteed to be covered in the end of this iteration. This immediately implies that the algorithm terminates with a feasible solution after at most $|E|$ iterations. It hence remains to bound the expected cost of the solution returned by Algorithm 2.

▶ **Theorem 13.** *The expected cost of the solution returned by Algorithm 2 is* $O(\log n) \cdot \mathrm{OPT}$, *where* OPT *is the optimal solution value for the* RAP *instance.*

Proof. The feasibility of the obtained solution and the bound on the running time are guaranteed by Lemma 12.

For a set $Q \subseteq E$ of edges we denote by $c_{\mathrm{LP}}(Q)$ the contribution of the edges in Q to the LP cost, i.e. $c_{\mathrm{LP}}(Q) = \sum_{e \in Q} c_e y_e$. For a node $v \in R \dot\cup T$, we denote by $\delta(v) \subseteq E$ the set of edges incident to v. To bound the approximation guarantee we bound the expectation of the ratio $c(X)/c_{\mathrm{LP}}(E)$. Since the LP is a relaxation of the problem we have $c_{\mathrm{LP}}(E) \leq \mathrm{OPT}$. Thus, this ratio is a valid bound on the approximation guarantee.

To obtain the bound we design a scheme that charges every selected edge in any stage of the algorithm to one of its endpoints. We then show that the expected cost charged to any node $v \in V$ is bounded by $O(\log n)$ times the *fractional cost* $c_{\mathrm{LP}}(\delta(v))$ associated with the node. This then implies that the expected cost of all edges added by the algorithm is at most

$$O(\log n) \cdot \sum_{v \in R \dot\cup T} c_{\mathrm{LP}}(\delta(v)) \leq O(\log n) \cdot \mathrm{OPT},$$

where the last inequality follows from linearity of expectation, $c_{\mathrm{LP}}(E) \leq \mathrm{OPT}$ and $c_{\mathrm{LP}}(E) = \frac{1}{2} \sum_{v \in R \dot\cup T} c_{\mathrm{LP}}(\delta(v))$.

We describe next how the costs of the selected edges are charged to the nodes of the graph. Let $\bar{X} \subseteq E$ be the set of edges returned by the algorithm. Formally, with each node $v \in R \dot\cup T$ we associate a collection of edges $R_v \subseteq \bar{X}$ such that $\bigcup_{v \in R \dot\cup T} R_v = \bar{X}$ and such that $c(R_v)$ is bounded by $O(\log n)$ times the fractional load at v in expectation.

The sets R_v are constructed as follows. In the beginning $R_v = \emptyset$ for all $v \in R \dot\cup T$. Let X be the set of edges selected so far by the algorithm and let $M \subseteq E \setminus X$ be the set of edges selected to be added to X in Step 6 of the current iteration. At this stage, the sets R_v might already contain some edges. We describe how these sets change as a result of the selection of M. Consider an edge $f = \{r, t\} \in M$. Recall that the algorithm only includes edges in the solution if they connect different connected components in $(R \dot\cup T, X)$. Thus, r and t lie in different connected components. Let D_r and D_t be the node sets of the connected components to which r and t belong, respectively, and assume without loss of generality that $|D_r| \leq |D_t|$. Then, f is charged to r, i.e. f is included in R_r. In other words, an edge added by the algorithm in any iteration is charged to the node contained in the *smaller connected component*, with ties broken arbitrarily.

It is obvious that the latter scheme charges all edges in $\bar{X}$ to some nodes, such that $\bigcup_{v \in V} R_v = \bar{X}$ holds in the end of the last iteration. It remains to analyze the quantity $c(R_v)$ for a single node $v \in R \dot{\cup} T$. The bound on $c(\bar{X})$ will then follow from linearity of expectation and the previous discussion. To arrive at the desired bound it suffices to make the following two observations.

First, at any time, if an edge is charged to v, its expected cost is at most $c_{\text{LP}}(\delta(v))$. Indeed, recall that the edges in M come from a perfect matching chosen at random from the decomposition of some fractional perfect matching x^{-f} in the graph (this x^{-f} corresponds to the edge f chosen in Step 4 in the current iteration). Let this decomposition be

$$x^{-f} = \sum_{i \in [k]} \lambda_i^{-f} \chi_{M_i^{-f}}.$$

The distribution over the integral matchings defining x^{-f} induces a distribution over the edges incident to v: each edge $e \in \delta(v)$ is contained in the perfect matching with a probability $p_e \in [0,1]$ given by

$$p_e = \sum_{i \in [k] \,:\, e \in M_i^{-f}} \lambda_i^{-f} = x_e^{-f}.$$

Since $x_e^{-f} \leq y_e$ for all $e \in E$ we have that the expected cost of the edge charged to v is at most $\sum_{e \in \delta(v)} c_e x_e^{-f} \leq c_{\text{LP}}(\delta(v))$, proving the first property.

The second observation concerns the number of times the node v is charged in the course of the algorithm. Consider any iteration in which some edge was charged to v, and let D_v be the nodes in the component of v in the beginning of the iteration. Since we always charge an edge to the node in the smaller component, and since charged edges always merge connected components, the size of the connected component containing v in the end of the iteration is *at least* $2|D_v|$. Since the graph only contains n nodes, this doubling can only happen at most $\log n$ times.

We conclude that $c(R_v)$ is, in expectation, indeed at most $O(\log n) c_{LP}(\delta(v))$, which concludes the proof of the theorem. ◀

Lemma 12 and Theorem 13 imply the correctness of Theorem 8 for the uniform case. The generalization to the non-uniform case is explained in the proof of Theorem 8, which we bring next.

Proof of Theorem 8. It remains to show how to treat the case $F \neq E$. For this, we provide a transformation to reduce such an instance to a uniform instance by losing only a factor of 2 in the approximation guarantee.

The transformation first adds to the graph one parallel edge $\bar{e}$ for every $e \notin F$. Let G' be the obtained graph. The new set of vulnerable edges is set to $F' = E[G']$. Solutions for the two RAP instances are in obvious correspondence: A solution $X \subseteq E[G]$ to the original instance can be transformed to a solution for the new instance of at most double the cost by taking $X' = X \cup \{\bar{e} \mid e \in X \setminus F\}$. Conversely, a solution X' for the new instance can be transformed to a solution for the new instance with the same, or better cost, by choosing $X = X' \cap E[G]$.

Let OPT' denote the optimal solution value of the transformed uniform instance. Our $O(\log n)$-approximation algorithm for the general case proceeds by first transforming the instance to a uniform instance of RAP, as above, then invoking Algorithm 2 to obtain the set X', having expected cost at most $O(\log n)\text{OPT}' = O(\log n)\text{OPT}$ and then returning $X = X' \cap E[G]$. ◀

We conclude this section by arguing why simpler randomized rounding techniques, for instance, ones that lead to logarithmic approximation to many covering problems, are unlikely to lead to a similar result for RAP. The reason for this is that there does not seem to be a simple way to obtain a compact set cover-type representation of RAP without losing a super-logarithmic factor in the approximation guarantee. One natural attempt could be to consider every vulnerable edge $f \in F$ as an *element* that needs to be covered, and every possible perfect matching $M \subseteq E$ that does not contain f, a *covering set* that covers the edge f (and all other edges in $F \setminus M$). The cost of the covering set is simply the sum of the costs of edges in the corresponding perfect matching. Unfortunately, it is easy to come up with examples in which the optimal solution value in the latter set covering model has cost $\Omega(n)\text{OPT}$. Such instances can be constructed, for example by choosing an instance, such that any feasible solution must have some nodes with very high degree, while an optimal solution has cost $O(n)$.

4 Conclusion and Future Work

This paper studies a novel practically relevant robust variant of the assignment problem (RAP). We showed tight connections between RAP and classical notions in matching theory, including matching-covered graphs and ear decompositions, and used these connections to obtain asymptotically tight approximation results for RAP. In our approximation algorithm for the general variant of RAP we combined classical results for matching-covered graphs with LP randomized rounding techniques.

Some ongoing and future work includes the following lines of research. Study a version of RAP with node failures, or with a combination of node and edge failures. This problem has many potential applications beyond the ones listed here. Study the variant of RAP where each scenario consists of at most k edges, for some input parameter $k > 1$. This paper treats the case $k = 1$. Besides, it is interesting to study the complexity of RAP in general graphs.

Acknowledgement. The work of the second author and the third author is part of the Research Training Group *"Discrete Optimization of Technical Systems under Uncertainty"* (RTG 1855).

References

1 D. Adjiashvili. Non-uniform robust network design in planar graphs. In *Proc. of APPROX*, pages 61–77, 2015.

2 D. Adjiashvili, S. Stiller, and R. Zenklusen. Bulk-robust combinatorial optimization. *Math Program*, 149(1-2):361–390, 2014.

3 H. Aissi, C. Bazgan, and D. Vanderpooten. Complexity of the min–max and min–max regret assignment problems. *Oper Res Lett*, 33(6):634–640, 2005.

4 H. Aissi, C. Bazgan, and D. Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *Eur J Oper Res*, 197(2):427–438, 2009.

5 D. Bertsimas, D.B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM review*, 53:464–501, 2011.

6 R.C. Brigham, F. Harary, E.C. Violin, and J. Yellen. Perfect-matching preclusion. *Congressus Numerantium*, 174:185–192, 2005. 00042.

7 G. Carello and E. Lanzarone. A cardinality-constrained robust model for the assignment problem in Home Care services. *Eur J Oper Res*, 236(2):748–762, 2014.

8 S. Chechik, M. Langberg, D. Peleg, and L. Roditty. Fault-tolerant spanners for general graphs. In *Proc. of STOC*, pages 435–444, 2009.

9 S. Chechik and D. Peleg. Robust fault tolerant uncapacitated facility location. In *Proc. of STACS*, pages 191–202, 2010.

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

11 E. Cheng, L. Lesniak, M.J. Lipman, and L. Liptak. Conditional matching preclusion sets. *Information Sciences*, 179(8):1092–1101, 2009. 00039.

12 J. Cheriyan, A. Sebo, and Z. Szigeti. Improving on the 1.5-Approximation of a Smallest 2-Edge Connected Spanning Subgraph. *SIAM J Discrete Math*, 14(2):170–180, 2001.

13 M.H. de Carvalho and J. Cheriyan. An O(VE) algorithm for ear decompositions of matching-covered graphs. In *Proc. of SODA*, pages 415–423, 2005.

14 V.G. Deineko and G.J. Woeginger. On the robust assignment problem under a fixed number of cost scenarios. *Oper Res Lett*, 34(2):175–179, 2006.

15 M. Dinitz and R. Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proc. of PODC*, pages 169–178. ACM, 2011.

16 M.C. Dourado, D. Meierling, L.D. Penso, D. Rautenbach, F. Protti, and A.R. de Almeida. Robust recoverable perfect matchings. *Networks*, 66(3):210–213, 2015.

17 R. Fujita, Y. Kobayashi, and K. Makino. Robust matchings and matroid intersections. In *Proc. of ESA*, pages 123–134. Springer, 2010.

18 H. N. Gabow, M. X. Goemans, É. Tardos, and D. P. Williamson. Approximating the smallest k-edge connected spanning subgraph by LP-rounding. In *Proc. of SODA*, pages 562–571, 2005.

19 F. Grandoni, R. Ravi, M. Singh, and R. Zenklusen. New approaches to multi-objective optimization. *Math Program*, 146(1-2):525–554, 2014.

20 M. Grötschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Algorithms and Combinatorics. Springer Berlin Heidelberg, 1993.

21 M. Hajiaghayi, N. Immorlica, and V.S. Mirrokni. Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks. In *Proc. of MobiCom*, pages 300–312. ACM, 2003.

22 R. Hassin and S. Rubinstein. Robust matchings. *SIAM J Discrete Math*, 15(4):530–537, 2002.

23 W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *Eur J Oper Res*, 165(2):289–306, 2005.

24 K. Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21:39–60, 2001.

25 K. Jain and V.V. Vazirani. An approximation algorithm for the fault tolerant metric facility location problem. In *Proc. of APPROX*, pages 177–183, 2000.

26 P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media, 1997.

27 P. Laroche, F. Marchetti, S. Martin, and Z. Roka. Bipartite complete matching vertex interdiction problem: Application to robust nurse assignment. In *Proc. of CoDIT*, pages 182–187, 2014.

28 L. Lovász and M.D. Plummer. *Matching theory*. North-Holland, Amsterdam, 1986.

29 J. Plesník. Connectivity of regular graphs and the existence of 1-factors. *Matematický časopis*, 22(4):310–318, 1972.

30 A. Sebő and J. Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-tsp, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 5(34):597–629, 2014.

31 C. Swamy and D. B. Shmoys. Fault-tolerant facility location. In *Proc. of SODA*, pages 735–736, 2003.

32 C.S. Tang. Robust strategies for mitigating supply chain disruptions. *International Journal of Logistics: Research and Applications*, 9(1):33–45, 2006.

33 G. Yu and J. Yang. On the robust shortest path problem. *Computers & Operations Research*, 25(6):457–468, 1998.

34 R. Zenklusen. Matching interdiction. *Discrete Appl Math*, 158(15):1676–1690, 2010.

Closing the Gap for Makespan Scheduling via Sparsification Techniques* †

Klaus Jansen[1], Kim-Manuel Klein[2], and José Verschae[3]

1 **Department of Computer Science, University of Kiel, Kiel, Germany**
kj@informatik.uni-kiel.de
2 **Department of Computer Science, University of Kiel, Kiel, Germany**
kmk@informatik.uni-kiel.de
3 **Facultad de Matemáticas and Escuela de Ingeniería, Pontificia Universidad Católica de Chile, Santiago, Chile**
jverschae@uc.cl

Abstract

Makespan scheduling on identical machines is one of the most basic and fundamental packing problem studied in the discrete optimization literature. It asks for an assignment of n jobs to a set of m identical machines that minimizes the makespan. The problem is strongly NP-hard, and thus we do not expect a $(1+\varepsilon)$-approximation algorithm with a running time that depends polynomially on $1/\varepsilon$. Furthermore, Chen et al. [3] recently showed that a running time of $2^{(1/\varepsilon)^{1-\delta}} + \text{poly}(n)$ for any $\delta > 0$ would imply that the Exponential Time Hypothesis (ETH) fails. A long sequence of algorithms have been developed that try to obtain low dependencies on $1/\varepsilon$, the better of which achieves a running time of $2^{\tilde{O}(1/\varepsilon^2)} + O(n \log n)$ [10]. In this paper we obtain an algorithm with a running time of $2^{\tilde{O}(1/\varepsilon)} + O(n \log n)$, which is tight under ETH up to logarithmic factors on the exponent.

Our main technical contribution is a new structural result on the *configuration-IP*. More precisely, we show the existence of a highly symmetric and sparse optimal solution, in which all but a constant number of machines are assigned a configuration with small support. This structure can then be exploited by integer programming techniques and enumeration. We believe that our structural result is of independent interest and should find applications to other settings. In particular, we show how the structure can be applied to the minimum makespan problem on related machines and to a larger class of objective functions on parallel machines. For all these cases we obtain an efficient PTAS with running time $2^{\tilde{O}(1/\varepsilon)} + \text{poly}(n)$.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases scheduling, approximation, PTAS, makespan, ETH

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.72

1 Introduction

Minimum makespan scheduling is one of the foundational problems in the literature on approximation algorithms [6, 7]. In the *identical machine* setting the problem asks for an assignment of a set of n jobs $\mathcal{J}$ to a set of m identical machines $\mathcal{M}$. Each job $j \in \mathcal{J}$ is

* Full version: http://arxiv.org/abs/1604.07153.

† This work was partially supported by DFG Project "Entwicklung und Analyse von effizienten polynomiellen Approximationsschemata für Scheduling- und verwandte Optimierungsprobleme," Ja 612/14-2, by FONDECYT project 3130407, and by Núcleo Milenio Información y Coordinación en Redes ICM/FIC RC130003.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 72; pp. 72:1–72:13

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

characterized by a non-negative processing time $p_j \in \mathbb{Z}_{>0}$. The load of a machine is the total processing time of jobs assigned to it, and our objective is to minimize the *makespan*, that is, the maximum machine load. This problem is usually denoted $P||C_{\max}$. It is well known to admit a *polynomial time approximation scheme* (PTAS) [9], and there has been many subsequent works improving the running time or deriving PTAS's for more general settings. The fastest PTAS for $P||C_{\max}$ achieves a running time of $2^{O(1/\varepsilon^2)\log^3(1/\varepsilon))} + O(n \log n)$ for $(1+\varepsilon)$-approximate solutions [10]. Very recently, Chen et al. [3] showed that, assuming the *exponential time hypothesis* (ETH), there is no PTAS that yields $(1+\varepsilon)$-approximate solutions for $\varepsilon > 0$ with running time $2^{(1/\varepsilon)^{1-\delta}} + \text{poly}(n)$ for any $\delta > 0$ [3].

Given a guess $T \in \mathbb{N}$ on the optimal makespan, which can be found with binary search, the problem reduces to deciding the existence of a packing of the jobs to m machines (or bins) of capacity T. If we aim for a $(1+\varepsilon)$-approximate solution, for some $\varepsilon > 0$, we can assume that all processing times are integral and T is a constant number, namely $T \in O(1/\varepsilon^2)$. This can be achieved with well known rounding and scaling techniques [1, 2, 8] which will be specified later. Let $\pi_1 < \pi_2 < \ldots < \pi_d$ be the job sizes appearing in the instance after rounding, and let b_k denote the number of jobs of size π_k. The mentioned rounding procedure implies that the number of different job sizes is $d = O((1/\varepsilon)\log(1/\varepsilon))$. Hence, for large n we obtain a highly symmetric problem where several jobs will have the same processing time. Consider the *knapsack polytope* $\mathcal{P} = \{c \in \mathbb{R}^d_{\geq 0} : \pi \cdot c \leq T\}$. A packing on one machine can be expressed as a vector $c \in Q = \mathbb{Z}^d \cap \mathcal{P}$, where c_k denotes the number of jobs of size π_k assigned to the machine. Elements in $Q = \mathbb{Z}^d \cap \mathcal{P}$ are called *configurations*. Considering a variable $x_c \in \mathbb{Z}_{\geq 0}$ that decides the multiplicity of configuration c in the solution, our problem reduces to solving the following linear integer program (ILP):

$$\text{[conf-IP]} \quad \sum_{c \in Q} c \cdot x_c = b, \tag{1}$$

$$\sum_{c \in Q} x_c = m, \tag{2}$$

$$x_c \in \mathbb{Z}_{\geq 0} \qquad \text{for all } c \in Q. \tag{3}$$

In this article we derive new insights on this ILP that help us to design faster algorithms for $P||C_{\max}$ and other more general problems. These including makespan scheduling on *related machines* $Q||C_{\max}$, and a more general class of objective functions on parallel machines. We show that all these problems admit a PTAS with running time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + \text{poly}(n)$. Hence, our algorithm is best possible up to polylogarithmic factors in the exponent assuming ETH [3].

1.1 Literature Review

There is an old chain of approximation algorithms for $P||C_{\max}$, starting from the seminal work by Graham [6, 7]. The first PTAS was given by Hochbaum and Shmoys [9] and had a running time of $(n/\varepsilon)^{O((1/\varepsilon)^2)} = n^{O((1/\varepsilon)^2 \log(1/\varepsilon))}$. This was improved to $n^{O((1/\varepsilon)\log^2(1/\varepsilon))}$ by Leung [14]. Subsequent articles improve further the running time. In particular Hochbaum and Shmoys (see [8]) and Alon et al. [1, 2] obtain an *efficient PTAS*[1] (EPTAS) with running time $2^{(1/\varepsilon)^{\text{poly}(1/\varepsilon)}} + O(n \log n)$. Alon et al. [1, 2] consider general techniques that work for

[1] That is, a PTAS whose running time is $f(1/\varepsilon)\text{poly}(|I|)$ where $|I|$ is the encoding size of the input and f is some function.

several objective functions, including all L_p-norm of the loads and maximizing the minimum machine load.

The previously fastest PTAS for $P||C_{\max}$ achieves a running time of $2^{O((1/\varepsilon)^2 \log^3(1/\varepsilon))} + O(n \log n)$ [10]. More generally, this work gives an EPTAS for the case of related (uniform) machines, where each machine $i \in \mathcal{M}$ has a speed s_i and assigning to i job j implies a processing time of p_j/s_i. For this more general case the running time is $2^{O((1/\varepsilon)^2 \log^3(1/\varepsilon))} + \text{poly}(n)$. For the simpler case of $P||C_{\max}$, the ILP can be solved directly since the number of variables is a constant. This can be done with Lentras' algorithm [13], or even with Kannan's algorithm [12] that gives an improved running time. This technique yields a running time that is doubly exponential in $1/\varepsilon$. This was, in essence, the approach by Alon et al. [1, 2] and Hochbaum and Shmoys [8]. To lower the dependency on $1/\varepsilon$, Jansen [10] uses a result by Eisenbrand and Shmonin [4] that implies the existence of a solution x with support of size at most $O(d \log(dT)) = O((1/\varepsilon) \log^2(1/\varepsilon))$. First guessing the support and then solving the ILP with $O((1/\varepsilon) \log^2(1/\varepsilon))$ integer variables and using Kannan's algorithm yields the desired running time of $2^{O((1/\varepsilon)^2 \log^3(1/\varepsilon))} + O(n \log n)$.

The configuration ILP has recently been studied in the context of the (1-dimensional) cutting stock problem. In this case, the dimension d is constant, $T = 1$, and π is a rational vector. Moreover, π and b are part of the input. Goemans and Rothvoß [5] obtain an optimal solution in time $\log(\Delta)^{2^{O(d)}}$, where Δ is the largest number appearing in the denominator of π_k or the multiplicities b_k. This is achieved by first showing that there exists a pre-computable set $\tilde{Q} \subseteq Q$ with polynomial many elements, such that there exists a solution x that gives all but constant (depending only on d) amount of weight to $\tilde{Q}$. We remark that applying this result to a rounded instance of $P||C_{\max}$ yields a running time that is doubly exponential on $1/\varepsilon$.

1.2 Our Contributions

Our main contribution is a new insight on the structure of the solutions of [conf-IP]. These properties are specially tailored to problems in which T is bounded by a constant, which in the case of $P||C_{\max}$ can be guaranteed by rounding and scaling. The same holds for $Q||C_{\max}$ with a more complex rounding and case analysis.

We first classify configurations by their support. We say that a configuration is *simple* if its support is of size at most $\log(T+1)$, otherwise it is *complex*. Our main structural result[2] states that there exists a solution x in which all but $O(d \log(dT))$ weight is given to simple configurations, the support is bounded by $O(d \log(dT))$ (as implied by Eisenbrand and Shmonin [4]) and no complex configuration has weight larger than 1.

▶ **Theorem 1** (Thin solutions). *Assume that [conf-IP] is feasible. Then there exists a feasible solution x to [conf-IP] such that:*

1. *if $x_c > 1$ then the configuration c is simple,*
2. *the support of x satisfies $|\text{supp}(x)| \leq 4(d+1)\log(4(d+1)T)$, and*
3. *$\sum_{c \in Q_c} x_c \leq 2(d+1)\log(4(d+1)T)$, where Q_c denotes the set of complex configurations.*

[2] We remark the resemblance of this structure to the result by Goemans and Rothvoß [5]. Indeed, similarly to their result, we can precompute a subset of configurations such that all but a constant amount of weight of the solution is given to such set. In their case the set is of cardinality polynomial on the input and is constructed by covering the integral solutions of the knapsack polytope by parallelepipeds. In our case, all but $O(d \log dT)$ weight is given to simple configurations.

We call a solution satisfying the properties of the theorem *thin*. The theorem can be shown by iteratively applying a sparsification lemma that shows that if a solution gives a weight of two or more to a complex configuration, then we can replace this partial solution by two configurations with smaller support. The sparsification lemma is shown by a simple application of the pigeonhole principle. The theorem can be shown by mixing this technique with the theorem of Eisenbrand and Shmonin [4] and a potential function argument.

As an application to our main structural theorem, we derive a PTAS for $P||C_{\max}$ by first guessing the jobs assigned to complex configurations. An optimal solution for this subinstance can be derived by a dynamic program. For the remaining instance we know the existence of a solution using only simple configurations. Then we can guess the support of such solution and solve the corresponding [conf-IP] restricted to the guessed variables. The main use of having simple configurations is that we can guess the support of the solution much faster, as the number of simple configuration is (asymptotically) smaller than the total number of configurations. The complete procedure takes time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + O(n \log n)$. Moreover, using the rounding and case analysis of Jansen [10], we derive an mixed integer linear program that can be suitably decomposed in order to apply our structural result iteratively. This yields a PTAS with a running time of $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + \text{poly}(n)$ for $Q||C_{\max}$.

Similarly, we can extend our results to derive PTAS's for a larger family of objective functions as considered by Alon et al. [1, 2]. Let ℓ_i denote the load of machine i, that is, the total processing time of jobs assigned to machine i for a given solution. Our techniques then gives a PTAS with the same running time for the problem of minimizing the L_p-norms of the loads (for fixed p), and maximizing $\min_{i \in M} \ell_i$, among others. To solve this problem, we can round the instance and state an IP analogous to [conf-IP] but considering an objective function. However, the objective function prevents us to use the main theorem as it is stated. To get over this issue, we study several ILPs. In each ILP we consider x_c to be a variable only if c has a given load, and fix the rest to be some optimal solution. Applying to each such ILP Theorem 1, plus some extra ideas, yields an analogous structural theorem. Afterwards, an algorithm similar to the one for makespan minimization yields the desired PTAS.

From an structural point of view, our sparsification lemma has other consequences on the structure of the knapsack polytope and the LP-relaxation of the [conf-IP]. More precisely, we can show that any vertex of the convex hull of Q must be simple. This, for example, helps us to upper bound the number of vertices by $2^{O(\log^2(T)+\log^2(d))}$. Moreover, we can show that the configuration-LP, obtained by replacing the integrality restriction in [conf-IP] by $x \geq 0$, if it is feasible then admits a solution whose support consist purely of simple configurations. Due to space limitations we leave many details and proofs to the full version.

2 Preliminaries

We will use the following notation throughout the paper. By default $\log(\cdot) = \log_2(\cdot)$, unless stated otherwise. Given two sets A, I, we will denote by A^I the set of all vectors indexed by I with entries in A, that is, $A^I = \{(a_i)_{i \in I} : a_i \in A \text{ for all } i \in I\}$. Moreover, for $A \subseteq \mathbb{R}$, we denote the support of a vector $a \in A^I$ as $\text{supp}(a) = \{i \in I : a_i \neq 0\}$.

We consider an arbitrary knapsack polytope $\mathcal{P} = \{c \in \mathbb{R}^d_{\geq 0} : \pi \cdot c \leq T\}$ where $\pi \in \mathbb{Z}^d_{>0}$ is a non-negative integral (row) vector and T is a positive integer. We assume without loss of generality that each coordinate π_k of π is upper bounded by T (otherwise $c_k = 0$ for all $c \in \mathbb{Z}^d \cap \mathcal{P}$). We focus on the set of integral vectors in $\mathcal{P}$ which we denote by $Q = \mathbb{Z}^d \cap \mathcal{P}$. We call an element $c \in Q$ a *configuration*. Given $b \in \mathbb{R}^d$, consider the problem of decomposing b as a conic integral combination of m configurations. That is, our aim is to find a feasible solution to [conf-IP], defined above.

A crucial property of the [conf-IP] is that there is always a solution with a support of small cardinality. This follows from a Caratheodory-type bound obtained by Eisenbrand and Shmonin [4]. Since we will need the argument later, we state the result applied to our case and revise its (very elegant) proof. We split the proof in two lemmas.

For a given subset $A \subseteq Q$, let us denote by x^A the indicator vector of A, that is $x^A_c = 1$ if $c \in A$, and 0 otherwise. Let us also denote by M the $(d+1) \times |Q|$ matrix that defines the system of equalities (1) and (2).

▶ **Lemma 2** (Eisenbrand and Shmonin [4]). *Let $x \in \mathbb{Z}^Q_{\geq 0}$ be a vector such that $|\operatorname{supp}(x)| > 2(d+1)\log(4(d+1)T)$. Then there exist two disjoint sets A, B with $\emptyset \neq A, B \subseteq \operatorname{supp}(x)$ such that $Mx^A = Mx^B$.*

▶ **Lemma 3** (Eisenbrand and Shmonin [4]). *If [conf-IP] is feasible, then there exists a feasible solution x such that $|\operatorname{supp}(x)| \leq 2(d+1)\log(4(d+1)T)$.*

Proof. Let x be a solution to [conf-IP] that minimizes $|\operatorname{supp}(x)| = s$. Assume by contradiction that $s > 2(d+1)\log(4(d+1)T)$. We show that we can find another solution x' to [conf-IP] with $|\operatorname{supp}(x')| < |\operatorname{supp}(x)|$, contradicting the minimality of $|\operatorname{supp}(x)|$. By Lemma 2, there exist two disjoint subsets $A, B \in \operatorname{supp}(x)$ such that $Mx^A = Mx^B$. Moreover, let $\lambda = \min\{x_c : c \in A\}$. Vector $x' := x - \lambda x^A + \lambda x^B$ is also a solution to [conf-IP] and has a strictly smaller support since a configuration $c^* \in \arg\min\{x_c : c \in A\}$ satisfies $x'_{c^*} = 0$. ◀

3 Structural Results

Recall that we call a configuration c simple if $|\operatorname{supp}(c)| \leq \log(T+1)$ and complex otherwise. An important observation to show Theorem 1 is that if c is a complex configuration, then $2c$ can be written as the sum of two configurations of smaller support. This is shown by the following Sparsification Lemma.

▶ **Lemma 4** (Sparsification Lemma). *Let $c \in Q$ be a complex configuration. Then there exist two configurations $c_1, c_2 \in Q$ such that*

1. $\pi \cdot c_1 = \pi \cdot c_2 = \pi \cdot c$,
2. $2c = c_1 + c_2$,
3. $\operatorname{supp}(c_1) \subsetneq \operatorname{supp}(c)$ *and* $\operatorname{supp}(c_2) \subsetneq \operatorname{supp}(c)$.

Proof. Consider for each subset $S \subseteq \operatorname{supp}(c)$, a configuration $c^S \in Q$ such that $c^S_i = c_i$ if $i \in S$ and $c^S = 0$ otherwise. As the number of subsets of $\operatorname{supp}(c)$ is $2^{|\operatorname{supp}(c)|}$, and $c^R \neq c^S$ if and only if $R \neq S$, the collection of vectors $V := \{c^S : S \subseteq \operatorname{supp}(c)\}$ has cardinality $|V| = 2^{|\operatorname{supp}(c)|}$.

On the other hand, for any vector $c^S \in V$ it holds that $\pi \cdot c^S \leq \pi \cdot c \leq T$. Hence, $\pi \cdot c^S \in \{0, 1 \ldots, T\}$ can take only $T+1$ different values. Using that c is a complex configuration and hence $2^{|\operatorname{supp}(c)|} > 2^{\log(T+1)} = T+1$, the pigeonhole principle ensures that there are two different non-empty configurations $c^S, c^R \subseteq V$ with $\pi \cdot c^S = \pi \cdot c^R$. By removing the intersection, we can assume w.l.o.g. that S and R have no intersection. We define $c_1 = c - c^S + c^R$ and $c_2 = c - c^R + c^S$, which satisfy the properties of the lemma as

$$\pi \cdot c_1 = \pi \cdot c - \pi \cdot c^S + \pi \cdot c^R = \pi \cdot c \quad \text{and}$$
$$2c = c - c^S + c^R + c - c^R + c^S = c_1 + c_2.$$

Since $\operatorname{supp}(c_1) \subseteq \operatorname{supp}(c) \setminus S$ and $\operatorname{supp}(c_2) \subseteq \operatorname{supp}(c) \setminus R$, property 3 is satisfied. ◀

With Lemma 4 we are ready to show Theorem 1. For the proof it is tempting to apply the lemma iteratively, replacing any complex configuration that is used twice by two configurations with smaller support. This can be repeated until there is no complex configuration taken multiple times. Then we can apply the technique of Lemma 3 to the obtained solution to bound the cardinality of the support. However, the last step might break the structure obtained if the solution implied by Lemma 3 uses a complex configuration more than once. In order to avoid this issue we consider a potential function. We show that a vector minimizing the chosen potential uses each complex configuration at most once, and that the number of complex configurations in the support is bounded. Finally, we apply the techniques from Lemma 3 restricted to variables corresponding to simple configurations.

Proof of Theorem 1. Consider the following potential function of a solution $x \in \mathbb{Z}^Q_{\geq 0}$ of [conf-IP],

$$\Phi(x) = \sum_{\text{complex config. } c} x_c |\operatorname{supp}(c)|.$$

Let x be a solution of [conf-IP] with minimum potential $\Phi(x)$, which is well defined since the set of feasible solutions has finite cardinality. We show two properties of x.

P1: *$x_c \leq 1$ for each complex configuration $c \in Q$.*

Assume otherwise. Consider the two configurations c_1 and c_2 implied by the previous lemma. We define a new solution $x'_e = x_e$ for $e \notin \{c, c_1, c_2\}$, $x'_{c_1} = x_{c_1} + 1$, $x'_{c_2} = x_{c_2} + 1$ and $x'_c = x_c - 2$. Since $|\operatorname{supp}(c_1)| < |\operatorname{supp}(c)|$ and $|\operatorname{supp}(c_2)| < |\operatorname{supp}(c)|$, we obtain that $\Phi(x') < \Phi(x)$ which contradicts the minimality of $\Phi(x)$.

P2: *The number of complex configurations in* $\operatorname{supp}(x)$ *is at most* $2(d+1)\log(4(d+1)T)$.

Let $\tilde{x}$ be the vector defined as $\tilde{x}_c = x_c$ if $c \in Q$ is complex, and $\tilde{x} = 0$ if $c \in Q$ is simple. Then Lemma 2 implies that there are exist two disjoint subsets $A, B \subseteq \operatorname{supp}(\tilde{x})$ of complex configurations such that $Mx^A = Mx^B$. Thus, the solution $x' = x - x^A + x^B$ and the solution $x'' = x - x^B + x^A$ are feasible for [config-IP]. By linearity, the potential function on the new solutions are $\Phi(x') = \Phi(x) - \Phi(x^A) + \Phi(x^B)$ or respectively $\Phi(x'') = \Phi(x) - \Phi(x^B) + \Phi(x^A)$. If $\Phi(x^A) > \Phi(x^B)$ or $\Phi(x^B) > \Phi(x^A)$ then we have constructed a new solution with smaller potential, contradicting our assumption on the minimality of $\Phi(x)$. We conclude that $\Phi(x^B) = \Phi(x^A)$ and thus $\Phi(x) = \Phi(x')$. By construction of x', we obtain that $x'_c > x_c \geq 1$ for any complex configuration $c \in B$. Having multiplicity ≥ 2 for a complex configuration c, we can proceed as in Case 1 to find a new solution with decreased potential, which yields a contradiction.

Given these two properties, to conclude the theorem it suffices to upper bound the number of simple configurations by $2(d+1)\log(4(d+1)T)$. Suppose this property is violated, then we find two sets $A, B \subseteq \operatorname{supp}(x)$ of simple configurations (see Lemma 2) with $Mx^A = Mx^B$ and proceed as in Lemma 3. Since Lemma 3 is only applied to simple configurations, properties P1 and P2 continue to hold and the theorem follows. ◀

Our techniques, in particular our Sparsification Lemma, imply two corollaries on the structure of the knapsack polytope and the LP-relaxation implied by the [conf-IP].

▶ **Corollary 5.** *Every vertex of conv.hull(Q) is a simple configuration. Moreover, the total number of simple configurations in Q is upper bounded by $2^{O(\log^2(T)+\log^2(d))}$ and thus the same expression upper bounds the number of vertices of conv.hull(Q).*

The following corollary follows as each complex configuration can be represented by a convex combination of simple configurations.

▶ **Corollary 6.** *Let [conf-LP] be the LP relaxation of [conf-IP], obtained by changing each constraint $x_c \in \mathbb{Z}_{\geq 0}$ to $x_c \geq 0$ for all $c \in Q$. If the LP is feasible then there exists a solution x such that each configuration $c \in \operatorname{supp}(x)$ is simple.*

4 Applications to Scheduling on Parallel Machines

In what follows we show how to exploit the structural insights of the previous section to derive faster algorithms for parallel machines scheduling problems. We start by considering $P||C_{\max}$, where we seek to assign a set of jobs $\mathcal{J}$ with processing times $p_j \in \mathbb{Z}_{>0}$ to a set $\mathcal{M}$ of m machines. For a given assignment $a : \mathcal{J} \mapsto \mathcal{M}$, we define the load of a machine i as $\sum_{j:a(j)=i} p_j$ and the *makespan* as the maximum load of jobs over all machines, which is the minimum time needed to complete the execution of all jobs on the processors. The goal is to find an assignment $\mathcal{J} \mapsto \mathcal{M}$ that minimizes the makespan.

We first follow well known rounding techniques [1, 2, 9, 8]. Consider an error tolerance $0 < \varepsilon < 1/3$ such that $1/\varepsilon^2$ is an integer. To get an estimation of the optimal makespan, we follow the standard dual approximation approach. First, we can use, e.g., the 2-approximation algorithm by Graham [6] to get an initial guess of the optimal makespan. Using binary search, we can then estimate the optimal makespan within a factor of $(1+\varepsilon)$ in $O(\log(1/\varepsilon))$ iterations. Therefore, it remains to give an algorithm that decides for a given makespan T, if there exists an assignment with makespan $(1+O(\varepsilon))T$ or reports that there exists no assignment with makespan $\leq T$.

For a given makespan T we define the set of big jobs $\mathcal{J}_{\text{big}} = \{j \in \mathcal{J} : p_j \geq \varepsilon T\}$ and the set of small jobs $\mathcal{J}_{\text{small}} = \mathcal{J} \setminus \mathcal{J}_{\text{big}}$. The following lemma shows that small jobs can be replaced from the instance by adding big jobs, each of size εT, as placeholders. Let S be the sum of processing times of jobs in $\mathcal{J}_{\text{small}}$ and let S^* denote the next value of S rounded up to the next multiple of εT, that is, $S^* = \varepsilon T \cdot \lceil S/(\varepsilon T) \rceil$. We define a new instance containing only big jobs by $\mathcal{J}^* = \mathcal{J}_{\text{big}} \cup \mathcal{J}_{\text{new}}$, where $\mathcal{J}_{\text{new}}$ contains $S^*/(\varepsilon T) \in \mathbb{N}$ jobs of size εT. The proof of the next lemma and the rest of the missing proofs of this section can be found in the full version.

▶ **Lemma 7.** *Given a feasible assignment $a : \mathcal{J} \mapsto \mathcal{M}$ of jobs with makespan T. Then there exists a feasible assignment $a_B : \mathcal{J}^* \mapsto \mathcal{M}$ of makespan $T^* \leq (1+\varepsilon)T$. Similarly, an assignment of jobs in $\mathcal{J}^*$ of makespan T^* can be transformed to an assignment of $\mathcal{J}$ of makespan at most $(1+\varepsilon)T^*$.*

By scaling the processing times of jobs in $\mathcal{J}^*$, we can assume that the makespan T has value $1/\varepsilon^2$. Also notice that we can assume that $p_j \leq T$ for all j, otherwise we cannot pack all jobs within makespan T. This implies that each job $j \in \mathcal{J}^*$ has a processing time of $1/\varepsilon \leq p_j \leq 1/\varepsilon^2$. In the following we give a transformation of big jobs in $\mathcal{J}^*$ by rounding their processing times. We first round the jobs to the next power of $1+\varepsilon$ as $p'_j = (1+\varepsilon)^{\lceil \log_{(1+\varepsilon)} p_j \rceil}$, and thus all rounded processing times belong to $\Pi' = \{(1+\varepsilon)^k : 1/\varepsilon \leq (1+\varepsilon)^k \leq (1+\varepsilon)/\varepsilon^2 \text{ and } k \in \mathbb{N}\}$. We further round processing times p'_j to the next integer $\bar{p}_j = \lceil p'_j \rceil$ and define a new set $\Pi = \{\lceil p \rceil : p \in \Pi'\}$. Notice that Π only contains integers and $|\Pi| \leq |\Pi'| \in O((1/\varepsilon)\log(1/\varepsilon))$.

▶ **Lemma 8.** *If there is a feasible schedule of jobs $\mathcal{J}^*$ with processing times p_j onto m machines with makespan $T^* \leq (1+\varepsilon)T$, then there is also a feasible schedule of jobs $\mathcal{J}^*$ with*

rounded processing $\bar{p}_j$ with a makespan of at most $(1+5\varepsilon)T$. Furthermore, the number of different processing times is at most $|\Pi| \in O((1/\varepsilon)\log(1/\varepsilon))$.

In what follows we give an algorithm that decides in polynomial time the existence of a solution for instance $\mathcal{J}^*$ with processing times $\bar{p}_j$ and makespan $\bar{T} = \lfloor (1+5\varepsilon)T \rfloor$. We call numbers in Π by $\pi_1, \ldots, \pi_d$ and define the vector $\pi = (\pi_1, \pi_2, \ldots, \pi_d) \in \mathbb{N}^d$ of rounded processing times. We consider *configurations* to be vectors in $Q = \mathcal{P} \cap \mathbb{Z}^d$, where $\mathcal{P} = \{c \in \mathbb{R}^d_{\geq 0} : \pi \cdot c \leq \bar{T}\}$ is a knapsack polytope (see Section 3). As before, we say that a configuration is simple if $|\operatorname{supp}(c)| \leq \log(\bar{T}+1)$, and complex otherwise. For a given assignment of jobs to machines, we say that a machine follows a configuration c if c_k is the number of jobs of size π_k assigned to the machine. We denote by $Q_c \subseteq Q$ the set of complex configurations and by $Q_s \subseteq Q$ the set of simple configurations.

Let b_k be the number of jobs of size π_k in the instance $\mathcal{J}^*$ (with processing times $\bar{p}$). Consider an ILP with integer variables x_c for each $c \in Q$, which denote the number of machines that follow configuration c. With these parameters the problem of scheduling all jobs in a solution of makespan $\bar{T}$ is equivalent to finding a solution to [conf-IP]. To solve the ILP we use, among other techniques, Kannan's algorithm [12] which is an improvement on the algorithm by Lenstra [13]. The algorithm has a running time of $2^{O(N \log N)} s$ where N is the number of variables and s is number of bits used to encode the input of the ILP in binary.

By Theorem 1, if [conf-IP] is feasible then there exists a thin solution. In particular if one configuration c is used by more than one machine then c is simple, and the total number of used configurations is $4(d+1)\log(4(d+1)\bar{T}) \in O((1/\varepsilon)\log^2(1/\varepsilon))$. Additionally, the number of machines following a complex configurations is at most $2(d+1)\log(4(d+1)\bar{T}) \in O((1/\varepsilon)\log^2(1/\varepsilon))$. We consider the following strategy to decide the existence of a schedule of makespan $\bar{T}$.

▶ Algorithm 9.

1. For each processing time π_k, guess the number $b^c_k \leq b_k$ of jobs covered by complex configurations.
2. Find a minimum number of machines m^c to schedule jobs b^c with makespan $\bar{T}$.
3. Guess the support of simple configurations $\bar{Q}_s \subseteq Q_s$ used by a thin solution, with $|\bar{Q}_s| \leq 4(d+1)\log(4(d+1)\bar{T}) \in O((1/\varepsilon)\log^2(1/\varepsilon))$.
4. Solve the ILP restricted to configurations in $\bar{Q}_s$:

$$\sum_{c \in \bar{Q}_s} c \cdot x_c = b - b^c,$$
$$\sum_{c \in \bar{Q}_s} x_c = m - m^c,$$
$$x_c \in \mathbb{Z}_{\geq 0} \qquad \text{for all } c \in \bar{Q}_s.$$

One of the key observations to prove the running time of the algorithm is that the number of simple configurations $|Q_s|$ is bounded by a quasi polynomial term:

$$|Q_s| \leq 2^{O(\log^2(1/\varepsilon))}.$$

This follows easily by Corollary 5, using that $|\bar{T}| \in O(1/\varepsilon^2)$ and $d = |\Pi| \in O((1/\varepsilon)\log(1/\varepsilon))$.

▶ **Lemma 10.** *Algorithm 9 can be implemented with a running time of $2^{O((1/\varepsilon)\log^4(1/\varepsilon))}\log(n)$.*

Proof. In step 1, the algorithm guesses which jobs are processed on machines following a complex configurations. Since each configuration contains at most $O(1/\varepsilon)$ jobs, there are at most $O(m^c/\varepsilon) = O((1/\varepsilon^2)\log^2(1/\varepsilon))$ jobs assigned to such machines. For each size $\pi_k \in \Pi$, we guess the number b_k^c of jobs of size π_k assigned to such machines. Hence, we can enumerate all possibilities for jobs assigned to complex machines in time $2^{O((1/\varepsilon)\log^2(1/\varepsilon))}$. After guessing the jobs, we can assign them to a minimum number of machines in step 2 (with makespan $\bar{T}$) with a simple dynamic program that stores vectors $(\ell, z_1, \ldots, z_d)$ with $z_k \leq b_k^c$ being the number of jobs of size π_k used in the first $\ell \leq m^c$ processors [11]. The size of the dynamic programming table is $O(m^c \prod_{k=1}^{d}(b_k^c+1))$. For any vector $(\ell, z_1, \ldots, z_d)$, determining whether it corresponds to a feasible solution can be done by checking all vectors of the type $(\ell-1, z_1', \ldots, z_d')$ for $z_k' \leq z_k$. Thus, the running time of the dynamic program is $O(m^c[\prod_{k=1}^{d}(b_k^c+1)]^2)$. Since $b_k^c \in O((1/\varepsilon^2)\log^2(1/\varepsilon))$ for each k, recalling that $m^c \in O((1/\varepsilon)\log^2(1/\varepsilon))$, and that $d = |\Pi| \in O((1/\varepsilon)\log(1/\varepsilon))$, we obtain that step 2 can be implemented with $2^{O((1/\varepsilon)\log^2(1/\varepsilon))}$ running time.

In step 3, our algorithm guesses the support of a thin solution x. Recall that if x is thin then $|\operatorname{supp}(x)| \leq 4(d+1)\log(4(d+1)\bar{T}) = O((1/\varepsilon)\log^2(1/\varepsilon))$. Let $D = 4(d+1)\log(4(d+1)\bar{T})$. Then this guess can be done in time

$$\sum_{i=0}^{D} \binom{|Q_s|}{i} \leq (D+1)|Q_s|^D \leq 2^{O((1/\varepsilon)\log^4(1/\varepsilon))}.$$

We remark that for this step is that thin solutions are particularly useful. Indeed, guessing the support on the original ILP takes time $2^{O((1/\varepsilon)^2\log^3(1/\varepsilon))}$.

In step 4, the restricted ILP with $4(d+1)\log(4(d+1)\bar{T}) = O((1/\varepsilon)\log^2(1/\varepsilon))$ variables is solved. Moreover, the size of the input can be bounded by $O((1/\varepsilon^2)\log^3(1/\varepsilon)\log(n))$. Running Kannan's algorithm [12] to solve the ILP takes time $2^{O((1/\varepsilon)\log^3(1/\varepsilon))}\log(n)$. Hence, the total running time of our algorithm can be bounded by $2^{O((1/\varepsilon)\log^4(1/\varepsilon))}\log(n)$. ◀

Putting all pieces together, we conclude with the following theorem.

▶ **Theorem 11.** *The minimum makespan problem on parallel machines $P||C_{\max}$ admits an EPTAS with running time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + O(n\log n)$.*

4.1 Extension to other objectives

We now consider a more general family of objective functions defined by Alon et al. [1, 2]. For a fixed function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, we consider the following two objective functions: (I) $\min \sum_{i\in\mathcal{M}} f(\ell_i)$, and (II) $\min \max_{i\in\mathcal{M}} f(\ell_i)$, where ℓ_i denotes the load of machine i. Analogously, we study maximization versions of the problems: (I') $\max \sum_{i\in\mathcal{M}} f(\ell_i)$ and (II') $\max \min_{i\in\mathcal{M}} f(\ell_i)$.

For the minimization versions of the problem we assume that f is convex, while for (I') and (II') we assume it is concave. Moreover, we will need that the function satisfies the following sensitivity condition.

▶ **Condition 12.** *For all $\varepsilon > 0$ there exists $\delta = \delta(\varepsilon) > 0$ such that for all $x, y \in \mathbb{R}_{\geq 0}$,*

$$(1-\delta)y \leq x \leq (1+\delta)y \quad \Rightarrow \quad (1-\varepsilon)f(y) \leq f(x) \leq (1+\varepsilon)f(y).$$

Alon et al. showed that each problem in that family admits a PTAS with running time $h(\varepsilon)+O(n\log n)$, where $h(\varepsilon)$ is a constant term that depends only on ε. Moreover, if $\delta(\varepsilon)$ in the condition further satisfies that $1/(\delta(\varepsilon)) \in O(1/\varepsilon)$, the running time is $2^{(1/\varepsilon)^{\mathrm{poly}(1/\varepsilon)}} + O(n\log n)$.

In what follows we show how to improve this dependency if we have the additional condition that function f is non-decreasing, i.e., for all $0 \leq x \leq y$ we have that $f(x) \leq f(y)$. Since $1/(\delta(\varepsilon)) \in O(1/\varepsilon)$, we know that, for small enough ε, there exists a constant γ (independent of ε and δ) such that $1/\delta \leq \gamma/\varepsilon$. Moreover, we can assume w.l.o.g. that $\delta \leq \varepsilon$, and thus $\delta \leq \varepsilon \leq \gamma\delta$.

It is worth noticing that many interesting functions belong to this family. In particular (II) with $f(x) = x$ corresponds to the minimum makespan problem, (I) with $f(x) = x^p$, for constant p, corresponds to a problem that is equivalent to minimizing the L_p-norm of the vector of loads. Similarly, (II') with $f(x) = x$ corresponds to maximizing the minimum machine load. Notice that for all those objectives we have that $1/\delta = O(1/\varepsilon)$.

The techniques of Alon et al. are based on a rounding method and then solving an ILP. We based our results in the same rounding techniques and extend them further. We show that to obtain a PTAS in time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + O(n \log n)$ it suffices to obtain a $(1 + O(\varepsilon))$-approximate solution to the rounded instance in the same running time. The details of the rounding are given in the full version.

Let $L = \sum_j p_j/m$ be the average machine load (of the original instance). After our rounding we obtain an instance I' with job set $\mathcal{J}'$ and processing times $\bar{p}_j$ for $j \in \mathcal{J}'$. Moreover, the $\bar{p}_j$ are multiples of L/λ^2, where $\lambda \geq 1/\delta$ is an integer such that $\lambda = O(1/\delta)$, and also $\bar{p}_j \geq L/\lambda$. It holds that there exists an optimal solution of the rounded instance with makespan at most $4L$ (see full version). Let $\Pi = \{\pi_1, \ldots, \pi_d\}$ be the distinct values that the processing times $\bar{p}_j$ can take. Our rounding guarantees that $d = |\Pi| = O((1/\delta)\log(1/\delta))$. We consider the knapsack polytope with capacity $\bar{T} := 4L$, that is $\mathcal{P} = \{c \in \mathbb{R}^d_{\geq 0} : \pi \cdot c \leq \bar{T}\}$. Notice that π and $\bar{T}$ are integer multiples of L/λ^2, and that $\mathcal{P}$ can also be written as $\{c \in \mathbb{R}^d_{\geq 0} : \pi/(L/\lambda^2) \cdot c \leq \bar{T}/(L/\lambda^2)\}$.

As before, we say that a configuration is simple if $|\operatorname{supp}(c)| \leq \log(\bar{T} + 1)$, and complex otherwise. We denote by $Q_c \subseteq Q$ the set of complex configurations and by $Q_s \subseteq Q$ the set of simple configurations. In what follows we focus on objective function (I).

We set an ILP for the problem as before. Notice that each configuration c incurs a cost of $f_c := f(\pi \cdot c)$. Moreover, we round and scale the values f_c by defining $\bar{f}_c = \lceil f_c/(\varepsilon f_{\min})\rceil$, where $f_{\min} = \min_{c \in Q} f_c$. It is not hard to see that solving a problem with those coefficients yields a $(1 + \varepsilon)$-approximate solution to the optimal solution of I' with processing times $\bar{p}_j$. Let also b_k be the number of jobs j of processing time $\bar{p}_j = \pi_k$ in $\mathcal{J}'$. Consider the ILP obtained by adding to [conf-IP] the objective function $\min \sum_{c \in Q} \bar{f}_c \cdot x_c$. We call this ILP [cost-conf-IP]. With our previous discussion, it suffices to solve this ILP optimally. To solve this problem, we first notice that the largest coefficient in the objective can be bounded as follows.

▶ **Lemma 13.** *If f satisfies Condition 12 and it is non-decreasing, then the largest value $\max_{c \in Q} \bar{f}_c$ is upper bounded by $1/\delta^{O(1)}$.*

As we now must consider the objective function, we cannot simply apply Theorem 1 to [cost-conf-ILP]. However, we can prove a slightly weaker version by decomposing the ILP in several smaller ones and applying the theorem to each of them.

▶ **Theorem 14.** *If* [cost-conf-IP] *is feasible, then there exists an optimal solution x satisfying:*

1. $\sum_{c \in Q_c} x_c \in O((1/\delta^3)\log^2(1/\delta))$, *and*
2. $|\operatorname{supp}(x) \cap Q_s| \in O((1/\delta)\log^2(1/\delta))$.

Proof. Notice that the load of each configuration $\pi \cdot c$ is a multiple of L/λ^2, and thus $\pi \cdot c \in \{L/\lambda, L/\lambda + L/(\lambda^2), \ldots, 4L\}$. We classify the configurations according to their loads,

$Q^\ell := \{c \in Q : \pi \cdot c = L/\lambda + \ell \cdot L/(\lambda^2)\}$, for $\ell \in \{0, \dots, 4\lambda^2 - \lambda\}$. Let x^* be an optimal solution of [cost-conf-IP]. Then we can considered an ILP for each load value ℓ:

$$[\text{conf-IP}]_\ell \quad \sum_{c \in Q^\ell} c \cdot x_c = \sum_{c \in Q^\ell} c \cdot x^*_c, \tag{4}$$

$$\sum_{c \in Q^\ell} x_c = \sum_{c \in Q^\ell} x^*_c, \tag{5}$$

$$x_c \in \mathbb{Z}_{\geq 0} \qquad \text{for all } c \in Q^\ell. \tag{6}$$

Scaling π by multiplying it by λ^2/L we obtain an integral vector (since π is an integer multiple of $L/(\lambda^2)$), we can apply Theorem 1 to each ILP $[\text{conf-IP}]_\ell$, which yields that there exists a thin solution x^ℓ. In particular the number of complex configurations in x^ℓ is $\sum_{c \in Q_c \cap Q^\ell} x^\ell_c \in O((1/\delta)\log^2(1/\delta))$. Since $\bar{f}_c$ depends only on the load of c, concatenating these solutions yields a solution $x' := (x^\ell)_\ell$ that is optimal for [cost-conf-IP], such that $\sum_{c \in Q_c} x'_c \in O((\lambda^2) \cdot (1/\delta)\log^2(1/\delta)) = O((1/\delta^3)\log^2(1/\delta))$. It remains to bound the number of simple configurations in the support. To this end, we consider the ILP restricted to simple configurations as follows:

$$[\text{cost-conf-IP}]_s \quad \min \sum_{c \in Q_s} \bar{f}_c \cdot x_c$$

$$\sum_{c \in Q_s} c \cdot x_c = b - \sum_{c \in Q_c} c \cdot x'_c, \tag{7}$$

$$\sum_{c \in Q_s} x_c = m - \sum_{c \in Q_c} x'_c, \tag{8}$$

$$x_c \in \mathbb{Z}_{\geq 0} \qquad \text{for all } c \in Q_s. \tag{9}$$

We apply the result of Eisenbrand and Shmonin [4] to this ILP. In its more general form, this result ensures the existence of a solution x'' with support of size $O(N(\log(N)+\Delta))$, where N is the number of restrictions and Δ is the encoding size of the largest coefficient appearing in the cost vector and restriction matrix. In our case $N = d + 1 = O((1/\delta)\log(1/\delta))$, and $\Delta = O(\log(\max\{1/\delta, \max_{c \in Q} \bar{f}_c)\}) = O(\log(1/\delta))$ (Lemma 13). Thus $O(N(\log(N) + \Delta)) = O((1/\delta)\log^2(1/\delta))$. The theorem follows by concatenating $(x''_c)_{c \in Q_s}$ with $(x'_c)_{c \in Q_c}$. ◀

Finally, we use the structure given by the theorem to solve this ILP optimally. The idea is similar to Algorithm 9 and thus we defer the details to the full version.

▶ **Theorem 15.** *Consider the scheduling problem on parallel machines with objective functions* (I), (II) *for f convex (respectively* (I') *and* (II') *for f concave). If f satisfies Condition 12 for $1/\delta = O(1/\varepsilon)$ and it is non-decreasing, then the problem admits an EPTAS with running time $2^{O((1/\varepsilon)\log^4(1/\varepsilon))} + O(n \log n)$.*

5 Minimum makespan scheduling on uniform machines

In this section we generalize our result for $P||C_{\max}$ to uniform machines. Consider a set of jobs $\mathcal{J}$ with processing times p_j and a set of m non-identical machines $\mathcal{M}$ where machine $i \in \mathcal{M}$ runs at speed s_i. If job j is executed on machine i the machine needs p_j/s_i time units to complete the job. The problem is to find an assignment $a : \mathcal{J} \to \mathcal{M}$ for the jobs to the machines that minimizes the makespan; $\max_i \sum_{j:a(j)=i} p_j/s_i$. The problem is denoted by $Q||C_{max}$. We suppose that $s_1 \geq s_2 \geq \ldots \geq s_m$. Jansen [10] found an efficient polynomial

time approximation scheme (EPTAS) for this scheduling problem which has a running time of $2^{O(1/\varepsilon^2 \log^3(1/\varepsilon))} + \text{poly}(n)$. Here we show how to improve the running time and prove the main result of this section.

▶ **Theorem 16.** *There is an EPTAS (a family of algorithms $\{A_\varepsilon : \varepsilon > 0\}$) which, given an instance I of $Q||C_{max}$ with n jobs and m machines and a positive number $\varepsilon > 0$, produces a schedule of makespan $A_\varepsilon(I) \leq (1+\varepsilon)\textsc{Opt}(I)$. The running time of A_ε is $2^{O(1/\varepsilon \log^4(1/\varepsilon))} + \text{poly}(n)$.*

Let $0 < \delta < \varepsilon$. We follow the approach by Jansen [10], transforming the scheduling problem into a bin packing problem with different bin capacities, rounding the processing times and bin capacities, and dividing the bins into at most three groups $\mathcal{B}_1, \mathcal{B}_2$ and $\mathcal{B}_3$ depending on the bin capacities.

Here, we focus on a special case which contains the main difficulty of the problem. The full exposition can be found in the full version. In group $\mathcal{B}_2$, the bins can have a capacity of value $\bar{c}(1) > \ldots > \bar{c}(L)$ for some $L \in O(1/\delta \log(1/\delta))$. We call B_ℓ the set of bins in $\mathcal{B}_2$ with capacity $\bar{c}(\ell)$. Our rounded instance contains jobs of sizes $\pi_1, \ldots, \pi_d$ for $d \in O(1/\delta \log(1/\delta))$. For each B_ℓ we consider configurations $\bar{C}_1^{(\ell)}, \ldots, \bar{C}_{\bar{h}_\ell}^{(\ell)}$. The configurations are defined using job of size at least $\delta\bar{c}(\ell)$ which are rounded up to multiples of $\delta^2\bar{c}(\ell)$. Thus, regarding these configurations (and only these configurations), jobs have sizes of the form $q(k,\ell)\delta^2\bar{c}(\ell)$ with $q(k,\ell) \in \mathbb{Z}^+$ and $k \in \{1,\ldots,d\}$. We denote by $a(k, \bar{C}_i^{(\ell)})$ the multiplicity of jobs of size $q(k,\ell)\delta^2\bar{c}(\ell)$ in configuration $\bar{C}_i^{(\ell)}$. The rounding implies also that the rounded size $size(\bar{C}_i^{(\ell)})$ of a configuration is a multiple of $\delta^2\bar{c}(\ell)$. Each such configuration corresponds to an integral point inside the knapsack polytope $\mathcal{P}_\ell = \{C = (a(k,C))_k : \sum_k q(k,\ell)\delta^2\bar{c}(\ell)a(k,C) \leq (1+\delta)\bar{c}(\ell)\}$.

Let $\bar{m}_\ell = |B_\ell|$ be the number of machines of capacity $\bar{c}(\ell)$. Consider a given solution to our scheduling problem. For this case we say that a bin in B_ℓ follows a configuration $\bar{C}_i^{(\ell)}$ if it has $a(k, \bar{C}_i^{(\ell)})$ jobs whose size, rounded to the next multiple of $\delta^2\bar{c}(\ell)$, equals to $q(k,\ell)\delta^2\bar{c}(\ell)$. Let $\bar{x}_i^{(\ell)}$ be the number of bins in B_ℓ that follows configuration $\bar{C}_i^{(\ell)}$. Notice that the configurations do not consider jobs of size smaller than $\delta\bar{c}(\ell)$ that might be assigned to a bin in B_ℓ. Consider the following ILP, which we denote by $[\text{conf-IP}]_Q$:

$$\sum_i x_i^{(\ell)} = \bar{m}_\ell \qquad \text{for } \ell = 1,\ldots,L,$$

$$\sum_{\ell,i} a(k, \bar{C}_i^{(\ell)}) x_i^{(\ell)} = \sum_{\ell,i} a(k, \bar{C}_i^{(\ell)}) \bar{x}_i^{(\ell)} \qquad \text{for } k = 1\ldots,d,$$

$$\sum_i \frac{size(\bar{C}_i^{(\ell)})}{\delta^2\bar{c}(\ell)} x_i^{(\ell)} = \sum_i \frac{size(\bar{C}_i^{(\ell)})}{\delta^2\bar{c}(\ell)} \bar{x}_i^{(\ell)},$$

$$x_i^{(\ell)} \geq 0 \,\text{integral} \qquad \text{for } i = 1,\ldots,\bar{h}_\ell, \ell = 1,\ldots,L.$$

The second equality of the ILP ensures that the solution constructed maintains the same subset of jobs larger than $\delta\bar{c}(\ell)$ to bins in B_ℓ as solution $\bar{x}$. The third equality ensures that we are leaving enough space for jobs not covered by a configuration (i.e., a job that is assigned within B_ℓ but whose size is less than $\delta\bar{c}(\ell)$). As in previous sections, a configuration $\bar{C}_i^{(\ell)}$ is called simple if $|\operatorname{supp}(\bar{C}_i^{(\ell)})| \leq \log(\bar{c}(\ell)(1+\delta)/(\delta^2\bar{c}(\ell)) + 1) = \log(1/\delta^2 + 1/\delta + 1)$ (here we are scaling the capacity of a configuration by $\delta^2\bar{c}(\ell)$, since all rounded job sizes are multiples of $\delta^2\bar{c}(\ell)$). Otherwise, we call a configuration $\bar{C}_i^{(\ell)}$ complex. Crucially, the ILP above satisfies that all coefficients are at most $\text{poly}(1/\delta)$. This allows us to generalize our result in Theorem 1 to our ILP above, with a similar proof technique as Theorem 14, which yields the following lemma. Using this lemma, we can define an algorithm similar to

the algorithm for the case of identical machines, to obtain an improved running time for $Q||C_{max}$ and prove Theorem 16.

▶ **Lemma 17.** *Assume that the ILP [conf-IP]$_Q$ is feasible and let S denote the set of all simple configurations. Then there exists a feasible solution x' such that:* (1) *If $x'^{(\ell)}_i > 1$ then the configuration $\bar{C}^{(\ell)}_i$ is simple,* (2) *the support of x' satisfies $|\operatorname{supp}(x') \cap S| \in O(1/\delta \log^2(1/\delta))$, and* (3) *the support of x' satisfies $|\operatorname{supp}(x') \setminus S| \in O(1/\delta^2 \log^3(1/\delta))$.*

References

1. N. Alon, Y. Azar, G. Woeginger, and T. Yadid. Approximation schemes for scheduling. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '97)*, pages 493–500. ACM/SIAM, 1997.
2. N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1:55–66, 1998.
3. L. Chen, K. Jansen, and G. Zhang. On the optimality of approximation schemes for the classical scheduling problem. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*, pages 657–668. ACM/SIAM, 2013.
4. F. Eisenbrand and G. Shmonin. Carathéodory bounds for integer cones. *Operations Research Letters*, 34:564–568, 2006.
5. M. X. Goemans and T. Rothvoß. Polynomiality for bin packing with a constant number of item types. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 830–839. ACM/SIAM, 2014.
6. R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
7. R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17:416–429, 1969.
8. D. Hochbaum, editor. *Approximation algorithms for NP-hard problems.* PWS Publishing Company, 1997.
9. D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.
10. K. Jansen. An EPTAS for scheduling jobs on uniform processors: Using an MILP relaxation with a constant number of integral variables. *SIAM Journal on Discrete Mathematics*, 24:457–485, 2010.
11. K. Jansen and C. Robenek. Scheduling jobs on identical and uniform processors revisited. In *Approximation and Online Algorithms (WAOA'11)*, number 7164 in Lecture Notes in Computer Science, pages 109–122. Springer, 2011.
12. R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12:415–440, 1987.
13. H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.
14. J. Y-T. Leung. Bin packing with restricted piece sizes. *Information Processing Letters*, 31:145–149, 1989.

Constant Approximation for Capacitated k-Median with $(1+\epsilon)$-Capacity Violation*

Gökalp Demirci[1] and Shi Li†[2]

1 **Department of Computer Science, University of Chicago, Chicago, IL, USA**
demirci@cs.uchicago.edu

2 **Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA**
shil@buffalo.edu

Abstract

We study the Capacitated k-Median problem for which existing constant-factor approximation algorithms are all pseudo-approximations that violate either the capacities or the upper bound k on the number of open facilities. Using the natural LP relaxation for the problem, one can only hope to get the violation factor down to 2. Li [SODA'16] introduced a novel LP to go beyond the limit of 2 and gave a constant-factor approximation algorithm that opens $(1+\epsilon)k$ facilities.

We use the configuration LP of Li [SODA'16] to give a constant-factor approximation for the Capacitated k-Median problem in a seemingly harder configuration: we violate only the capacities by $1+\epsilon$. This result settles the problem as far as pseudo-approximation algorithms are concerned.

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

Keywords and phrases Approximation Algorithms, Capacitated k-Median, Pseudo Approximation, Capacity Violation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.73

1 Introduction

In the capacitated k-median problem (CKM), we are given a set F of facilities together with their capacities $u_i \in \mathbb{Z}_{>0}$ for $i \in F$, a set C of clients, a metric d on $F \cup C$, and a number k. We are asked to open some of these facilities $F' \subseteq F$ and give an assignment $\sigma : C \to F'$ connecting each client to one of the open facilities so that the number of open facilities is not bigger than k, i.e. $|F'| \leq k$ (*cardinality constraint*), and each facility $i \in F'$ is connected to at most u_i clients, i.e. $\left|\sigma^{-1}(i)\right| \leq u_i$ (*capacity constraint*). The goal is to minimize the sum of the connection costs, i.e. $\sum_{j \in C} d(\sigma(j), j)$.

Without the capacity constraint, i.e. $u_i = \infty$ for all $i \in F$, this is the famous k-median problem (KM). The first constant-factor approximation algorithm for KM is given by Charikar et al. [9], guaranteeing a solution within $6\frac{2}{3}$ times the cost of the optimal solution. Then the approximation ratio has been improved by a series of papers [13, 8, 3, 12, 17, 5]. The current best ratio for KM is $2.675 + \epsilon$ due to Byrka et al. [5], which was obtained by improving a part of the algorithm given by Li and Svensson [17].

On the other hand, we don't have a true constant approximation for CKM. All known constant-factor results are pseudo-approximations which violate either the cardinality or the

* A full version of the paper can be found at http://arxiv.org/abs/1603.02324.
† Supported in part by NSF grant CCF-1566356.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 73; pp. 73:1–73:14

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

capacity constraint. Aardal et al. [1] gave an algorithm which finds a $(7+\epsilon)$-approximate solution to CKM by opening at most $2k$ facilities, i.e. violating the cardinality constraint by a factor of 2. Guha [11] gave an algorithm with approximation ratio 16 for the more relaxed *uniform* CKM, where all capacities are the same, by connecting at most $4u$ clients to each facility, thus violating the capacity constraint by 4. Li [14] gave a constant-factor algorithm for uniform CKM with capacity violation of only $2+\epsilon$ by improving the algorithm in [9]. For non-uniform capacities, Chuzhoy and Rabani [10] gave a 40-approximation for CKM by violating the capacities by a factor of 50 using a mixture of primal-dual schema and lagrangian relaxations. Their algorithm is for a slightly relaxed version of the problem called *soft* CKM where one is allowed to open multiple collocated copies of a facility in F. The CKM definition we gave above is sometimes referred to as *hard* CKM as opposed to this version. Recently, Byrka et al. [4] gave a constant-factor algorithm for hard CKM by keeping capacity violation factor under $3+\epsilon$.

All these algorithms for CKM use the basic LP relaxation for the problem which is known to have an unbounded integrality gap even when we are allowed to violate either the capacity or the cardinality constraint by $2-\epsilon$. In this sense, results of [1] and [14] can be considered as reaching the limits of the basic LP relaxation in terms of restricting the violation factor. In order to go beyond these limits, Li [15] introduced a novel LP called the *rectangle* LP and presented a constant-factor approximation algorithm for soft uniform CKM by opening $(1+\epsilon)k$ facilities. This was later generalized by the same author to non-uniform CKM [16], where he introduced an even stronger LP relaxation called the *configuration* LP. Very recently, independently of the work in this paper, Byrka et al. [6] used this configuration LP to give a similar algorithm for uniform CKM violating the capacities by $1+\epsilon$.

1.1 Our Result

In this paper, we use the configuration LP of [16] to give an $O(1/\epsilon^5)$-approximation algorithm for non-uniform hard CKM which respects the cardinality constraint and connects at most $(1+\epsilon)u_i$ clients to any open facility $i \in F$. The running time of our algorithm is $n^{O(1/\epsilon)}$. Thus, with this result, we now have settled the CKM problem from the view of pseudo-approximation algorithms: either $(1+\epsilon)$-cardinality violation or $(1+\epsilon)$-capacity violation is sufficient for a constant approximation for CKM.

The known results for the CKM problem have suggested that designing algorithms with capacity violation (satisfying the cardinality constraint) is harder than designing algorithms with cardinality violation. Note, for example, that the best known cardinality violation factor for non-uniform CKM among algorithms using only the basic LP relaxation (a factor of 2 in [1]) matches the smallest possible cardinality violation factor dictated by the gap instance. In contrast, the best capacity-violation factor is $3+\epsilon$ due to [4], but the gap instance for the basic LP with the largest known gap eliminates only the algorithms with capacity violation smaller than 2. Furthermore, we can argue that, for algorithms based on the basic LP and the configuration LP, a β-capacity violation can be converted to a β-cardinality violation, suggesting that allowing capacity violation is more restrictive than allowing cardinality violation. We leave the detail to the full version of the paper.

Our Techniques. Our algorithm uses the configuration LP introduced in [16] and the framework of [16] that creates a two-level clustering of facilities. [16] considered the $(1+\epsilon)$-cardinality violation setting, which is more flexible in the sense that one has the much freedom to distribute the ϵk extra facilities. In our $(1+\epsilon)$-capacity violation setting, each facility i can provide an extra ϵu_i capacity; however, these extra capacities are restricted by the

locations of the facilities. In particular, we need one more level of clustering to form so-called "groups" so that each group contains $\Omega(1/\epsilon)$ fractional open facility. Only with groups of $\Omega(1/\epsilon)$ facilities, we can benefit from the extra capacities given by the $(1+\epsilon)$-capacity scaling. Our algorithm then constructs distributions of local solutions. Using a dependent rounding procedure we can select a local solution from each distribution such that the solution formed by the concatenation of local solutions has a small cost. This initial solution may contain more than k facilities. We then remove some already-open facilities, and bound the cost incurred due to the removal of open facilities. When we remove a facility, we are guaranteed that there is a close group containing $\Omega(1/\epsilon)$ open facilities and the extra capacities provided by these facilities can compensate for the capacity of the removed facility.

Organization. The remaining part of the paper is organized as follows. In Sections 2 and 3, we describe the configuration LP introduced in [16] and our three-level clustering procedure respectively. In Section 4, we show how to construct the distributions of local solutions. Then finally in Section 5, we show how to obtain our final solution by combining the distributions we constructed. Due to the page limit, some proofs are omitted and they can be found in the full version of the paper.

2 The Basic LP and the Configuration LP

In this section, we give the configuration LP of [16] for CKM. We start with the following basic LP relaxation:

$$\min \quad \sum_{i\in F, j\in C} d(i,j)x_{i,j} \quad \text{s.t.} \qquad \text{(Basic LP)}$$

$$\sum_{i\in F} y_i \le k; \qquad (1)$$

$$\sum_{i\in F} x_{i,j} = 1, \qquad \forall j \in C; \qquad (2)$$

$$x_{i,j} \le y_i, \qquad \forall i \in F, j \in C; \qquad (3)$$

$$\sum_{j\in C} x_{i,j} \le u_i y_i, \qquad \forall i \in F; \qquad (4)$$

$$0 \le x_{i,j}, y_i \le 1, \qquad \forall i \in F, j \in C. \qquad (5)$$

In the LP, y_i indicates whether a facility $i \in F$ is open, and $x_{i,j}$ indicates whether client $j \in C$ is connected to facility $i \in F$. Constraint (1) is the cardinality constraint assuring that the number of open facilities is no more than k. Constraint (2) says that every client must be fully connected to facilities. Constraint (3) requires a facility to be open in order to connect clients. Constraint (4) is the capacity constraint.

It is well known that the basic LP has unbounded integrality gap, even if we are allowed to violate the cardinality constraint or the capacity constraint by a factor of $2-\epsilon$. The description of the instance can be found in the full version of the paper. In order to overcome the gap in the cardinality-violation setting, Li [16] introduced a novel LP for CKM called the configuration LP, which we formally state below. Let us fix a set $B \subseteq F$ of facilities. Let $\ell = \Theta(1/\epsilon)$ and $\ell_1 = \Theta(\ell)$ be sufficiently large integers. Let $\mathcal{S} = \{S \subseteq B : |S| \le \ell_1\}$ and $\widetilde{\mathcal{S}} = \mathcal{S} \cup \{\perp\}$, where $\perp$ stands for "any subset of B with size more than ℓ_1"; for convenience, we also treat $\perp$ as a set such that $i \in \perp$ holds for every $i \in B$. For $S \in \mathcal{S}$, let z_S^B indicate the event that the set of open facilities in B is exactly S and $z_\perp^B$ indicate the event that the number of open facilities in B is more than ℓ_1.

For every $S \in \widetilde{\mathcal{S}}$ and $i \in S$, $z_{S,i}^B$ indicates the event that $z_S^B = 1$ and i is open. (If $i \in B$ but $i \notin S$, then the event will not happen.) Notice that when $i \in S \ne \perp$, we always have

$z^B_{S,i} = z^B_S$; we keep both variables for notational purposes. For every $S \in \widetilde{\mathcal{S}}, i \in S$ and client $j \in C$, $z^B_{S,i,j}$ indicates the event that $z^B_{S,i} = 1$ and j is connected to i. In an integral solution, all the above variables are $\{0,1\}$ variables. The following constraints are valid. To help understand the constraints, it is good to think of $z^B_{S,i}$ as $z^B_S \cdot y_i$ and $z^B_{S,i,j}$ as $z^B_S \cdot x_{i,j}$.

$$\sum_{S \in \widetilde{\mathcal{S}}} z^B_S = 1; \tag{6}$$

$$\sum_{S \in \widetilde{\mathcal{S}}: i \in S} z^B_{S,i} = y_i, \qquad \forall i \in B; \tag{7}$$

$$\sum_{S \in \widetilde{\mathcal{S}}: i \in S} z^B_{S,i,j} = x_{i,j}, \qquad \forall i \in B, j \in C; \tag{8}$$

$$0 \le z^B_{S,i,j} \le z^B_{S,i} \le z^B_S, \qquad \forall S \in \widetilde{\mathcal{S}}, i \in S, j \in C; \tag{9}$$

$$z^B_{S,i} = z^B_S, \qquad \forall S \in \mathcal{S}, i \in S; \tag{10}$$

$$\sum_{i \in S} z^B_{S,i,j} \le z^B_S, \qquad \forall S \in \widetilde{\mathcal{S}}, j \in C; \tag{11}$$

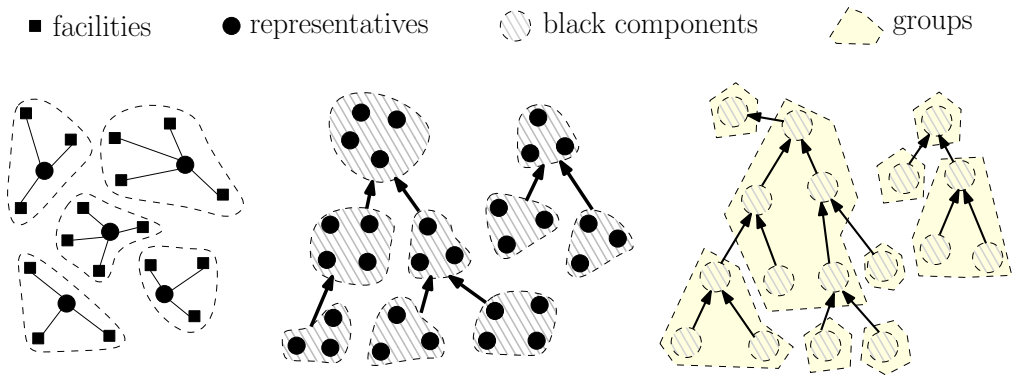
$$\sum_{j \in C} z^B_{S,i,j} \le u_i z^B_{S,i}, \qquad \forall S \in \widetilde{\mathcal{S}}, i \in S; \tag{12}$$

$$\sum_{i \in B} z^B_{\perp,i} \ge \ell_1 z^B_{\perp}. \tag{13}$$

Constraint (6) says that $z^B_S = 1$ for exactly one $S \in \widetilde{\mathcal{S}}$. Constraint (7) says that if i is open then there is exactly one $S \in \widetilde{\mathcal{S}}$ with $z^B_{S,i} = 1$. Constraint (8) says that if j is connected to i then there is exactly one $S \in \widetilde{\mathcal{S}}$ such that $z^B_{S,i,j} = 1$. Constraint (9) is by the definition of variables. Constraint (10) holds as we mentioned earlier. Constraint (11) says that if $z^B_S = 1$ then j can be connected to at most 1 facility in S. Constraint (12) is the capacity constraint. Constraint (13) says that if $z^B_\perp = 1$, there are at least ℓ_1 open facilities in B.

The configuration LP is obtained from the basic LP by adding the z variables and Constraints (6) to (13) for every $B \subseteq F$. Since there are exponentially many subsets $B \subseteq F$, we don't know how to solve this LP efficiently. However, note that there are only polynomially many ($n^{O(\ell_1)}$) z^B variables for a fixed $B \subseteq F$. Given a fractional solution (x, y) to the basic LP relaxation, we can construct the values of z^B variables and check their feasibility for Constraints (6) to (13) in polynomial time as in [16]. Our rounding algorithm either constructs an integral solution with the desired properties, or outputs a set $B \subseteq F$ such that Constraints (6) to (13) are infeasible. In the latter case, we can find a constraint in the configuration LP that (x, y) does not satisfy. Then we can run the ellipsoid method and the rounding algorithm in an iterative way (see, e.g., [7, 2]).

Notations. From now on, we fix a solution $(\{x_{i,j} : i \in F, j \in C\}, \{y_i : i \in F\})$ to the basic LP. We define $d_{\mathsf{av}}(j) := \sum_{i \in F} x_{i,j} d(i,j)$ to be the connection cost of j, for every $j \in C$. Let $D_i := \sum_{j \in C} x_{i,j} (d(i,j) + d_{\mathsf{av}}(j))$ for every $i \in F$, and $D_S := \sum_{i \in S} D_i$ for every $S \subseteq F$. We denote the value of the solution (x, y) by $\mathsf{LP} := \sum_{i \in F, j \in C} x_{i,j} d(i,j) = \sum_{j \in C} d_{\mathsf{av}}(j)$. Note that $D_F = \sum_{i \in F, j \in C} x_{i,j} (d(i,j) + d_{\mathsf{av}}(j)) = \sum_{i \in F, j \in C} x_{i,j} d(i,j) + \sum_{j \in C} d_{\mathsf{av}}(j) \sum_{i \in F} x_{i,j} = 2\mathsf{LP}$. For any set $F' \subseteq F$ of facilities and $C' \subseteq C$ of clients, we shall let $x_{F',C'} := \sum_{i \in F', j \in C'} x_{i,j}$; we simply use $x_{i,C'}$ for $x_{\{i\},C'}$ and $x_{F',j}$ for $x_{F',\{j\}}$. For any $F' \subseteq F$, let $y_{F'} := \sum_{i \in F'} y_i$. Let $d(A,B) := \min_{i \in A, j \in B} d(i,j)$ denote the minimum distance between A and B, for any $A, B \subseteq F \cup C$; we simply use $d(i,B)$ for $d(\{i\}, B)$.



(a). bundles $\{U_v\}_{v\in R}$ (b). black components $\mathcal{J}$ and forest $\Upsilon^*_{\mathcal{J}}$ (c). groups $\mathcal{G}$ and forest $\Upsilon_{\mathcal{G}}$

Figure 1 The three-phase clustering procedure. In the first phase (Figure (a)), we partition F into bundles, centered at the set R of representatives. In the second phase (Figure (b)), we partition R into a family $\mathcal{J}$ of black components and construct a degree-2 rooted forest over $\mathcal{J}$. In the third phase (Figure(c)), we partition $\mathcal{J}$ into a family $\mathcal{G}$ of groups; $\Upsilon_{\mathcal{G}}$ is formed from $\Upsilon^*_{\mathcal{J}}$ by contracting each group into a single node.

Moving of Demands. After the set of open facilities is decided, the optimum connection assignment from clients to facilities can be computed by solving the minimum cost b-matching problem. Due to the integrality of the matching polytope, we may allow the connections to be fractional. That is, if there is a good fractional assignment, then there is a good integral assignment. So we can use the following framework to design and analyze the rounding algorithm. Initially there is one unit of demand at each client $j \in C$. During the course of our algorithm, we move demands fractionally within $F \cup C$; moving α units of demand from i to j incurs a cost of $\alpha d(i,j)$. At the end, all the demands are moved to F and each facility $i \in F$ has at most $(1+O(\frac{1}{\ell}))u_i$ units of demand. We open a facility if it has positive amount of demand. Our goal is to bound the total moving cost by $O(\ell^5)\mathsf{LP}$ and the number of open facilities by k.

3 Representatives, Black Components, and Groups

Our algorithm starts with bundling facilities together with a three-phase process each of which creates bigger and bigger clusters. At the end, we have a nicely formed network of sufficiently big clusters of facilities. See Figure 1 for illustration of the three-phase clustering.

3.1 Representatives, Bundles and Initial Moving of Demands

In the first phase, we use a standard approach to facility location problems ([18, 19, 9, 16]) to partition the facilities into *bundles* $\{U_v\}_{v\in R}$, where each bundle U_v is associated with a center $v \in C$ that is called a *representative* and $R \subseteq C$ is the set of representatives. Each bundle U_v has a total opening at least $1/2$.

Let $R = \emptyset$ initially. Repeat the following process until C becomes empty: we select the client $v \in C$ with the smallest $d_{\mathsf{av}}(v)$ and add it to R; then we remove all clients j such that $d(j,v) \le 4d_{\mathsf{av}}(j)$ from C (thus, v itself is removed). We use v and its variants to index representatives, and j and its variants to index general clients. The family $\{U_v : v \in R\}$ is the Voronoi diagram of F with R being the centers: let $U_v = \emptyset$ for every $v \in R$ initially; for

each location $i \in F$, we add i to U_v for $v \in R$ that is closest to i. For any subset $V \subseteq R$, we use $U(V) := \bigcup_{v \in V} U_v$ to denote the union of Voronoi regions with centers V.

▶ **Lemma 1.** *The following statements hold:*

- **(1a)** *for all $v, v' \in R, v \neq v'$, we have $d(v, v') > 4\max\{d_{\mathsf{av}}(v), d_{\mathsf{av}}(v')\}$*
- **(1b)** *for all $j \in C$, there exists $v \in R$, such that $d_{\mathsf{av}}(v) \leq d_{\mathsf{av}}(j)$ and $d(v, j) \leq 4d_{\mathsf{av}}(j)$;*
- **(1c)** *$y_{U_v} \geq 1/2$ for every $v \in R$;*
- **(1d)** *for any $v \in R$, $i \in U_v$, and $j \in C$, we have $d(i, v) \leq d(i, j) + 4d_{\mathsf{av}}(j)$.*

The next lemma shows that moving demands from facilities to their corresponding representative doesn't cost much.

▶ **Lemma 2.** *For every $v \in R$, we have $\sum_{i \in U_v} x_{i,C} d(i, v) \leq O(1) D_{U_v}$.*

Since $\{U_v : v \in R\}$ forms a partition of F, we get the following corollary.

▶ **Corollary 3.** $\sum_{v \in R, i \in U_v} x_{i,C} d(i, v) \leq O(1)\mathsf{LP}$.

Initial Moving of Demands. With this corollary, we now move all the demands from C to V. First for every $j \in C$ and $i \in F$, we move $x_{i,j}$ units of demand from j to i. The moving cost of this step is exactly LP. After the step, all demands are at F and every $i \in F$ has $x_{i,C}$ units of demand. Then, for every $v \in R$ and $i \in U_v$, we move the $x_{i,C}$ units of demand at i to v. The moving cost for this step is $O(1)\mathsf{LP}$. Thus, after the initial moving, all demands are at the set R of representatives: a representative v has $x_{U_v,C}$ units of demand.

3.2 Black Components

In the second phase, we employ the minimum-spanning-tree construction of [16] to partition the set R of representatives into a family $\mathcal{J}$ of so-called *black components*. There is a degree-2 rooted forest $\Upsilon^*_{\mathcal{J}}$ over $\mathcal{J}$ with many good properties. For example, each non-root black component is not far away from its parent, and each root black component of $\Upsilon^*_{\mathcal{J}}$ contains a total opening of $\Omega(\ell)$. (For simplicity, we say the total opening at a representative $v \in R$ is y_{U_v}, which is the total opening at the bundle U_v.) The forest in [16] can have a large degree, while our algorithm requires the forest to have degree 2. This property is guaranteed by using the left-child-right-sibling representation.

Due to the page limit, we leave the description of the framework of [16] to the full version of the paper, and give its summary in the following lemma:

▶ **Lemma 4.** *There is an efficient algorithm to partition R into a set $\mathcal{J}$ of* black components *(or components, for simplicity) and construct a rooted forest $\Upsilon^*_{\mathcal{J}}$ over $\mathcal{J}$, such that if we let $L(J) = d(J, R \setminus J)$ for every black component $J \in \mathcal{J}$, then the following properties hold:*

- **(4a)** *for every $J \in \mathcal{J}$, there is a spanning tree over the representatives in J such that for every edge (v, v') in the spanning tree we have $d(v, v') \leq L(J)$;*
- **(4b)** *every root component $J \in \mathcal{J}$ of $\Upsilon^*_{\mathcal{J}}$ has $y_{U(J)} \geq \ell$ and every non-root component $J \in \mathcal{J}$ has $y_{U(J)} < \ell$;*
- **(4c)** *every root component $J \in \mathcal{J}$ of $\Upsilon^*_{\mathcal{J}}$ has either $y_{U(J)} < 2\ell$ or $|J| = 1$;*
- **(4d)** *for any non-root component J and its parent J', we have $L(J) \geq L(J')$;*
- **(4e)** *for any non-root component J and its parent J', we have $d(J, J') \leq O(\ell)L(J)$;*
- **(4f)** *every component J has at most two children.*

3.3 Groups

In the third phase, we apply a simple greedy algorithm to the forest $\Upsilon^*_{\mathcal{J}}$ to partition the set $\mathcal{J}$ of black components into a family $\mathcal{G}$ of *groups*, where each group $G \in \mathcal{G}$ contains many black components that are connected in $\Upsilon^*_{\mathcal{J}}$. By contracting each group $G \in \mathcal{G}$, the forest $\Upsilon^*_{\mathcal{J}}$ over the set $\mathcal{J}$ of black components becomes a forest $\Upsilon_{\mathcal{G}}$ over the set $\mathcal{G}$ of groups. Each group has a total opening of $\Omega(\ell)$, unless it is a leaf-group in $\Upsilon_{\mathcal{G}}$.

We partition the set $\mathcal{J}$ into groups using a technique similar to [4, 6]. For each rooted tree $T = (\mathcal{J}_T, E_T)$ in $\Upsilon^*_{\mathcal{J}}$, we construct a group G of black components as follows. Initially, let G contain the root component of T. While $\sum_{J \in G} y_{U(J)} < \ell$ and $G \neq \mathcal{J}_T$, repeat the following procedure. Choose the component $J \in \mathcal{J}_T \setminus G$ that is adjacent to G in T, with the smallest L-value, and add J to G.

Thus, by the construction G is connected in T. After we have constructed the group G, we add G to $\mathcal{G}$. We remove all black components in G from T. Then, each T is broken into many rooted trees; we apply the above procedure recursively for each rooted tree.

So, we have constructed a partition $\mathcal{G}$ for the set $\mathcal{J}$ of components. If for every $G \in \mathcal{G}$, we contract all components in G into a single node, then the rooted forest $\Upsilon^*_{\mathcal{J}}$ over $\mathcal{J}$ becomes a rooted forest $\Upsilon_{\mathcal{G}}$ over the set $\mathcal{G}$ of groups. $\Upsilon_{\mathcal{G}}$ naturally defines a parent-child relationship over $\mathcal{G}$. The following lemma uses Properties **(4a)** to **(4f)** of $\mathcal{J}$ and the way we construct $\mathcal{G}$.

▶ **Lemma 5.** *The following statements hold for the set $\mathcal{G}$ of groups and the rooted forest $\Upsilon_{\mathcal{G}}$ over $\mathcal{G}$:*

- **(5a)** *any root group $G \in \mathcal{G}$ contains a single root component $J \in \mathcal{J}$;*
- **(5b)** *if $G \in \mathcal{G}$ is not a root group, then $\sum_{J \in G} y_{U(J)} < 2\ell$;*
- **(5c)** *if $G \in \mathcal{G}$ is a non-leaf group, then $\sum_{J \in G} y_{U(J)} \geq \ell$;*
- **(5d)** *let $G \in \mathcal{G}, G' \in \mathcal{G}$ be the parent of G, $J \in G$ and $v \in J$, then the distance between v and any representative in $\bigcup_{J' \in G'} J'$ is at most $O(\ell^2)L(J)$;*
- **(5e)** *any group G has at most $O(\ell)$ children.*

4 Constructing Local Solutions

In this section, we shall construct a local solution, or a distribution of local solutions, for a given set $V \subseteq R$ which is the union of some black components. A local solution for V contains a pair $(S \subseteq U(V), \beta \in \mathbb{R}_{\geq 0}^{U(V)})$, where S is the facilities we open in $U(V)$ and β_i for each $i \in U(V)$ is the amount of supply at i: the demand that can be satisfied by i. Thus $\beta_i = 0$ if $i \in U(V) \setminus S$. We shall use the supplies at $U(V)$ to satisfy the $x_{U(V),C}$ demands at V after the initial moving of demands; thus, we require $\sum_{i \in U(V)} \beta_i = x_{U(V),C}$. There are two other main properties we need the distribution to satisfy: (a) the expected size of S from the distribution is not too big, and (b) the cost of matching the demands at V and the supplies at $U(V)$ is small.

We distinguish between *concentrated* black components and *non-concentrated* black components. Roughly speaking, a component $J \in \mathcal{J}$ is concentrated if in the fractional solution (x, y), for most clients $j \in C$, j is either almost fully served by facilities in $U(J)$, or almost fully served by facilities in $F \setminus U(J)$. We shall construct a distribution of local solutions for each concentrated component J. We require Constraints (6) to (13) to be satisfied for $B = U(J)$ (if not, we return the set $U(J)$ to the separation oracle) and let z^B be the vector satisfying the constraints. Roughly speaking, the z^B-vector defines a distribution of local solutions for V. A local solution (S, β) is good if S is not too big and the total demand $\sum_{i \in S} \beta_i$ satisfied by S is not too small. Then, our algorithm randomly selects (S, β)

from the distribution defined by z^B, under the condition that (S, β) is good. The fact that J is concentrated guarantees that the total mass of good local solutions in the distribution is large; therefore the factors we lose due to the conditioning are small.

For non-concentrated components, we construct a single local solution (S, β), instead of a distribution of local solutions. Moreover, the construction is for the union V of some non-concentrated components, instead of an individual component. The components that comprise V are close to each other; by the fact that they are non-concentrated, we can move demands arbitrarily within V, without incurring too much cost. Thus we can essentially treat the distances between representatives in V as 0. Then we are only concerned with two parameters for each facility $i \in U(V)$: the distance from i to V and the capacity u_i. Using a simple argument, the optimum fractional local solution (that minimizes the cost of matching the demands and supplies) is almost integral: it contains at most 2 fractionally open facilities. By fully opening the two fractional facilities, we find an integral local solution with small number of open facilities.

The remaining part of this section is organized as follows. We first formally define concentrated black components, and explain the importance of the definition. We then define the earth-mover-distance, which will be used to measure the cost of satisfying demands using supplies. The construction of local solutions for concentrated components and non-concentrated components will be stated in Theorem 9 and Lemma 10 respectively. Due to the page limit, their proofs will only appear in the full version of the paper.

Concentrated Black Components. The definition of concentrated black component is the same as that of [16], except that we choose the parameter ℓ_2 differently.

▶ **Definition 6.** Define $\pi_J = \sum_{j \in C} x_{U(J),j}(1 - x_{U(J),j})$, for every black component $J \in \mathcal{J}$. A black component $J \in \mathcal{J}$ is said to be *concentrated* if $\pi_J \le x_{U(J),C}/\ell_2$, and *non-concentrated* otherwise, where $\ell_2 = \Theta(\ell^3)$ is large enough.

We use $\mathcal{J}^{\mathsf{C}}$ to denote the set of concentrated components and $\mathcal{J}^{\mathsf{N}}$ to denote the set of non-concentrated components. The next lemma from [16] shows the importance of π_J.

▶ **Lemma 7.** *For any $J \in \mathcal{J}$, we have $L(J)\pi_J \le O(1)D_{U(J)}$.*

Recall that $L(J) = d(J, R \setminus J)$ and $x_{U(J),C}$ is the total demand in J after the initial moving. Thus, according to Lemma 7, if J is not concentrated, we can use $D_{U(J)}$ to charge the cost for moving all the $x_{U(J),C}$ units of demand out of J, provided that the moving distance is not too big compared to $L(J)$. This gives us freedom for handling non-concentrated components. If J is concentrated, the amount of demand that is moved out of J must be comparable to π_J; this will be guaranteed by the configuration LP.

Earth Mover Distance. In order to measure the moving cost of satisfying demands using supplies, we define the earth mover distance:

▶ **Definition 8** (Earth Mover Distance)**.** Given a set $V \subseteq R$ with $B = U(V)$, a demand vector $\alpha \in \mathbb{R}^V_{\ge 0}$ and a supply vector $\beta \in \mathbb{R}^B_{\ge 0}$ such that $\sum_{v \in V} \alpha_v \le \sum_{i \in B} \beta_i$, the earth mover distance from α to β is defined as $\mathrm{EMD}_V(\alpha, \beta) := \inf_f \sum_{v \in V, i \in B} f(v,i)d(v,i)$, where f is over all functions from $V \times B$ to $\mathbb{R}_{\ge 0}$ such that

- $\sum_{i \in B} f(v,i) = \alpha_v$ for every $v \in V$;
- $\sum_{v \in V} f(v,i) \le \beta_i$ for every $i \in B$.

For some technical reason, we allow some fraction of a supply to be unmatched. From now on, we shall use $\alpha_v = x_{U_v,C}$ to denote the amount of demand at v after the initial moving. For any set $V \subseteq R$ of representatives, we use $\alpha|_V$ to denote the vector α restricted to the coordinates in V.

We now summarize our constructions of local solutions for concentrated and non-concentrated black components, respectively.

▶ **Theorem 9.** *Let $J \in \mathcal{J}^{\mathsf{C}}$ and let $B = U(J)$. Assume Constraints (6) to (13) are satisfied for B. Then, we can find a distribution $(\phi_{S,\beta})_{S\subseteq B,\beta\in\mathbb{R}^B_{\geq 0}}$ of pairs (S,β), such that*

(9a) *$s_\phi := \mathbb{E}_{(S,\beta)\sim\phi}|S| \in [y_B, y_B(1+2\ell\pi_J/x_{B,C})]$, and $s_\phi = y_B$ if $y_B > 2\ell$,*

and for every (S,β) in the support of ϕ, we have

(9b) *$|S| \in \{\lfloor s_\phi \rfloor, \lceil s_\phi \rceil\}$;*
(9c) *$\beta_i \leq (1+O(1/\ell))u_i$ if $i \in S$ and $\beta_i = 0$ if $i \in B \setminus S$;*
(9d) *$\sum_{i\in S}\beta_i = x_{B,C} = \sum_{v\in J}\alpha_v$.*

Moreover, the distribution ϕ satisfies

(9e) *the support of ϕ has size at most $n^{O(\ell)}$;*
(9f) *$\mathbb{E}_{(S,\beta)\sim\phi}\mathrm{EMD}_J(\alpha|_J,\beta) \leq O(\ell^4)D_B$.*

▶ **Lemma 10.** *Let $\mathcal{J}' \subseteq \mathcal{J}^{\mathsf{N}}$ be a set of non-concentrated black components, $V = \bigcup_{J\in\mathcal{J}'} J$ and $B = U(V)$. Assume there exists $v^* \in R$ such that $d(v,v^*) \leq O(\ell^2)L(J)$ for every $J \in \mathcal{J}'$ and $v \in J$. Then, we can find a pair $(S \subseteq B, \beta \subseteq \mathbb{R}^B_{\geq 0})$ such that*

(10a) *$|S| \in \{\lceil y_B \rceil, \lceil y_B \rceil + 1\}$;*
(10b) *$\beta_i \leq u_i$ if $i \in S$ and $\beta_i = 0$ if $i \in B \setminus S$;*
(10c) *$\sum_{i\in S}\beta_i = x_{B,C} = \sum_{v\in V}\alpha_v$;*
(10d) *$\mathrm{EMD}_V(\alpha|_V,\beta) \leq O(\ell^2\ell_2)D_B$.*

5 Rounding Algorithm

In this section we describe our rounding algorithm. We start by giving the intuition behind the algorithm. For each concentrated component $J \in \mathcal{J}$, we construct a distribution of local solutions using Theorem 9. We shall construct a partition $\mathcal{V}^{\mathsf{N}}$ of the representatives in $\bigcup_{J\in\mathcal{J}^{\mathsf{N}}} J$ so that each $V \in \mathcal{V}^{\mathsf{N}}$ is the union of some nearby components in $\mathcal{J}^{\mathsf{N}}$. For each set $V \in \mathcal{V}^{\mathsf{N}}$, we apply Lemma 10 to construct a local solution. If we independently and randomly choose a local solution from every distribution we constructed, then we can move all the demands to the open facilities at a small cost, by Property **(9f)** and Property **(10d)**.

However, we may open more than k facilities, even in expectation. Noticing that the fractional solution opens y_B facilities in a set B, the extra number of facilities come from two places. In Property **(9a)** of Theorem 9, we may open in expectation $y_B \cdot 2\ell\pi_J/x_{B,C}$ more facilities in B than y_B. Then in Property **(10a)** of Lemma 10, we may open $\lceil y_B \rceil$ or $\lceil y_B \rceil + 1$ facilities in B. To reduce the number of open facilities to k, we shall shut down (or remove) some already-open facilities and move the demands satisfied by these facilities to the survived open facilities: a concentrated component $J \in \mathcal{J}^{\mathsf{C}}$ is responsible for removing $y_B \cdot 2\ell\pi_J/x_{B,C} < 1$ facilities in expectation; a set $V \in \mathcal{V}^{\mathsf{N}}$ is responsible for removing up to 2 facilities. Lemma 7 allows us to bound the cost of moving demands caused by the removal, provided that the moving distance is not too big. To respect the capacity constraint up to a factor of $1+\epsilon$, we are only allowed to scale the supplies of the survived open facilities by a factor of $1+O(1/\ell)$. Both requirements will be satisfied by the forest structure over groups and the fact that each non-leaf group contains $\Omega(\ell)$ fractional opening (Property **(5c)**). Due

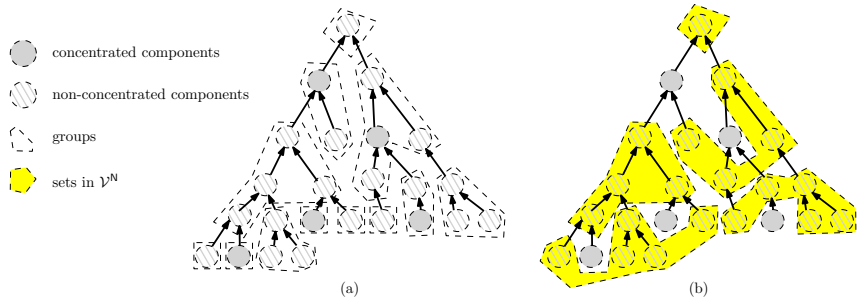


Figure 2 Figure (a) gives the forest $\Upsilon^*_{\mathcal{J}}$ over $\mathcal{J}$ and the set $\mathcal{G}$ of groups (denoted by empty polygons). Figure (b) gives $\mathcal{V}^{\mathsf{N}}$: each set $V \in \mathcal{V}^{\mathsf{N}}$ is the union of components in a solid polygon.

to the forest structure and Property **(5c)**, we always have enough open facilities locally that can support the removing of facilities.

In order to guarantee that we always open k facilities, we need to use a dependent rounding procedure for opening and removing facilities. As in many of previous algorithms, we incorporate the randomized rounding procedure into random selections of vertex points of polytopes respecting marginal probabilities. In many cases, a randomized selection procedure can be derandomized since there is an explicit linear objective we shall optimize.

We now formally describe our rounding algorithm. For every group $G \in \mathcal{G}$, we use Λ_G to denote the set of child-groups of G. We construct a partition $\mathbb{J}^{\mathsf{C}}$ of $\mathcal{J}^{\mathsf{C}}$ as follows. For each root group $G \in \mathcal{G}$, we add $G \cap \mathcal{J}^{\mathsf{C}}$ to $\mathbb{J}^{\mathsf{C}}$ if it is not empty. For each non-leaf group $G \in \mathcal{G}$, we add $\bigcup_{G' \in \Lambda_G}(G' \cap \mathcal{J}^{\mathsf{C}})$ to $\mathbb{J}^{\mathsf{C}}$, if it is not empty. We construct the partition $\mathbb{J}^{\mathsf{N}}$ for $\mathcal{J}^{\mathsf{N}}$ in the same way, except that we consider components in $\mathcal{J}^{\mathsf{N}}$. We also define a set $\mathcal{V}^{\mathsf{N}}$ as follows: for every $\mathcal{J}' \in \mathbb{J}^{\mathsf{N}}$, we add $\bigcup_{J \in \mathcal{J}'} J$ to $\mathcal{V}^{\mathsf{N}}$; thus, $\mathcal{V}^{\mathsf{N}}$ forms a partition for $\bigcup_{J \in \mathcal{J}^{\mathsf{N}}} J$. See Figure 2 for the definition of $\mathcal{V}^{\mathsf{N}}$.

In Section 5.1, we describe the procedure for opening a set S^* of facilities, whose cardinality may be larger than k. Then in Section 5.2, we define the procedure remove, which removes one open facility. We wrap up the algorithm in Section 5.3.

5.1 Constructing Initial Set S^* of Open Facilities

In this section, we open a set S^* of facilities, whose cardinality may be larger than k, and construct a supply vector $\beta^* \in \mathbb{R}^F_{\geq 0}$ such that $\beta^*_i = 0$ if $i \notin S^*$. (S^*, β^*) will be the concatenation of all local solutions we constructed.

It is easy to construct local solutions for non-concentrated components. For each set $\mathcal{J}' \in \mathbb{J}^{\mathsf{N}}$ of components and its correspondent $V = \bigcup_{J \in \mathcal{J}'} J \in \mathbb{J}^{\mathsf{N}}$, we apply Lemma 10 to obtain a local solution $\left(S \subseteq U(V), \beta \in \mathbb{R}^{U(V)}_{\geq 0}\right)$. Then, we add S to S^* and let $\beta^*_i = \beta_i$ for every $i \in U(V)$. Notice that $\mathcal{J}'$ either contains a single root black component J, or contains all the non-concentrated black components in the child-groups of some group G. In the former case, the diameter of J is at most $O(\ell)L(J)$ by Property **(4a)**; in the latter case, we let v^* be an arbitrary representative in $\bigcup_{J' \in G} J'$ and then any representative $v \in J, J \in \mathcal{J}'$ has $d(v, v^*) \leq O(\ell^2)L(J)$ by Property **(5d)**. Thus, all the properties in Lemma 10 are satisfied.

For concentrated components, we only obtain distributions of local solutions by applying Theorem 9. For every $J \in \mathcal{J}^{\mathsf{C}}$, we check if Constraints (6) to (13) are satisfied for $B = U(J)$. If not, we return a separation plane for the fractional solution; otherwise we apply Theorem 9 to each component J to obtain a distribution $\left(\phi^J_{S,\beta}\right)_{S \subseteq U(J), \beta \in \mathbb{R}^{U(J)}_{\geq 0}}$. To produce local solutions for concentrated components, we shall use a dependent rounding procedure that respects the

marginal probabilities. As mentioned earlier, we shall define a polytope and the procedure randomly selects a vertex point of the polytope.

We let $s_J := s_{\phi_J} := \mathbb{E}_{(S,\beta)\sim\phi^J}|S|$ be the expectation of $|S|$ according to distribution ϕ^J. For notational convenience, we shall use $a \approx b$ to denote $a \in \left[\lfloor b \rfloor, \lceil b \rceil\right]$. Consider the following polytope $\mathcal{P}$ defined by variables $\{\psi^J_{S,\beta}\}_{J\in\mathcal{J}^{\mathsf{C}},S,\beta}$ and $\{q_J\}_{J\in\mathcal{J}^{\mathsf{C}}}$.[1]

$$\psi^J_{S,\beta}, p_J \in [0,1] \qquad \forall J \in \mathcal{J}^{\mathsf{C}}, S, \beta; \tag{14}$$

$$\sum_{S,\beta} \psi^J_{S,\beta} = 1, \qquad \forall J \in \mathcal{J}^{\mathsf{C}}; \tag{15}$$

$$\sum_{J\in\mathcal{J}'} q_J \le 1, \qquad \forall \mathcal{J}' \in \mathbb{J}^{\mathsf{C}}; \tag{16}$$

$$\sum_{S,\beta} \psi^J_{S,\beta}|S| - q_J \approx y_{U(J)}, \qquad \forall J \in \mathcal{J}^{\mathsf{C}}; \tag{17}$$

$$\sum_{J\in\mathcal{J}'} \left(\sum_{S,\beta} \psi^J_{S,\beta}|S| - q_J\right) \approx \sum_{J\in\mathcal{J}'} y_{U(J)}, \quad \forall \mathcal{J}' \in \mathbb{J}^{\mathsf{C}}; \tag{18}$$

$$\sum_{J\in\mathcal{J}^{\mathsf{C}}} \left(\sum_{S,\beta} \psi^J_{S,\beta}|S| - q_J\right) \approx \sum_{J\in\mathcal{J}^{\mathsf{C}}} y_{U(J)}. \tag{19}$$

In the above LP, ψ^J is the indicator vector for local solutions for J and q_J indicates whether J is responsible for removing one facility; if $q_J = 1$, we shall call $\mathsf{remove}(J)$ later. Up to changing of variables, any vertex point of $\mathcal{P}$ is defined by two laminar families of tight constraints and thus $\mathcal{P}$ is integral:

▶ **Lemma 11.** *$\mathcal{P}$ is integral.*

We set $\psi^{*J}_{S,\beta} = \phi^J_{S,\beta}$ and $q^*_J = s_J - y_{U(J)}$ for every $J \in \mathcal{J}^{\mathsf{C}}$ and (S,β). Then,

▶ **Lemma 12.** *(ψ^*, q^*) is a point in polytope $\mathcal{P}$.*

We randomly select a vertex point (ψ, q) of $\mathcal{P}$ such that $\mathbb{E}[\psi^J_{S,\beta}] = \psi^{*J}_{S,\beta} = \phi^J_{S,\beta}$ for every $J \in \mathcal{J}^{\mathsf{C}}, (S,\beta)$, and $\mathbb{E}[q_J] = q^*_J = s_J - y_{U(J)}$ for every $J \in \mathcal{J}^{\mathsf{C}}$. Since ψ is integral, for every $J \in \mathcal{J}$, there is a unique local solution $\left(S \subseteq U(J), \beta \in \mathbb{R}^{U(J)}_{\ge 0}\right)$ such that $\psi^J_{S,\beta} = 1$; we add S to S^* and let $\beta^*_i = \beta_i$ for every $i \in U(J)$.

This finishes the definition of the initial S^* and β^*. Let $\alpha^* = \alpha$ (recall that $\alpha_v = x_{U_v,C}$ is the demand at v after the initial moving, for every $v \in R$) be the initial demand vector. Later we shall remove facilities from S^* and update α^* and β^*. S^*, α^*, β^* satisfy the following properties, which will be maintained as the rounding algorithm proceeds.

(13a) $\sum_{v\in V} \alpha^*_v = \sum_{v\in V} \beta^*_v$ for every $V \in \mathcal{J}^{\mathsf{C}} \cup \mathcal{V}^{\mathsf{N}}$;
(13b) $\sum_{v\in R} \alpha^*_v = |C|$.

Property **(13a)** is due to Properties **(9d)** and **(10c)**. Property **(13b)** holds since $\sum_{v\in R} \alpha^*_v = \sum_{v\in R} x_{U_v,C} = x_{F,C} = |C|$.

5.2 The remove procedure

In this section, we define the procedure remove that removes facilities from S^* and updates α^* and β^*. The procedure takes a set $V \in \mathcal{J}^{\mathsf{C}} \cup \mathcal{V}^{\mathsf{N}}$ as input. If V is a root black component,

[1] For every $J \in \mathcal{J}^{\mathsf{C}}$, we only consider the pairs (S,β) in the support of ϕ^J; thus the total number of variables is $n^{O(\ell)}$.

then we let $G = \{V\}$ be the root group containing V; if V is a non-root concentrated component, let G be the parent group of the group containing V; otherwise V is the union of non-concentrated components in all child-groups of some group, and we let G be this group. Let $V' = \bigcup_{J' \in G} J'$. Before calling $\mathsf{remove}(V)$, we require the following properties to hold:

- **(14a)** $|S^* \cap U(V)| \geq 1$;
- **(14b)** $\left|S^* \cap U(V')\right| \geq \ell - 6$.

While maintaining Properties **(13a)** and **(13b)**, the procedure $\mathsf{remove}(V)$ will

- **(15a)** remove from S^* exactly one open facility, which is in $U(V \cup V')$,
- **(15b)** not change $\alpha^*|_{R\setminus(V\cup V')}$ and $\beta^*|_{F\setminus U(V\cup V')}$,
- **(15c)** increase α^*_v by at most a factor of $1 + O(1/\ell)$ for every $v \in V \cup V'$ and increase β^*_i by at most a factor of $1 + O(1/\ell)$ for every $i \in U(V \cup V')$.

Moreover,

- **(15d)** the moving cost for converting the old α^* to the new α^* is at most $O(\ell^2)\beta^*_{i^*} L(J)$ for some black component $J \subseteq V$ and facility $i^* \in U(J)$;
- **(15e)** for every $V'' \in \mathcal{J}^{\mathsf{C}} \cup \mathcal{V}^{\mathsf{N}}$, $\mathrm{EMD}_{V''}\big(\alpha^*|_{V''}, \beta^*|_{U(V'')}\big)$ will be increased by at most a factor of $1 + O(1/\ell)$.

Due to the page limit, we only highlight the key ideas used to implement $\mathsf{remove}(V)$ and leave the formal description to the full version of the paper. Assume V is not a root component. We choose an arbitrary facility $i \in S^* \cap U(V)$. Notice that there are $\Omega(\ell)$ facilities in $S^* \cap U(V')$. If the $\beta^*_i \leq \sum_{v' \in V'} \alpha^*_{v'}/\ell$, then we can shut down i and send the demands that should be sent to i to V'. We only need to increase the supplies in $U(V')$ by a factor of $1 + O(1/\ell)$. Otherwise, we shall shut down the facility $i' \in S^* \cap U(V')$ with the smallest $\beta^*_{i'}$ value. Since there are at least $\Omega(\ell)$ facilities in $U(V')$, we can satisfy the $\beta^*_{i'}$ units of unsatisfied demands using other facilities in $S^* \cap U(V')$. For this i', we have $\beta^*_{i'} \leq O(1)\beta^*_i$. Thus, the total amount of demands that will be moved is comparable to β^*_i. In either case, the cost of redistributing the demands is not too big. When V is a root component, we shall shut down the facility $i' \in S^* \cap U(V)$ with the smallest $\beta^*_{i'}$ value.

5.3 Obtaining the Final Solution

To obtain our final set S^* of facilities, we call the remove procedures in some order. We consider each group G using the top-to-bottom order. That is, before we consider a group G, we have already considered its parent group. If G is a root group, then it contains a single root component J. If $J \in \mathcal{J}^{\mathsf{N}}$, repeat the the following procedure twice: if there is some facility in $S^* \cap U(J)$ then we call $\mathsf{remove}(J)$. If $J \in \mathcal{J}^{\mathsf{C}}$ and $q_J = 1$ then we call $\mathsf{remove}(J)$. Now if G is a non-leaf group, then do the following. Let $V = \bigcup_{G' \in \Lambda_G, J \in G' \cap \mathcal{J}^{\mathsf{N}}} J$. Repeat the following procedure twice: if there is some facility in $S^* \cap U(V)$ then we call $\mathsf{remove}(V)$. For every $G' \in \Lambda_G$ and $J \in G' \cap \mathcal{J}^{\mathsf{C}}$ such that $q_J = 1$ we call $\mathsf{remove}(J)$.

▶ **Lemma 16.** *After the above procedure, we have $|S^*| \leq y_F \leq k$.*

By Properties **(15b)** and **(15c)**, and Constraint (16), our final β^*_i is at most $1 + O(1/\ell)$ times the initial β^*_i for every $i \in V$. Finally we have $\beta^*_i \leq (1 + O(1/\ell))u_i$ for every $i \in F$. Thus, the capacity constraint is violated by a factor of $1 + \epsilon$ if we set ℓ to be large enough.

It remains to bound the expected cost of the solution S^*; this is done by bounding the cost for transferring the original α^* to the final α^*, as well as the cost for matching our final α^* and β^*.

We first focus on the transferring cost. By Property **(15e)**, when we call $\mathsf{remove}(V)$, the transferring cost is at most $O(\ell^2)\beta^*_{i^*} L(J)$ for some black component $J \subseteq V$ and i^*. Notice that

$\beta^*_{i^*}$ is scaled by at most a factor of $(1 + O(1/\ell))$, we always have $\beta^*_{i^*} \leq (1 + O(1/\ell))\alpha_{U(J),C}$. So, the cost is at most $O(\ell^2)x_{U(J),C}L(J)$. If V is the union of some non-concentrated components, then this quantity is at most $O(\ell^2)\ell_2\pi_J L(J) \leq O(\ell^2\ell_2)D_{U(J)} \leq O(\ell^2\ell_2)D_{U(V)}$. We call $\mathsf{remove}(V)$ at most twice, thus the contribution of V to the transferring cost is at most $O(\ell^2\ell_2)D_{U(V)}$. If V is a concentrated component J, then the quantity might be large. However, the probability we call $\mathsf{remove}(J)$ is $\mathbb{E}[q_J] = q^*_J = s_J - y_{U(J)} \leq 2\ell y_{U(J)}\pi_J/x_{U(J),C}$ if $y_{U(J)} \leq 2\ell$ and it is 0 otherwise (by Property **(9a)**). So, the expected contribution of this V to the transferring cost is at most $O(\ell^2)x_{U(J),C}L(J) \times 2\ell y_{U(J)}\pi_J/x_{U(J),C} \leq O(\ell^4)\pi_J L(J) \leq O(\ell^4)D_{U(J)}$ by Lemma 7. Thus, overall, the expected transferring cost is at most $O(\ell^5)D_F = O(\ell^5)\mathsf{LP}$.

Then we consider the matching cost. Since we maintained Property **(13a)**, the matching cost is bounded by $\sum_{V\in\mathcal{J}^{\mathsf{C}}\cup\mathcal{V}^{\mathsf{N}}} \mathrm{EMD}_V(\alpha^*|_V, \beta^*|_{U(V)})$. Due to Property **(15e)**, this quantity has only increased by a factor of $1 + O(1/\ell)$ during the course of removing facilities. For the initial α^* and β^*, the expectation of this quantity is at most $\sum_{J\in\mathcal{J}^{\mathsf{C}}} O(\ell^4)D_{U(J)} + \sum_{V\in\mathcal{V}^{\mathsf{N}}} O(\ell^2\ell_2)D_{U(V)}$ due to Properties **(9f)** and **(10d)**. This is at most $O(\ell^5)D_F = O(\ell^5)\mathsf{LP}$.

We have found a set S^* of at most k facilities and a vector $\beta^* \in \mathbb{R}^F_{\geq 0}$ such that $\beta^*_i = 0$ for every $i \notin S^*$ and $\beta^*_i \leq (1 + O(1/\ell))u_i$. If we set $\ell = \Theta(1/\epsilon)$ to be large enough, then $\beta^*_i \leq (1 + \epsilon)u_i$. The cost for matching the α-demand vector and the β^* vector is at most $O(\ell^5)\mathsf{LP} = O(1/\epsilon^5)\mathsf{LP}$. Thus, we obtained a $O(1/\epsilon^5)$-approximation for CKM with $(1 + \epsilon)$-capacity violation.

References

1. Karen Aardal, Pieter L. van den Berg, Dion Gijswijt, and Shanfei Li. Approximation algorithms for hard capacitated k-facility location problems. *European Journal of Operational Research*, 242(2):358–368, 2015. doi:10.1016/j.ejor.2014.10.011.
2. Hyung-Chan An, Mohit Singh, and Ola Svensson. LP-based algorithms for capacitated facility location. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2014*, 2014.
3. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, STOC'01, pages 21–29, New York, NY, USA, 2001. ACM. doi:10.1145/380752.380755.
4. Jarosław Byrka, Krzysztof Fleszar, Bartosz Rybicki, and Joachim Spoerhase. Bi-factor approximation algorithms for hard capacitated k-median problems. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, 2015.
5. Jarosław Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, 2015.
6. Jarosław Byrka, Bartosz Rybicki, and Sumedha Uniyal. An approximation algorithm for uniform capacitated k-median problem with $1+\epsilon$ capacity violation, 2015. arXiv:1511.07494. arXiv:arXiv:1511.07494.
7. Robert D. Carr, Lisa K. Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'00, pages 106–115, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=338219.338241.

8 M. Charikar and S. Guha. Improved combinatorial algorithms for the facility location and k-median problems. In *In Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

9 M. Charikar, S. Guha, É. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem (extended abstract). In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, STOC'99, pages 1–10, New York, NY, USA, 1999. ACM. doi:10.1145/301250.301257.

10 Julia Chuzhoy and Yuval Rabani. Approximating k-median with non-uniform capacities. In *SODA '05*, pages 952–958, 2005.

11 Sudipto Guha. *Approximation Algorithms for Facility Location Problems*. PhD thesis, Stanford University, Stanford, CA, USA, 2000.

12 K. Jain, M. Mahdian, and A. Saberi. A new greedy approach for facility location problems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC'02, pages 731–740, New York, NY, USA, 2002. ACM. doi:10.1145/509907.510012.

13 K Jain and V. V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001. doi:10.1145/375827.375845.

14 Shanfei Li. An improved approximation algorithm for the hard uniform capacitated k-median problem. In *APPROX'14/RANDOM'14: Proceedings of the 17th International Workshop on Combinatorial Optimization Problems and the 18th International Workshop on Randomization and Computation*, APPROX'14/RANDOM'14, 2014.

15 Shi Li. On uniform capacitated k-median beyond the natural LP relaxation. In *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2015)*, 2015.

16 Shi Li. Approximating capacitated k-median with $(1+\epsilon)k$ open facilities. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2016)*, pages 786–796, 2016.

17 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC'13, pages 901–910, New York, NY, USA, 2013. ACM. doi:10.1145/2488608.2488723.

18 J. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC), Victoria, British Columbia, Canada*, pages 771–782, 1992.

19 D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *STOC'97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274, New York, NY, USA, 1997. ACM. doi:10.1145/258533.258600.

Approximating Directed Steiner Problems via Tree Embedding

Bundit Laekhanukit*

The Weizmann Institute of Science, Rehovot, Israel
bundit.laekhanukit@weizmann.ac.il

Abstract

Directed Steiner problems are fundamental problems in Combinatorial Optimization and Theoretical Computer Science. An important problem in this genre is the *k-edge connected directed Steiner tree* (k-DST) problem. In this problem, we are given a directed graph G on n vertices with edge-costs, a root vertex r, a set of h terminals T and an integer k. The goal is to find a min-cost subgraph $H \subseteq G$ that connects r to each terminal $t \in T$ by k edge-disjoint r,t-paths. This problem includes as special cases the well-known *directed Steiner tree* (DST) problem (the case $k = 1$) and the *group Steiner tree* (GST) problem. Despite having been studied and mentioned many times in literature, e.g., by Feldman et al. [SODA'09, JCSS'12], by Cheriyan et al. [SODA'12, TALG'14], by Laekhanukit [SODA'14] and in a survey by Kortsarz and Nutov [Handbook of Approximation Algorithms and Metaheuristics], there was no known non-trivial approximation algorithm for k-DST for $k \geq 2$ even in a special case that an input graph is directed acyclic and has a constant number of layers. If an input graph is not acyclic, the complexity status of k-DST is not known even for a very strict special case that $k = 2$ and $h = 2$.

In this paper, we make a progress toward developing a non-trivial approximation algorithm for k-DST. We present an $O(D \cdot k^{D-1} \cdot \log n)$-approximation algorithm for k-DST on directed acyclic graphs (DAGs) with D layers, which can be extended to a special case of k-DST on "general graphs" when an instance has a *D-shallow* optimal solution, i.e., there exist k edge-disjoint r,t-paths, each of length at most D, for every terminal $t \in T$. For the case $k = 1$ (DST), our algorithm yields an approximation ratio of $O(D \log h)$, thus implying an $O(\log^3 h)$-approximation algorithm for DST that runs in quasi-polynomial-time (due to the height-reduction of Zelikovsky [Algorithmica'97]). Our algorithm is based on an LP-formulation that allows us to embed a solution to a tree-instance of GST, which does not preserve connectivity. We show, however, that one can randomly extract a solution of k-DST from the tree-instance of GST.

Our algorithm is almost tight when k and D are constants since the case that $k = 1$ and $D = 3$ is NP-hard to approximate to within a factor of $O(\log h)$, and our algorithm archives the same approximation ratio for this special case. We also remark that the $k^{1/4-\epsilon}$-hardness instance of k-DST is a DAG with 6 layers, and our algorithm gives $O(k^5 \log n)$-approximation for this special case. Consequently, as our algorithm works for general graphs, we obtain an $O(D \cdot k^{D-1} \cdot \log n)$-approximation algorithm for a D-shallow instance of the *k edge-connected directed Steiner subgraph* problem, where we wish to connect every pair of terminals by k edge-disjoint paths.

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

Keywords and phrases Approximation Algorithms, Network Design, Graph Connectivity, Directed Graph

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.74

* The work was partly done while the author was at McGill University, Simons Institute for the Theory of Computing and the Swiss AI Lab IDSIA. Partially supported by the ERC Starting Grant NEWNET 279352 and by Swiss National Science Foundation project 200020_144491/1.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 74; pp. 74:1–74:13

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

1 Introduction

Network design is an important class of problems in Combinatorial Optimization and Theoretical Computer Science as it formulates scenarios that appear in practical settings. In particular, we might wish to design an overlay network that connects a server to clients, and this can be formulated as the *Steiner tree* problem. In a more general setting, we might have an additional constraint that the network must be able to function after link or node failures, leading to the formulation of the *survivable network design* problem. These problems are well-studied in symmetric case where a network can be represented by an undirected graph. However, in many practical settings, links in networks are not symmetric. For example, we might have different upload and download bandwidths in each connection, and sometimes, transmissions are only allowed in one direction. This motivates the study of network design problems in directed graphs, in particular, *directed Steiner* problems.

One of the most well-known directed network design problem is the *directed Steiner tree* problem (DST), which asks to find a minimum-cost subgraph that connects a given root vertex to each terminal. DST is a notorious problem as there is no known polynomial-time algorithm that gives an approximation ratio better than polynomial. A polylogarithmic approximation can be obtained only when an algorithm is allowed to run in quasi-polynomial-time [2, 13, 6]. A natural generalization of DST, namely, the k *edge-connected directed Steiner tree* (k-DST) problem, where we wish to connect a root vertex to each terminal by k edge-disjoint paths, is even more mysterious as there is no known non-trivial approximation algorithm, despite having been studied and mentioned many times in literature, e.g., by Feldman et al. [5], by Cheriyan et al. [3] and by Laekhanukit [10]. The problem is also mentioned in a survey by Kortsarz and Nutov [9] and in a later update by Nutov [12].

The focus of this paper is in studying the approximability of k-DST. Let us formally describe k-DST. In k-DST, we are given a directed graph G with edge-costs $\{c_e\}_{e\in E(G)}$, a root vertex r and a set of terminals $T \subseteq V(G)$. The goal is to find a min-cost subgraph $H \subseteq G$ such that H has a k edge-disjoint directed r,t-paths from the root r to each terminal $t \in T$. Thus, removing any $k-1$ edges from H leaves at least one path from the root r to each terminal $t \in T$, and DST is the case when $k = 1$ (i.e., we need only one path). The complexity status of k-DST tends to be negative. It was shown by Cheriyan et al. [3] that the problem is at least as hard as the *label cover* problem. Specifically, k-DST admits no $2^{\log^{1-\epsilon} n}$-approximation, for any $\epsilon > 0$, unless NP $\subseteq$ DTIME$(2^{\text{polylog}(n)})$. Laekhanukit [10], subsequently, showed that k-DST admits no $k^{1/4-\epsilon}$-approximation unless NP = ZPP. The integrality gap of a natural LP-relaxation for k-DST is $\Omega(k/\log k)$ which holds even for a special case of *connectivity-augmentation* where we wish to increase a connectivity of a graph by one. All the lower bound results are based on the same construction which are directed acyclic graphs (DAGs) with diameter 5, i.e., any path in an input graph has length (number of edges) at most 5 (we may also say that it has 6 *layers*). Even for a very simple variant of k-DST, namely $(1,2)$-DST, where we have two terminals, one terminal requires one path from the root and another terminal requires 2 edge-disjoint paths, it was not known whether the problem is NP-hard or polynomial-time solvable. To date, the only known positive result for k-DST is an $O(n^{kh})$-time (exact) algorithm for k-DST on DAGs [3], which thus runs in polynomial-time when kh is constant, and a folk-lore h-approximation algorithm, which can be obtained by computing min-cost k-flow for h times, one from the root r to each terminal t and then returning the union as a solution. We emphasize that there was no known non-trivial approximation algorithm even when an input graph is "directed acyclic" and has "constant number of layers". Also, in contrast to DST in which an $O(2^h \text{poly}(n))$-time (exact)

algorithm exists for general graphs, it is not known whether k-DST for $k = 2$ and $h = 2$ is polynomial-time solvable if an input graph is not acyclic. This leaves challenging questions whether ones can design a non-trivial approximation algorithm for k-DST on DAGs with at most D layers, and whether ones can design a non-trivial approximation algorithm when an input graph is not acyclic.

In this paper, we make a progress toward developing a non-trivial approximation algorithm for k-DST. We present the first "non-trivial" approximation algorithm for k-DST on DAGs with D layers that achieves an approximation ratio of $O(D \cdot k^{D-1} \cdot \log n)$. Our algorithm can be extended to a special case of k-DST on "general graphs" where an instance has a *D-shallow* optimal solution, i.e., there exist k edge-disjoint r, t-paths, each of length at most D, for every terminal $t \in T$. Consequently, as our algorithm works for a general graph, we obtain an $O(D \cdot k^{D-1} \cdot \log n)$-approximation algorithm for a D-shallow instance of the *k edge-connected directed Steiner subgraph* problem, where we wish to connect every pair of terminals by k edge-disjoint paths, i.e., the set of terminals T is required to be k-edge connected in the solution subgraph (there is no root vertex in this problem).

Our algorithm is almost tight when k and D are constants because the case that $k = 1$ and $D = 3$ is essentially the *set cover* problem, which is NP-hard to approximate to within a factor of $O(\log h)$ [11, 4], and our algorithm achieves the same approximation ratio. We also remark that the $k^{1/4-\epsilon}$-hardness instance of k-DST is a DAG with 6 layers, and our algorithm gives $O(k^5 \log n)$-approximation for this special case. For $k = 1$, we obtain a slightly better bound of $O(D \log h)$, thus giving an LP-based $O(\log^3 h)$-approximation algorithm for DST as a by product.

The key idea of our algorithm is to formulate an LP-relaxation with a special property that a fractional solution can be embedded into a tree instance of the *group Steiner tree* problem (GST). Thus, we can apply the GKR Rounding algorithm in [7] for GST on trees to round the fractional solution. However, embedding of an LP-solution to a tree instance of GST does not preserve connectivity. Also, it does not lead to a reduction from k-DST to the k edge-connected variant of GST, namely, k-GST. Hence, our algorithm is, although simple, not straightforward.

1.1 Our Results

Our main result is an $O(D \cdot k^{D-1} \cdot \log n)$-approximation algorithm for k-DST on a D-shallow instance, which includes a special case that an input graph is directed acyclic and has at most D layers.

▶ **Theorem 1.** *Consider the k edge-connected directed Steiner tree problem. Suppose an input instance has an optimal solution H^* in which, for every terminal $t \in T$, H^* has k edge-disjoint r, t-paths such that each path has length at most D. Then there exists an $O(D \cdot k^{D-1} \cdot \log n)$-approximation algorithm. In particular, there is an $O(D \cdot k^{D-1} \cdot \log n)$-approximation algorithm for k-DST on a directed acyclic graph with D layers.*

For the case $k = 1$, our algorithm yields a slightly better guarantee of $O(D \log h)$. Thus, we have as by product an LP-based approximation algorithm for DST. Applying Zelikovsky's height-reduction theorem [14, 8], this implies an LP-based quasi-polynomial-time $O(\log^3 h)$-approximation algorithm for DST. (The algorithm runs in time $O(\text{poly}(n^D)$ and has approximation ratio $O(h^{1/D} \cdot D^2 \log h)$.)

Theorem 1 also implies an algorithm of the same (asymptotic) approximation ratio for a D-shallow instance of the k edge-connected directed Steiner subgraph problem, where we wish to find a subgraph H such that the set of terminals T is k-edge-connected in H. To

see this, we invoke the algorithm in Theorem 1 as follows. Take any terminal $t^* \in T$ as a root vertex of a k-DST instance. Then we apply the algorithm for k-DST to find a subgraph H^{out} such that every terminal is k edge-connected from t^*. We apply the algorithm again to find a subgraph H^{in} such that every terminal is k edge-connected to t^*. Then the set of terminals T is k-edge connected in the graph $H^{out} \cup H^{in}$ by transitivity of edge-connectivity. The cost incurred by this algorithm is at most twice that of the algorithm in Theorem 1. Thus, we have the following theorem as a corollary of Theorem 1

▶ **Theorem 2.** *Consider the k edge-connected directed Steiner subgraph problem. Suppose an input instance has an optimal solution H^* in which, for every pair of terminals $s, t \in T$, H^* has k edge-disjoint s,t-paths such that each path has length at most D. Then there exists an $O(D \cdot k^{D-1} \cdot \log n)$-approximation algorithm.*

1.1.0.1 Overview of our algorithm

The key idea of our algorithm is to embed an LP solution for k-DST to a standard LP of GST on a tree. (We emphasize that we embed the LP solution of k-DST to that of GST not k-GST.) At first glance, a reduction from k-DST to GST on trees is unlikely to exist because any such reduction would destroy all the connectivity information. We show, however, that such tree-embedding exists, but we have to sacrifice running-time and cost to obtain such embedding.

The reduction is indeed the same as a folk-lore reduction from DST to GST on trees. That is, we list all rooted-paths (paths that start from the root vertex) of length at most D in an input graph and form a suffix tree. In the case of DST, if there is an optimal solution which is a tree of height D, then it gives an approximation preserving reduction from GST to DST which blows up the size (and thus the running time) of the instance to $O(n^D)$. Unfortunately, for the case of k-DST with $k > 1$, this reduction does not give an equivalent reduction from k-DST to k-GST on trees. The reduction is valid in one direction, i.e., any feasible solution to k-DST has a corresponding feasible solution to the tree-instance of k-GST. However, the converse is not true as a feasible solution to the tree-instance of k-GST might not give a feasible solution to k-DST. Thus, our reduction is indeed an "invalid" reduction from k-DST to a tree instance of "GST" (the case $k = 1$).

To circumvent this problem, we formulate an LP that provides a connection between an LP solution on an input k-DST instance and an LP solution of a tree-instance of GST. Thus, we can embed an LP solution to an LP-solution of GST on a (very large) tree. We then round the LP solution using the GKR Rounding algorithm for GST on trees [7]. This algorithm, again, does not give a feasible solution to k-DST as each integral solution we obtain only has "connectivity one" and thus is only feasible to DST. We cope with this issue by using a technique developed by Chalermsook et al. in [1]. Specifically, we sample a sufficiently large number of DST solutions and show that the union of all these solutions is feasible to k-DST using cut-arguments.

Each step of our algorithm and the proofs are mostly standard, but ones need to be careful in combining each step. Otherwise, the resulting graph would not be feasible to k-DST.

Organization. We provide definitions and notations in Section 2. We start our discussion by presenting a reduction from DST to GST in Section 3. Then we discuss properties of minimal solutions for k-DST in Section 4. We present standard LPs for k-DST and GST in

Section 5 and formulate a stronger LP-relaxation for k-DST in Section 6. Then we proceed to present our algorithm in Section 7. Finally, we provide some discussions in Section 8.

2 Preliminaries

We use standard graph terminologies. We refer to a directed edge (u, v), shortly, by uv (i.e., u and v are head and tail of uv, respectively), and we refer to an undirected edge by $\{u, v\}$. For a (directed or undirected) graph G, we denote by $V(G)$ and $E(G)$ the sets of vertices and edges of G, respectively. If a graph G is associated with edge-costs $\{c_e\}_{e\in E(G)}$, then we denote the cost of any subgraph $H \subseteq G$ by $\mathsf{cost}(H) = \sum_{e\in E(H)} c_e$. For any path P, we use *length* to mean the number of edges in a path P and use *cost* to mean the total costs of edges in P.

In the *directed Steiner tree* problem (DST), we are given a directed graph G with edge-costs $\{c_e\}_{e\in E(G)}$, a root vertex r and a set of terminals $T \subseteq V(G)$. The goal is to find a min-cost subgraph $H \subseteq G$ such that H has a directed path from the root r to each terminal $t \in T$. A generalization of DST is the k *edge-connected directed Steiner tree* problem (k-DST). In k-DST, we are given the same input as in DST plus an integer k. The goal is to find a min-cost subgraph H that has k edge-disjoint paths from the root r to each terminal $t \in T$. The k *edge-connected directed Steiner subgraph* problem is a variant of k-DST, where there is no root vertex, and the goal is to find a min-cost subgraph H such that the set of terminals T is k edge-connected in H.

The problems on undirected graphs that are closely related to of DST and k-DST are the *group Steiner tree* problem (GST) and the k *edge-connected group Steiner tree* problem (k-GST). In GST, we are given an undirected graph $\mathcal{G}$ with edge-costs $\{c_e\}_{e\in E(G)}$, a root vertex r and a collection of subset of vertices $\{\mathcal{T}_i\}_{i=1}^h$ called groups. The goal is to find a a min-cost subgraph $\mathcal{H}$ that connects r to each group $\mathcal{T}_i$. In k-GST, the input consists of an additional integer k, and the goal is to find a min-cost subgraph H with k edge-disjoint $r, \mathcal{T}_i$-paths for every group T_i.

Consider an instance of DST (resp., k-DST). We denote by Q the set of all paths in G that start from the root r. The set of paths in Q that end with a particular pattern, say $\sigma = (v_1, \ldots, v_q)$, is denoted by $Q(\sigma)$. This pattern σ can be a vertex v, an edge e or a path $\sigma = (v_1, \ldots, v_q)$ in G. For example, $Q(u, v, w)$ consists of paths P of the form $P = (r, \ldots, u, v, w)$. We say that a path P ends in a vertex v (resp., an edge e) if v (resp., e) is the last vertex (resp., edge) of P.

We may consider only paths with particular length, say D. We denote by Q_D the set of paths that start at r and has length at most D. The notation for Q_D is analogous to Q, e.g., $Q_D(uv) \subseteq Q_D$ is the set of paths in Q_D that end in an edge uv. A concatenation of a path p with an edge e or a vertex v are denoted by $p + e$ and $p + v$, respectively. For example, $(u_1, \ldots, u_\ell) + vw = (u_1, \ldots, u_\ell, v, w)$.

Given a subset of vertices S, the set of edges entering S is denoted by

$$\delta^-(S) = \{uv \in E : u \in S, v \notin S\}$$

The indegree of S is denoted by $\mathsf{indeg}(S) = |\delta^-(S)|$. Analogously, we use $\delta^+(S)$ and $\mathsf{outdeg}(S)$ for the set of edges leaving S. For undirected graphs, we simply use the notations $\delta(S)$ and $\mathsf{deg}(S)$.

We say that a feasible solution H to k-DST is D-*shallow* if, for every terminal $t \in T$, there exists a set of k edge-disjoint r, t-paths in H such that every path has length at most D. An instance of k-DST that has an optimal D-shallow solution is called a D-shallow instance.

We also use the term D-shallow analogously for k-GST and the k edge-connected Steiner subgraph problem.

To distinguish between the input of k-DST (which is a directed graph) and k-GST (which is an undirected graph), we use script fonts, e.g., $\mathcal{G}$, to denote the input of k-GST. Also, we use $\mathcal{Q}$ to denote the set of all paths from the root r to any vertex v in the graph $\mathcal{G}$. The cost of a set of edges F (or a graph) is defined by a function $\mathsf{cost}(F) = \sum_{e \in F} c_e$. At each point, we consider only one instance of k-DST (respectively, k-GST). So, we denote the cost of the optimal solution to k-DST by opt_{kDST} (respectively, opt_{kGST}).

3 Reduction from Directed Steiner Tree to Group Steiner Tree

In this section, we describe a reduction $\mathcal{R}$ from DST to GST. We recall that Q denotes all the r, v-paths in a DST instance G. The reduction is by simply listing paths in the directed graph G as vertices in a tree $\mathcal{G} = \mathcal{R}(G)$ and joining each path p to $p + e$ if $p + e$ is a path in G. In fact, $\mathcal{R}(G)$ is a *suffix tree* of paths in Q. To be precise,

$$V(\mathcal{G}) = \{p : p \text{ is an } r, v\text{-path in } G\}$$
$$E(\mathcal{G}) = \{\{p, p + e\} : \text{both } p \text{ and } p + e \text{ are paths in } G \text{ starting at } r\}$$

We set the cost of edges of $\mathcal{G}$ to be $c_{\{p,p+e\}} = c_e$. Since the root r has no incoming edges in G, r maps to a unique vertex $(r) \in \mathcal{G}$, and we define (r) as the root vertex of the GST instance. We will abuse r to mean both the root of DST and its corresponding vertex of GST. For each terminal $t_i \in T$, define a group of the GST instance as

$$\mathcal{T}_i := Q(t_i) = \{p \subseteq G : p \text{ is an } r, t_i\text{-path in } G\}$$

It is easy to see that the reduction $\mathcal{R}$ produces a tree, and there is a one-to-one mapping between a path in the tree $\mathcal{G} = \mathcal{R}(G)$ and a path in the original graph G. Thus, any tree in G corresponds to a subtree of $\mathcal{R}(G)$ (but not vice versa), which implies that the reduction $\mathcal{R}$ is approximation-preserving (i.e., $\mathsf{opt}_{DST} = \mathsf{opt}_{GST}$). Note, however, that the size of the instance blows up from $O(n + m)$ to $O(n^D)$, where D is the length of the longest path in G. The reduction holds for general graphs, but it is approximation-preserving only if the DST instance is D-shallow, i.e., it has an optimal solution H^* such that any r, t_i-path in H^* has length at most D, for all terminals $t_i \in T$. However, Zelikovsky [14, 8] showed that the *metric completion* of G always contains a D-shallow solution with cost at most $D|V(G)|^{1/D}$ of an optimal solution to DST. (This is now known as Zelikovsky's height reduction theorem.) Thus, we may list only paths of length at most D from the metric completion. We denote the reduction that lists only paths of length at most D by $\mathcal{R}_D$.

4 Properties of Minimal Solutions to k-DST

In this section, we provide structural lemmas which are building blocks in formulating a strong LP-relaxation for k-DST. These lemmas characterize properties of a minimal solution to k-DST.

▶ **Lemma 3.** *Let H be any minimal solution to k-DST. Then H has at most k edge-disjoint r, v-paths, for any vertex $v \in V(H)$.*

Proof. Suppose to a contrary that H has $k + 1$ edge-disjoint r, v-paths, for some vertex $v \in V(H)$. Then v must have indegree at least $k + 1$ in H. We take one of the $k - 1$ edges

entering v, namely, uv. By minimality of H, removing uv results in a graph $H' = H - uv$ that has less than k edge-disjoint r, t_i-paths for some terminal $t_i \in T$. Thus, by Menger's theorem, there must be a subset of vertices $S \subseteq V$ such that $t_i \in S$, $r \in V - S$ and $\mathsf{indeg}_{H'}(S) \leq k-1$. Observe that we must have uv in $\delta^-_H(S)$ because H is a feasible solution to k-DST, which means that $v \in S$. Since we remove only one edge uv from H, the graph H' must have k edge-disjoint r, v-paths. But, this implies that $\mathsf{indeg}_{H'}(S) \geq k$, a contradiction. ◀

▶ **Lemma 4.** *Let H be any minimal solution to k-DST. Any vertex $v \in V(H)$ has indegree exactly $\lambda(v)$, where $\lambda(v)$ is the maximum number of edge-disjoint r, v-paths in H.*

Proof. The proof follows a standard uncrossing argument. Assume a contradiction that v has indegree at least $\lambda(v)+1$ in H. By Menger's theorem, there is a subset of vertices $U \subseteq V$ such that $\mathsf{indeg}_H(U) = \lambda(v)$, $v \in U$ and $r \notin U$ that separates v from r. We assume that U is a minimum such set. Since $\mathsf{indeg}_H(v) > \lambda(v)$, there is an edge $uv \in E(H)$ that is not contained in $\delta^-_H(U)$, i.e., $u, v \in U$.

By minimality of H, removing uv results in the graph $H' = H - uv$ such that H' has less than k edge-disjoint r, t_i-paths for some terminal $t_i \in T$. Thus, by Menger's theorem, there is a subset of vertices W such that $t_i \in W$, $r \notin W$, $uv \in \delta^-_H(W)$ and $\mathsf{indeg}_H(W) = k$. (The latter is because H is a feasible solution to k-DST.)

Now we apply an uncrossing argument to U and W. By submodularity of indeg_H, we have $\mathsf{indeg}_H(U) + \mathsf{indeg}_H(W) \geq \mathsf{deg}_H(U \cap W) + \mathsf{deg}_H(U \cup W)$. Observe that $v \in U \cap W$, $t \in U \cup W$ and $r \notin S \cup S'$. So, by the edge-connectivity of v and t,

$$\mathsf{indeg}_H(U \cap W) \geq \lambda(v) \quad \text{and} \quad \mathsf{indeg}_H(U \cup W) \geq k \tag{1}$$

The sum of the left-hand side of Eq (1) is $\mathsf{indeg}_H(U) + \mathsf{indeg}_H(W) = k + \lambda(v)$. So, we conclude that $\mathsf{indeg}_H(U \cap W) = \lambda(v)$ and $\mathsf{indeg}_H(U \cup W) = k$. Consequently, we have the set $U' = U \cap W$ such that $\mathsf{indeg}_H(U') = \lambda(v)$, $v \in U'$ and $r \notin U'$ that separates v from r. Since $u \notin W$, we know that U' is strictly smaller than U. This contradicts to the minimality of U. ◀

The following is a corollary of Lemma 3 and Lemma 4

▶ **Corollary 5.** *Let H be a minimal solution to k-DST. Then any vertex $v \in V(H)$ has indegree at most k.*

The next lemma follows from Corollary 5.

▶ **Lemma 6.** *Consider any minimal solution H to k-DST (which is a simple graph). For any edge $e \in E(H)$ and $\ell \geq 2$, there are at most $k^{\ell-2}$ paths in H with length at most ℓ that start at the root r and ends in e. That is, $|Q^H_\ell(e)| \leq k^{\ell-2}$ for all $e \in E(H)$, where $Q^H_\ell(e)$ is the set of r, v-paths of length ℓ in H.*

Proof. We prove by induction. The base case $\ell = 2$ is trivial because any rooted path of length at most 2 cannot have a common edge.

Assume, inductively, that $|Q^H_{\ell-1}(e)| \leq k^{\ell-3}$ for some $\ell \geq 3$. Consider any edge $vw \in E(H)$. By Corollary 5, v has indegree at most k. Thus, there are at most k edges entering v, namely, $u_1 v, \ldots, u_d v$, where $d = \mathsf{indeg}(v)$. By the induction hypothesis, each edge is the last edge of at most $k^{\ell-3}$ paths in $Q^H_{\ell-1}$. Thus, we have at most $d \cdot k^{\ell-3} \leq k^{\ell-2}$ paths that end in uv. That is,

$$|Q^H_\ell(vw)| \leq \sum_{i=1}^{d} |Q^H_{\ell-1}(u_d v)| \leq \sum_{i=1}^{d} k^{\ell-3} = d \cdot k^{\ell-3} \leq k^{\ell-2}.$$

◀

5 Standard LPs for k-DST and GST

In this section, we describe standard LPs for k-DST and GST. Each LP consists of two sets of variables, a variable x_e on each edge e and a variable f_p^i on each path p and a terminal t_i. The variable x_e indicates whether we choose an edge e in a solution. The variable f_p^i is a flow-variable on each path and thus can be written in a compact form using a standard flow formulation.

$$\text{LP-k-DST} \begin{cases} \min & \sum_{e\in E(G)} c_e x_e & & \\ \text{s.t.} & \sum_{p\in Q(t_i):e\in E(p)} f_p^i & \le x_e & \forall e \in E(G), \forall t_i \in T \\ & \sum_{p\in Q(t_i)} f_p^i & \ge k & \forall t_i \in T \\ & x_e & \le 1 & \forall e \in E(G) \\ & x_e & \ge 0 & \forall e \in E(G) \\ & f_p^i & \ge 0 & \forall p \in Q(t_i), \forall t_i \in T. \end{cases}$$

The standard LP for GST is similar to LP-k-DST.

$$\text{LP-GST} \begin{cases} \min & \sum_{e\in E(\mathcal{G})} c_e x_e & & \\ \text{s.t.} & \sum_{v\in\mathcal{T}_i}\sum_{p\in\mathcal{Q}(v):e\in E(p)} f_p^i & \le x_e & \forall e \in E(\mathcal{G}), \forall i = 1,2,\ldots,h \\ & \sum_{v\in\mathcal{T}_i}\sum_{p\in\mathcal{Q}(v)} f_p^i & \ge 1 & \forall i = 1,2,\ldots,h \\ & x_e & \le 1 & \forall e \in E(\mathcal{G}) \\ & x_e & \ge 0 & \forall e \in E(\mathcal{G}) \\ & f_p^i & \ge 0 & \forall p \in \mathcal{Q}, \forall i = 1,2,\ldots,h \end{cases}$$

6 A Strong LP-relaxation for for k-DST

In this section, we formulate a strong LP-relaxation for k-DST that allows us to embed a fractional solution into an LP solution of LP-GST on a tree.

$$\text{LP-k-DST*} \begin{cases} \min & \sum_{e\in E} c_e x_e & & \\ \text{s.t.} & \sum_{p\in Q(t_i):e\in E(p)} f_p^i & \le x_e & \forall e \in E(G), \forall t_i \in T \\ & \sum_{p\in Q(t_i)} f_p^i & \ge k & \forall t_i \in T \\ & \sum_{p\in Q(t_i):q\subseteq p} f_p^i & \le y_q & \forall q \in Q, \forall t_i \in T \\ & & & \text{(Subflow Capacity)} \\ & \sum_{p\in Q_\ell(e)} y_p & \le \max\{1, k^{\ell-2}\}\cdot x_e & \forall e \in E, \forall \ell \ge 1 \\ & & & \text{(Aggregating } k\text{-Flow)} \\ & x_e & \le 1 & \forall e \in E(G) \\ & x_e & \ge 0 & \forall e \in E(G) \\ & f_p^i & \ge 0 & \forall p \in Q(t_i), \forall t_i \in T \\ & y_p & \ge 0 & \forall p \in Q \end{cases}$$

For D-shallow instances of k-DST, we replace Q by Q_D to restrict length of paths to be at most D. The next lemma shows that LP-k-DST* is an LP-relaxation for k-DST.

▶ **Lemma 7.** *LP-k-DST* is an LP-relaxation for k-DST. Moreover, replacing Q by Q_D gives an LP-relaxation for k-DST on D-shallow instances.*

Proof. LP-k-DST* is, in fact, obtained from LP-k-DST (which is a standard LP) by adding a new variable y_p and two constraints.

1. **Subflow-Capacity:** $\sum_{p\in Q(t_i):q\subseteq p} f_p^i \le y_q, \forall q \in Q, \forall t_i \in T.$

2. **Aggregating k-Flow:** $\sum_{p \in Q_\ell(e)} y_p \leq \max\{1, k^{\ell-2}\} \cdot x_e, \forall e \in E, \forall \ell \geq 1.$

To show that these two constraints are valid for k-DST, we take a minimal feasible (D-shallow) solution H of k-DST. We define a solution (x, f, y) to LP-k-DST as below.

$$x_e = \begin{cases} 1 & \text{if } e \in E(H) \\ 0 & \text{otherwise} \end{cases} \qquad y_p = \begin{cases} 1 & \text{if } p \subseteq H \wedge p \in Q \\ 0 & \text{otherwise} \end{cases}$$
$$f_p^i = \begin{cases} 1 & \text{if } p \subseteq H \wedge p \in Q(t_i) \\ 0 & \text{otherwise} \end{cases}$$

By construction, $f_p^i = 1$ implies that $y_p = 1$. Thus, (x, f, y) satisfies the Subflow-Capacity constraint. By minimality of H, Corollary 6 implies that even if we list all the paths of length $\ell \geq 2$ in H, at most $k^{\ell-2}$ of them end in the same edge, and we know that rooted paths of length one share no edge (given that H is a simple graph). Thus, (x, f, y) satisfies the Aggregating k-Flow constraint. Consequently, these two constraints are valid for k-DST.

On the other hand, any integral solution that is not feasible to k-DST could not satisfy the constraints of LP-k-DST* simply because LP-k-DST* contains the constraints of LP-k-DST, which is a standard LP for k-DST. Thus, LP-k-DST* is an LP-relaxation for k-DST.

The proof for the case of D-shallow instances is the same as above except that we take H as a minimal D-shallow solution and replace Q by Q_D. ◀

7 An Approximation Algorithm for k-DST

In this section, we present an approximation algorithm for k-DST on a D-shallow instance. Our algorithm is simple. We solve LP-k-DST* on an input graph G and then embed an optimal fractional solution (x, f, y) to an LP-solution $(\hat{x}, \hat{f})$ of LP-GST on the tree $\mathcal{R}(G)$. We lose a factor of $O(k^{D-2})$ in this process. As we now have a tree-embedding of an LP-solution, we can invoke the GKR Rounding algorithm [7] to round an LP-solution on the tree $\mathcal{R}(G)$. Our embedding guarantees that any edge-set of size $k-1$ in the original graph G never maps to an edge-set in the tree $\mathcal{G} = \mathcal{R}(G)$ that separates r and $\mathcal{T}_i = Q(t_i)$ in $\mathcal{G}$. So, the rounding algorithm still outputs a feasible solution to GST with constant probability even if we remove edges in the tree $\mathcal{G}$ that correspond to a subset of $k-1$ edges in G. Consequently, we only need to run the algorithm for $O(D \cdot k \cdot \log n)$ times to boost the probability of success so that, for any subset of $k-1$ edges and any terminal $t_i \in T$, we have at least one solution that contains an r, t_i-path using none of these $k-1$ edges. In other words, the union of all the solutions satisfies the connectivity requirement. Our algorithm is described in Algorithm 1.

Algorithm 1 Algorithm for k-DST

Solve LP-k-DST* and obtain an optimal solution (x, f, y).
Map (x, f, y) to a solution $(\hat{x}, \hat{f})$ to LP-GST on $\mathcal{G} = \mathcal{R}(G)$.
for $i = 1$ **to** $2Dk \log_2 n$ **do**
  Run GKR Rounding on $(\hat{x}, \hat{f})$ to get a solution $\mathcal{Z}_i$.
  Map $\mathcal{Z}_i$ back to a subgraph Z_i of G.
end for
return $H = \bigcup_i Z_i$ as a solution to k-DST.

We map a solution (x, f, y) of LP-k-DST* on G to a solution $(\hat{x}, \hat{f})$ of LP-GST on the tree $\mathcal{G} = \mathcal{R}(G)$ as below. Note that there is a one-to-one mapping between a path in G and

a path in the tree $\mathcal{G}$.

$$\begin{aligned} \hat{x}_{\{p,p+e\}} &:= y_{p+e} && \text{for all } p+e \in Q \\ \hat{f}^i_p &:= f^i_p && \text{for all } p \in Q \text{ and for all } t_i \in T \end{aligned}$$

7.1 Cost Analysis

We show that $\mathsf{cost}(\hat{x}, \hat{f}) \le k^{D-2} \cdot \mathsf{cost}(x, f, y)$.

▶ **Lemma 8.** *Consider a solution $(\hat{x}, \hat{f})$ to LP-GST, which is mapped from a solution (x, f) of LP-k-DST* when an input k-DST instance is D-shallow, for $D \ge 2$. The cost of $(\hat{x}, \hat{f})$ is at most $\mathsf{cost}(\hat{x}, \hat{f}) \le k^{D-2} \cdot \mathsf{cost}(x, f)$.*

Proof. By the constraint $\sum_{p \in Q_\ell(e)} f_p \le \max\{1, k^{\ell-2}\} \cdot x_e$, we have that

$$\begin{aligned} \mathsf{cost}(\hat{x}, \hat{f}) &= \textstyle\sum_{e' \in E(\mathcal{G})} c_{e'} \hat{x}_{e'} && = \textstyle\sum_{e \in E(G)} \sum_{\{p,p+e\} \in E(\mathcal{G})} c_e \hat{x}_{\{p,p+e\}} \\ &= \textstyle\sum_{e \in E(G)} \sum_{p+e \in Q} c_e f_{p+e} && = \textstyle\sum_{e \in E(G)} \sum_{p \in Q(e)} c_e f_p \\ &= \textstyle\sum_{e \in E(G)} \left(c_e \cdot \sum_{p \in Q(e)} f_p \right) && \le \textstyle\sum_{e \in E(G)} c_e \cdot k^{D-2} \cdot x_e \\ &= k^{D-2} \cdot \mathsf{cost}(x, f). \end{aligned}$$

◀

It can be seen from Algorithm 1 and Lemma 8 that the algorithm outputs a solution H with cost at most $O(Dk^{D-1} \log n) \cdot \mathsf{cost}(x, f)$. Thus, H is an $O(Dk^{D-1} \log n)$-approximate solution. It remains to show that H is feasible to k-DST.

7.2 Feasibility Analysis

Now we show that H is feasible to k-DST with high probability. To be formal, consider any subset $F \subseteq E(G)$ of $k-1$ edges. We map F to their corresponding edges $\mathcal{F}$ in the tree $\mathcal{G}$. Thus, $\mathcal{F} := \{\{P, P+e\} : P + e \in Q \wedge e \in F\}$.

Observe that no vertices in $\mathcal{G} - \mathcal{F}$ correspond to paths that contain an edge in F. Thus, we can define an LP solution (y^F, z^F) for LP-GST on the graph $\mathcal{G} - \mathcal{F}$ as follows.

$$y^F_e = \begin{cases} \hat{x}_e & \text{if } e \notin \mathcal{F} \\ 0 & \text{otherwise} \end{cases} \qquad z^{F,i}_p = \begin{cases} \hat{f}^i_p & \text{if } E(p) \cap \mathcal{F} = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

We show that (y^F, z^F) is feasible to LP-GST on $\mathcal{G} - \mathcal{F}$.

▶ **Lemma 9.** *For any subset of edges $F \subseteq E(G)$, define (y^F, z^F) from $(\hat{x}, \hat{f})$ as above. Then (y^F, z^F) is feasible to LP-GST on $\mathcal{G} - \mathcal{F}$.*

Proof. First, observe that $z^{F,i}_p > 0$ only if a path p contains no edges in $\mathcal{F}$. So, by construction, (y^F, z^F) satisfies $z^{F,i}_p = \hat{f}^i_p \le \hat{x}_e = y^F_e$ for all $e \in E(p)$. Hence, (y^F, z^F) satisfies the capacity constraint.

Next we show that (y^F, z^F) satisfies the connectivity constraint. Consider the solution (x, f, y) to LP-k-DST*. By the feasibility of (x, f, y) and the Max-Flow-Min-Cut theorem, the graph $G - F$ with capacities $\{x_e\}_{e \in G-F}$ can support a flow of value one from r to any terminal t_i. This implies that $\sum_{p \in Q(t_i) : E(p) \cap F = \emptyset} f^i_p \ge 1$. Consequently, we have

$$\begin{aligned} \textstyle\sum_{p \in Q(t_i) : E(p) \cap F = \emptyset} f^i_p &= \textstyle\sum_{p \in \mathcal{T}_i : E(p) \cap F = \emptyset} \sum_{p' \in \mathcal{Q}(v)} \hat{f}^i_{p'} \\ &= \textstyle\sum_{v \in \mathcal{T}_i} \sum_{p' \in \mathcal{Q}(v) : E(p') \cap \mathcal{F} = \emptyset} \hat{f}^i_{p'} \\ &= \textstyle\sum_{v \in \mathcal{T}_i} \sum_{p' \in \mathcal{Q}(v) : E(p') \cap \mathcal{F} = \emptyset} z^{F,i}_{p'} \\ &= \textstyle\sum_{v \in \mathcal{T}_i} \sum_{p' \in \mathcal{Q}(v)} z^{F,i}_{p'} \\ &\ge 1. \end{aligned}$$

All the other constraints are satisfied because (y^F, z^F) is constructed from $(\hat{x}, \hat{f})$. Thus, (y^F, z^F) is feasible to LP-GST on $\mathcal{G} - \mathcal{F}$. ◀

Lemma 9 implies that we can run the GKR Rounding algorithm on (y^F, z^F). The following is the property of GKR Rounding.

▶ **Lemma 10** ([7]). *There exists a randomized algorithm such that, given a solution $(\hat{x}, \hat{f})$ to LP-GST on a tree $\mathcal{G}$ with height D, the algorithm outputs a subgraph $\mathcal{H} \subseteq \mathcal{G}$ so that the probability that any subset of vertices $U \subseteq V(\mathcal{G})$ is connected to the root is at least*

$$\Pr[\mathcal{H} \text{ has an } r, U\text{-path.}] \geq \frac{\sum_{v \in U} \sum_{p \in \mathcal{Q}(v)} \hat{f}_p^i}{O(D)}$$

Moreover, the probability that each edge is chosen is at most $\hat{x}_e$. That is, $\mathbb{E}[\mathsf{cost}(\mathcal{H})] = \mathsf{cost}(\hat{x}, \hat{f})$. The running time of the algorithm is $O(|E(\mathcal{G})| + |V(\mathcal{G})|)$.

Since $(y^F, z^F) \leq (\hat{x}, \hat{f})$ (coordinate-wise), we can show that running GKR Rounding on $(\hat{x}, \hat{f})$ simulates the runs on (y^F, z^F) for all $F \subseteq E(G)$ with $|F| \leq k - 1$, simultaneously.

▶ **Lemma 11.** *Let $\mathcal{H}$ be a subgraph of $\mathcal{G}$ obtained by running GKR Rounding on $(\hat{x}, \hat{f})$, and let H be a subgraph of G corresponding to $\mathcal{H}$. Then, for any subset of edges $F \subseteq E(G)$ with $|F| \leq k - 1$ and for any terminal $t_i \in T$,*

$$\Pr[H - F \text{ has an } r, t_i\text{-path}] \geq \frac{1}{O(D)}.$$

Proof. Let us briefly describe the work of GKR Rounding. The algorithm marks each edge e in the tree with probability $x_e / x_{\varrho(e)}$, where $\varrho(e)$ is the parent of an edge e in $\mathcal{G}$, which is unique. Then the algorithm picks an edge e if all of its ancestors are marked. Clearly, removing any set of edges $\mathcal{F}$ from $\mathcal{G}$ only affects paths that contain an edge in $\mathcal{F}$.

Let (y^F, z^F) be defined from $(\hat{x}, \hat{f})$ as above. This LP solution is defined on a graph $\mathcal{G} - \mathcal{F}$. Thus, the probability of success is not affected by removing $\mathcal{F}$ from the graph. By Lemma 9, we can run GKR Rounding on (y^F, z^F) and obtain a subgraph $\mathcal{H}^F$ of $\mathcal{G} - \mathcal{F}$. Since $z_p^F \leq \hat{f}_p$ for all paths $p \in \mathcal{Q}$ and $z_p^F = 0$ for all $p \in \mathcal{Q} : E(p) \cap \mathcal{F} \neq \emptyset$, we have from Lemma 10 and Lemma 9 that

$$\begin{aligned}
\Pr[H - F \text{ has an } r, t_i\text{-path}] &= \Pr[\mathcal{H} - \mathcal{F} \text{ has an } r, \mathcal{T}_i\text{-path}] \\
&\geq \Pr[\mathcal{H}^F \text{ has an } r, \mathcal{T}_i\text{-path}] \\
&\geq \frac{\sum_{v \in \mathcal{T}_i} \sum_{p \in \mathcal{Q}(v)} z_p^F}{O(D)} \\
&\geq \tfrac{1}{O(D)}.
\end{aligned}$$

◀

Finally, we recall that Algorithm 1 employs GKR Rounding on $(\hat{x}, \hat{f})$ for $2Dk \log_2 n$ times. So, for any subset of $k - 1$ edges $F \subseteq E(G)$ and for any terminal $t_i \in T$, there exists one subgraph that has an r, t_i-path that contains no edge in F with large probability. In particular, the union is a feasible solution to k-DST with high probability.

▶ **Lemma 12.** *Consider the run of Algorithm 1. The solution subgraph $H = \bigcup_i Z_i$ is a feasible solution to k-DST with probability at least $1/n$.*

Proof. For $i = 1, 2, \ldots, 2Dk \log_2 n$, let Z_i be a subgraph of G obtained by running GKR Rounding on $(\hat{x}, \hat{f})$ and mapping the solution back to a subgraph of G as in Algorithm 1.

By Lemma 11, $Z_i - F$ has an r, t_i-path with probability $\Omega(1/D)$. Since each Z_i is sampled independently, we have

$$\Pr[\forall i\, Z_i - F \text{ has no } r, t_i\text{-path}] \leq \left(1 - \frac{1}{O(D)}\right)^{2Dk \log_2 n} \leq \left(\frac{1}{e}\right)^{2k \log_2 n} \leq n^{-2k}.$$

We have at most $|E(G)|^{k-1} \leq n^{2(k-1)}$ such sets F and at most $h \leq n$ terminals. So, there are at most n^{2k-1} bad events where there exists an edge-set of size $k-1$ that separates the root r and some terminal $t_i \in T$. Therefore, by union bound, $H = \bigcup_i Z_i$ is a feasible solution to k-DST with probability at least $1/n$. ◀

This completes the proof of Theorem 1. Note that, for the case of DST ($k = 1$), we only need to run GKR Rounding for $O(D \log h)$ times, thus implying an approximation ratio of $O(D \log h)$.

8 Conclusion and Discussion

We presented the first non-trivial approximation algorithm for k-DST in a special case of a D-shallow instance, which exploits the reduction from DST to GST. We hope that our techniques will shed some light in designing an approximation algorithm for k-DST in general case and perhaps lead to a bi-criteria approximation algorithm in the same manner as in [1].

One obstruction in designing an approximation algorithm in directed graphs is that there is no "true" (perhaps, probabilistic) tree-embedding for directed graphs. Both devising a tree-embedding for directed graphs and designing an approximation algorithm for k-DST with $k \geq 2$ are big open problems in the area. Another open problem, which is considered as the most challenging one by many experts, is whether there exists a polynomial-time algorithm for DST that yields a sub-polynomial approximation ratio.

Acknowledgements. Our work was inspired by the works of Rothvoß [13] and Friggstad et al. [6] and by discussions with Joseph Cheriyan and Lap Chi Lau. We also thank Zachary Friggstad for useful discussions and anonymous referees for valuable comments.

References

1 Parinya Chalermsook, Fabrizio Grandoni, and Bundit Laekhanukit. On survivable set connectivity. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 25–36, 2015. doi:10.1137/1.9781611973730.3.

2 Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *J. Algorithms*, 33(1):73–91, 1999. doi:10.1006/jagm.1999.1042.

3 Joseph Cheriyan, Bundit Laekhanukit, Guyslain Naves, and Adrian Vetta. Approximating rooted steiner networks. *ACM Transactions on Algorithms*, 11(2):8:1–8:22, 2014. doi:10.1145/2650183.

4 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

5 Moran Feldman, Guy Kortsarz, and Zeev Nutov. Improved approximation algorithms for directed steiner forest. *J. Comput. Syst. Sci.*, 78(1):279–292, 2012. doi:10.1016/j.jcss.2011.05.009.

6 Zachary Friggstad, Jochen Könemann, Young Kun-Ko, Anand Louis, Mohammad Shadravan, and Madhur Tulsiani. Linear programming hierarchies suffice for directed steiner tree. In *Integer Programming and Combinatorial Optimization – 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, pages 285–296, 2014. doi:10.1007/978-3-319-07557-0_24.

7 Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000. doi:10.1006/jagm.2000.1096.

8 Christopher S. Helvig, Gabriel Robins, and Alexander Zelikovsky. An improved approximation scheme for the group steiner problem. *Networks*, 37(1):8–20, 2001. doi:10.1002/1097-0037(200101)37:1<8::AID-NET2>3.0.CO;2-R.

9 Guy Kortsarz and Zeev Nutov. Approximating minimum-cost connectivity problems. In Teofilo F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*. Chapman and Hall/CRC, 2007. doi:10.1201/9781420010749.ch58.

10 Bundit Laekhanukit. Parameters of two-prover-one-round game and the hardness of connectivity problems. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1626–1643, 2014. doi:10.1137/1.9781611973402.118.

11 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994. doi:10.1145/185675.306789.

12 Zeev Nutov. Approximability status of survivable network problems. Preprint available at http://www.openu.ac.il/home/nutov/Survivable-Network.pdf.

13 Thomas Rothvoß. Directed steiner tree and the lasserre hierarchy. *CoRR*, abs/1111.5473, 2011. URL: http://arxiv.org/abs/1111.5473.

14 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997. doi:10.1007/BF02523690.

Tight Analysis of a Multiple-Swap Heurstic for Budgeted Red-Blue Median*

Zachary Friggstad[†1] **and Yifeng Zhang**[2]

1 **Department of Computing Science, University of Alberta, Edmonton, Canada**
zacharyf@ualberta.ca

2 **Department of Computing Science, University of Alberta, Edmonton, Canada**
yifeng2@ualberta.ca

Abstract

Budgeted Red-Blue Median is a generalization of classic k-Median in that there are two sets of facilities, say $\mathcal{R}$ and $\mathcal{B}$, that can be used to serve clients located in some metric space. The goal is to open k_r facilities in $\mathcal{R}$ and k_b facilities in $\mathcal{B}$ for some given bounds k_r, k_b and connect each client to their nearest open facility in a way that minimizes the total connection cost.

We extend work by Hajiaghayi, Khandekar, and Kortsarz [2012] and show that a *multiple-swap* local search heuristic can be used to obtain a $(5+\epsilon)$-approximation for Budgeted Red-Blue Median for any constant $\epsilon > 0$. This is an improvement over their single swap analysis and beats the previous best approximation guarantee of 8 by Swamy [2014].

We also present a matching lower bound showing that for every $p \geq 1$, there are instances of Budgeted Red-Blue Median with local optimum solutions for the p-swap heuristic whose cost is $5 + \Omega\left(\frac{1}{p}\right)$ times the optimum solution cost. Thus, our analysis is tight up to the lower order terms. In particular, for any $\epsilon > 0$ we show the single-swap heuristic admits local optima whose cost can be as bad as $7 - \epsilon$ times the optimum solution cost.

1998 ACM Subject Classification F.2.2 Computations on Discrete Structures, G.2.1 Combinatorial Algorithms, I.5.3 Clustering

Keywords and phrases Approximation Algorithms, Local search, Red-Blue Meidan

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.75

1 Introduction

Facility location problems crop up in many areas of computing science and operations research. A typical problem involves a set of clients and possible facility locations located in a metric space. The goal is to open some facilities and connect each client to some open facility as cheaply as possible. These problems become difficult when there are costs associated with opening facilities or additional constraints that ensure we cannot open too many facilities.

We study Budgeted Red-Blue Median, one particular instance of this type of problem. Here we are given a set of clients C, a set of *red* facilities $\mathcal{R}$, and a set of *blue* facilities $\mathcal{B}$. These are located in some metric space with metric distances $d(i,j) \geq 0$ for any two $i, j \in C \cup \mathcal{R} \cup \mathcal{B}$. Additionally, we are given two integer bounds $k_r \leq |\mathcal{R}|$ and $k_b \leq |\mathcal{B}|$. The

* The full version of this work is available at http://arxiv.org/abs/1603.00973.

† This research was undertaken, in part, thanks to funding from the Canada Research Chairs program and an NSERC Discovery Grant.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 75; pp. 75:1–75:13

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

goal is to select/open k_r red facilities R and k_b blue facilities B to minimize

$$\text{cost}(R \cup B) := \sum_{j \in C} \min_{i \in R \cup B} d(i,j)\,.$$

The classic NP-hard k-Median problem appears as a special case when, say, $\mathcal{R} = \emptyset$. Thus, Budgeted Red-Blue Median is NP-hard. In this paper, we focus on approximation algorithms for Budgeted Red-Blue Median, in particular on local search techniques.

1.1 Previous Work

The study of Budgeted Red-Blue Median from the perspective of approximation algorithms was initiated by Hajiaghayi, Khandekar, and Kortsarz [9], where they obtain a constant-factor approximation by a local search algorithm that iteratively tries to swap one red and/or one blue facility in the given solution. The do not specify the constant in their analysis, but it looks to be greater than 8. Citing [9] as inspiration, Krishnaswamy et al. studied a generalization of Budgeted Red-Blue Median known as Matroid Median [10]. Here, a matroid structure is given over the set of facilities and we can only open a set of facilities if they form an independent set in the matroid. They obtain a constant-factor approximation for Matroid Median through rounding an LP relaxation. This was later refined to an 8-approximation by Swamy [15].

The special case of k-Median is a classic optimization problem and has received a lot of attention from both theoretical and practical communities. The best approximation guarantee known so far is 2.675 by Byrka et al. [5], who build heavily on the breakthrough work of Li and Svensson for the problem [11].

While local search techniques have been used somewhat infrequently in the design of approximation algorithms in general, it may be fair to say that they have seen the most success in facility location problems. For almost 10 years, the best approximation for k-Median was based on a local search algorithm. Arya et al. [3] show that a multiple-swap heuristic leads to a $(3+\epsilon)$-approximation for k-Median for any constant $\epsilon > 0$. This analysis was simplified in [8], which inspires much of our analysis.

Another textbook application of local search is a $(1+\sqrt{2})$-approximation for Uncapacitated Facility Location [3, 6]. Local search has been very helpful in approximating Capacitated Facility Location, the first constant-factor approximation was by Pál, Tardos, and Wexler [13] and the current best approximation is a $(5+\epsilon)$-approximation by Bansal, Garg, and Gupta [4]. In the special case when all capacities are uniform, Aggarwal et al. [1] obtain a 3-approximation. Even more examples of local search applied to other facility location variants can be found in [2, 7, 8, 12, 14].

1.2 Our Results and Techniques

We show that a multiswap generalization of the local search algorithm considered in [9] is a $(5+\epsilon)$-approximation for Budgeted Red-Blue Median. That is, for a value p say the p-swap heuristic is the algorithm that, upon given an initial feasible solution, tries to swap up to p facilities of each colour. If no such swap produces a cheaper solution, it terminates. Otherwise, it iterates with the now cheaper solution. Algorithm 1 in Section 2 gives the formal description of our algorithm.

Say that a solution is locally optimum for the p-swap heuristic if no cheaper solution can be found by swapping up to p facilities of each colour. Let OPT denote the cost of an optimum solution. Our main result is the following.

▶ **Theorem 1.** *Any locally optimum solution for the p-swap heuristic has cost at most* $(5 + O(1/\sqrt{p})) \cdot OPT$.

Using standard techniques (briefly mentioned in Section 2), this readily leads to a polynomoial-time approximation algorithm. By choosing $p = \theta(1/\epsilon^2)$ we have the following.

▶ **Theorem 2.** *For any constant* $\epsilon > 0$, Budgeted Red-Blue Median *admits a polynomial-time* $(5 + \epsilon)$*-approximation.*

This improves over the 8-approximation for Budgeted Red-Blue Median in [15].

We emphasize the approximation guarantee from Theorem 1 result is for Budgeted Red-Blue Median only, the 8-approximation in [15] is still the best approximation for the general Matroid Median problem. Indeed, [10] show that Matroid Median cannot be approximated within any constant factor using any constant number of swaps even in the generalization of Budgeted Red-Blue Median where there can be a super-constant number facility colours.

We also present a lower bound that matches our analysis up the lower order terms.

▶ **Theorem 3.** *For any integers* p, ℓ *with* $1 \leq p \leq \ell/2$*, there is an instance of* Budgeted Red-Blue Median *that has a locally-optimum solution for the p-swap heuristic with cost at least* $\left(5 + \frac{2}{p} - \frac{10p}{\ell+1}\right) \cdot OPT$.

By letting $\ell \to \infty$ but keeping p fixed, we see that the p-swap heuristic cannot guarantee a ratio better than $5 + \frac{2}{p}$. So, Theorem 1 is tight up to lower order terms. Also, for $p = 1$ we see that the single-swap heuristic analyzed in [9] is not better than a 7-approximation.

Local search techniques are typically analyzed by constructing a set of candidate *test swaps* where some facilities in the optimum solution are swapped in and some from the local optimum are swapped out in order to generate a useful inequality. One of the main features of the k-Median analysis in [3] and [8] is that such swaps can be considered that ensure each facility in the global optimum is swapped in once and, by averaging some swaps, each facility in the local optimum is swapped out to the extent of at most $1 + O(\epsilon)$ times. Each time a facility in the local optimum is swapped out, they pay an additional 2 times the global optimum cost for some clients to reassign them.

We obtain only a $5 + \epsilon$ approximation because we end up swapping out some facilities in the local optimum solution to the extent of $2 + O(\epsilon)$, thereby paying an additional $2 + O(\epsilon)$ more than in the k-Median analysis. Ultimately, this is because some of our initial swaps generate inequalities that depend *positively* on client assignment costs in the local optimum. So we consider additional swaps that do not introduce any more positive dependence on the local optimum to cancel them out.

This issue was also encountered in the analysis in [9]. In some sense, we are showing that this is the only added difficulty over the standard k-Median analysis. However, the averaging arguments we use are a bit more sophisticated than the analysis for k-Median.

1.3 Organization

Section 2 presents the algorithm and describes some useful notation. In particular, it presents a way to decompose the global and local optimum solution into structured groups that are examined in the analysis. Section 3 analyzes the quality of locally optimum solutions to prove Theorem 1. Section 4 proves Theorem 3 with an explicit construction of a bad example. We conclude with some remarks in Section 5.

2 Notation and Preliminaries

Say that a *feasible solution* is a pair (R, B) of subsets $R \subseteq \mathcal{R}$ and $B \subseteq \mathcal{B}$ with $|R| = k_r$ and $|B| = k_b$. Algorithm 1 describes the local search algorithm.

Algorithm 1 The p-Swap Heuristic for Budgeted Red-Blue Median

Let (R, B) be an arbitrary feasible solution.
while there is some feasible solution (R', B') with $|R - R'| \leq p$
and $|B - B'| \leq p$ and $\text{cost}(R' \cup B') < \text{cost}(R \cup B)$ **do**
$(R, B) \leftarrow (R', B')$
end while
return (R, B)

While a single iteration of Algorithm 1 can be executed in $n^{O(p)}$ (where n is the total number of locations in the problem), it may be that the number of iterations is not polynomially bounded. We can employ a well-known trick to ensure it does terminate in a polynomial number of steps while losing only another ϵ in our analysis. The idea is to perform the update only if $\text{cost}(R' \cup B') \leq (1 - \epsilon/\Delta) \cdot \text{cost}(R \cup B)$ where Δ is some quantity that is polynomial in the input size. Our analysis is compatible with this approach; one can check that the total weight of all inequalities we consider is polynomially bounded. For example, see [3] for details. We do not focus any further on this issue, and instead work toward analyzing the cost of the solutions produced by Algorithm 1 as it is stated.

From now on, let $S = R \cup B$ with $R \subseteq \mathcal{R}, B \subseteq \mathcal{B}$ denote an arbitrary local optimum solution. That is, there is no cheaper solution (R', B') with $|R - R'| \leq p$ and $|B - B'| \leq p$. Also fix a global optimum solution $O = R^* \cup B^*$ where $R^* \subseteq \mathcal{B}$ and $B^* \subseteq \mathcal{B}$. We assume that $S \cap O = \emptyset$. This is without loss of generality, as we can duplicate each facility location in the input and say that S use the first copies and O use the second copies. It is easy to verify that S is still a local optimum solution.

To help analyze the cost, we will introduce some notation. For any client $j \in \mathcal{C}$, let $s_j \in S$ denote the local optimum facility is closest to j and $o_j \in O$ denote the global optimum facility that is closest to j, breaking ties arbitrarily. For brevity, let $c_j = d(j, s_j)$ be the cost of assigning j in the local optimum and $c^*_j = d(j, o_j)$ the cost of assigning j in the global optimum. Thus, $\text{cost}(S) = \sum_{j \in C} c_j$ and $\text{cost}(O) = \sum_{j \in C} c^*_j$. For any facility $i^* \in O$ we let $N^*(i^*) = \{j \in C : o_j = i^*\}$ and for any $i \in S$ we let $N(i) = \{j \in C : s_j = i\}$.

Let $\phi : O \to S$ map each facility in O to its nearest facility in S, breaking ties arbitrarily. For $i \in S$, let $\deg(i) = |\phi^{-1}(i)|$. If $\deg(i) \neq 0$, let $\text{cent}(i)$ be the facility in $\phi^{-1}(i)$ that is closest to i, again breaking ties arbitrarily.

We also borrow some additional notation from [9].

▶ **Definition 4** (very good, good, bad facility)**.** A facility $i \in S$ is *very good* if $\deg(i) = 0$, *good* if no $i^* \in \phi^{-1}(i)$ has the same colour as i, and *bad* otherwise.

The analysis in [9] divides $S \cup O$ into *blocks* that satisfy certain properties. We require slightly stronger properties than their blocks guarantee. We also use a slightly different notion of what it means for some $i \in S$ to be a *leader*. The required properties are summarized in the following lemma, which also serves as our definition of a block. The proof can be found in the full version of our work.

▶ **Lemma 5.** *We can partition $S \cup O$ into* blocks T *satisfying the following properties.*
- $|T \cap R| = |T \cap R^*|$ *and* $|T \cap B| = |T \cap B^*|$.

- *For every $i \in S \cap T$, we also have $\phi^{-1}(i) \subseteq T$. For every $i^* \in O \cap T$, we have $\phi(i^*) \in T$.*
- *There is some facility $\hat{i} \in T \cap S$ with $\deg(\hat{i}) > 0$ designated as the* leader *that has the following properties. Every other $i \in T \cap S - \{\hat{i}\}$ is either good or very good and all good $i \in T \cap S - \{\hat{i}\}$ have the same colour.*

We will focus on analyzing one block at a time to prove the approximation guarantee. This provides us with a cleaner way to describe the test swaps and the additional structure will help handle the inevitable cases where we have to swap out some $i \in S$ but cannot swap in all of $\phi^{-1}(i)$. For example, this can happen if all blue facilities $i \in B$ have $\deg(i)$ being very large (so all $\deg(i') = 0$ facilities are red). We will still need to close some of them in order to open facilities in B^* when generating bounds via test swaps.

Before delving into the analysis we note the following two bounds. The first has been used extensively in local search analysis and was first proven in [3] and the second was proven in [9].

▶ **Lemma 6.** *For any $j \in C$, $d(j, \phi(o_j)) - c_j \leq 2c_j^*$.*

▶ **Lemma 7.** *For any $j \in C$, $d(j, \text{cent}(\phi(o_j))) - c_j \leq 3c_j^* + c_j$.*

Finally, we often consider operations that add or remove a single item from a set (usually to exclude the leader $\hat{i}$ of a block from some parts of the analysis). To keep the notation cleaner we let $A + i$ and $A - i$ refer to $A \cup \{i\}$ and $A - \{i\}$, respectively, for sets of facilities A and a single facility i.

3 Multiswap Analysis

Recall that we are assuming $S = R \cup B$ is a locally optimum solution with respect to the heuristic that swaps at most p facilities of each colour and that $O = R^* \cup B^*$ is some globally optimum solution. We assume $p = t^2 + 1$ for some sufficiently large integer t.

Focus on a single block T. For brevity, let $T_R^* = T \cap R^*$ and $T_B^* = T \cap B^*$ denote the red and blue facilities from the optimum solution in T. Similarly let $T_R = T \cap R$ and $T_B = T \cap B$ denote the red and blue facilities from the local optimum solution in T. The main goal of this section is to demonstrate the following inequality for group T.

▶ **Theorem 8.** *For some absolute constant γ that is independent of t, we have*

$$0 \leq \sum_{j \in N^*(T_R^* \cup T_B^*)} \left[\left(1 + \frac{\gamma}{t}\right) c_j^* - c_j \right] + \sum_{j \in N(T_R \cup T_B)} \left[\left(4 + \frac{\gamma}{t}\right) \cdot c_j^* + \frac{\gamma}{t} \cdot c_j \right].$$

Theorem 1 follows by summing over all associated inequalities for the various blocks.

The analysis breaks into a number of cases based on whether T_R^* and/or T_B^* are large. In each of the cases, we use the following notation and assumptions. Let $\hat{i}$ denote the leader in T. Without loss of generality, assume all other $i \in T_B \cup T_R$ with $\deg(i) > 0$ are blue facilities. Let $\overline{B} = \{i \in T_B - \hat{i} : \deg(i) > 0\}$, so $\text{cent}(\overline{B})$ denotes $\{i^* \in T_B^* \cup T_R^* : \text{cent}(\phi(i^*)) = i^*\}$. Figure 1 illustrates this notation.

The swaps we consider in these cases are quite varied, but we always ensure we swap in $\text{cent}(i)$ whenever some $i \in S \cap T$ with $\deg(i) > 0$ is swapped out. This way, we can always bound the reassignment cost of each client j by using either Lemma 6 or Lemma 7.

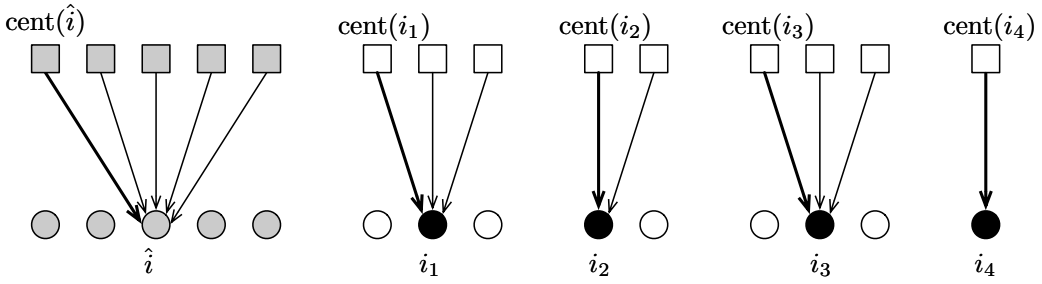


Figure 1 Illustration of a block T. The facilities on the top are in $T \cap O$ and the facilities on the bottom are in $T \cap S$. The directed edges depict ϕ, and the thick edges connect cent(i) to i. The facilities coloured black lie in $\mathcal{B}$, the facilities coloured white lie in $\mathcal{R}$, and the facilities coloured grey could either lie in $\mathcal{B}$ or $\mathcal{R}$. Note that $\overline{B} = \{i_1, i_2, i_3, i_4\}$ and cent($\overline{B}$) = {cent(i_1), cent(i_2), cent(i_3), cent(i_4)}. The layout of the figure is suggestive of how the block was constructed by adding "good" groups to the initial bad group in the procedure of generating blocks. The details of this procedure can be found in the full version of our work.

3.1 Case $|T_R^*| \leq t^2, |T_B^*| \leq t$

In this case, we simply swap out all of $T_R \cup T_B$ and swap in all of $T_R^* \cup T_B^*$. Because $R \cup B$ is a locally optimum solution and because this swaps at most t^2 facilities of each colour, we have:

$$0 \leq \text{cost}(S \cup (T_R^* \cup T_B^*) - (T_R \cup T_B)) - \text{cost}(S)\,.$$

Of course, after the swap each client will move to its nearest open facility. As is typical in local search analysis, we explicitly describe a (possibly suboptimal) reassignment of clients to facilities to upper bound this cost change.

Each $j \in N^*(T_R^* \cup T_B^*)$ is moved from s_j to o_j which incurs an assignment cost change of exactly $c_j^* - c_j$. Each $j \in N(T_R \cup T_B) - N^*(T_R^* \cup T_B^*)$ is moved to $\phi(o_j)$. Note that $\phi(o_j) \notin T$ so it remains open after the swap. By Lemma 6, the assignment cost change is bounded by $2c_j^*$. Every other client j that has not already been reassigned remains at s_j and incurs no assignment cost change. Thus,

$$0 \leq \sum_{j \in N^*(T_R^* \cup T_B^*)} (c_j^* - c_j) + \sum_{j \in N(T_R \cup T_B)} 2c_j^*$$

which is even better than what we are required to show for Theorem 8.

We note that the analysis Section 3.4 could be extended to subsume this analysis (with a worse constant), but we have included it here anyway to provide a gentle introduction to some of the simpler aspects of our approach.

3.2 Case $|T_R^*| \geq t^2 + 1, |T_B^*| \geq t + 1$

We start by briefly discussing some challenges in this case. In the worst case, all of the $i_b \in T_B$ have deg(i) being very large. The issue here is that we need to swap in each $i_b^* \in T_B^*$ in order to generate terms of the form $c_j^* - c_j$ for j with $o_j = i_b^*$. But this requires us to swap out some i_b. Since we cannot swap in all of $\phi^{-1}(i_b)$, we resort to only swapping in cent(i_b).

Any client j with s_j being closed and $o_j \in \phi^{-1}(i_b) - \text{cent}(i_b)$ cannot be reassigned to $\phi(o_j)$, so we send it to cent($\phi(o_j)$) and use Lemma 7 to bound the reassignment cost. This

leaves a term of the form $+c_j$, so we have to consider additional swaps involving $-c_j$ in their bound to cancel this out. These additional swaps cause us to lose a factor of roughly 5 instead of 3.

Another smaller challenge is that we do not want to swap out the leader $\hat{i} \in T \cap S$ for a variety of technical reasons. However, since $|T_R^*|$ and $|T_B^*|$ are both big, this is not much of a problem. When we swap in some $i^* \in T \cap O$, we will just swap out a randomly chosen facility in $T \cap S - \hat{i}$ of the same colour. The probability any particular facility is swapped in this way is very small. Ultimately, each facility in $T \cap S - \hat{i}$ will be swapped out at most $2 + O(1/t)$ times in expectation.

To be precise, we partition the set of clients in $N(T_R \cup T_B)$ into two groups:

$$C_{bad} := N(\overline{B}) \cap N^*(T_R^* - \text{cent}(\overline{B})) \quad \text{and} \quad C_{ok} := N(T_R \cup T_B - \hat{i}) - C_{bad}.$$

We have omitted $N(\hat{i})$ from C_{ok} because we will not close $\hat{i}$.

The first group is dubbed *bad* because there may be a swap where both s_j and $\phi(o_j)$ are closed yet o_j is not opened so we can only use Lemma 7 to bound their reassignment cost. In fact, some clients $j \in C_{good}$ may also be involved in such a swap, but we are able to use an averaging argument for these clients to show that the resulting $+s_j$ term from using Lemma 7 appears with negligible weight and does not need to be cancelled.

We consider the following two types of swaps to generate our initial inequality.

- For each $i_b^* \in T_B^*$, choose a random $i_b \in T_B - \hat{i}$. If $i_b \notin \overline{B}$ (i.e. $\deg(i_b) = 0$) then simply swap out i_b and swap in i_b^*. If $i_b \in \overline{B}$ then swap out i_b and a random $i_r \in T_R - \hat{i}$ and swap in i_b^* and $\text{cent}(i_b)$.
- For each $i_r^* \in T_R^* - \text{cent}(\overline{B})$, swap in i_r^* and swap out a randomly chosen $i_r \in T_R - \hat{i}$.

By choosing facilities at "random", we mean uniformly at random from the given set and this should be done independently for each invokation of the swap.

▶ **Lemma 9.**

$$0 \le \sum_{j \in N^*(T_B^* \cup T_R^*)} \left(\frac{t+1}{t} \cdot c_j^* - c_j \right) + \sum_{j \in C_{ok}} \left[\left(2 + \frac{5}{t} \right) c_j^* + \frac{1}{t} c_j \right] + \frac{t+1}{t} \sum_{j \in C_{bad}} (3c_j^* + c_j) \,.$$

Proof. For brevity, we will let $\beta_R = \frac{|T_R|}{|T_R - \hat{i}|}$ and $\beta_B = \frac{|T_B|}{|T_B - \hat{i}|}$. Note that $\beta_R, \beta_B \le \frac{t+1}{t}$ and that either $\beta_R = 1$ or $\beta_B = 1$.

First consider a swap of the first type that swaps in $\{i_b^*, \text{cent}(i_b)\}$ and swaps out $\{i_b, i_r\}$ for some i_b with $\deg(i_b) > 0$. Because $R \cup B$ is a local optimum the cost of the solution does not decrease after performing this swap. We provide an upper bound on the reassignment cost.

Each $j \in N^*(\{i_b^*, \text{cent}(i_b)\})$ is reassigned from s_j to o_j and incurs an assignment cost change of $c_j^* - c_j$. Every client $j \in N(\{i_b, i_r\})$ that has not yet been reassigned is first moved to $\phi(o_j)$. If this $\phi(o_j)$ remains open, assign j to it. By Lemma 6, the assignment cost for j increases by at most $2c_j^*$. If $\phi(o_j)$ is not open then $\phi(o_j) = i_b$ (because $\deg(i_r) = 0$) so we instead move j to $\text{cent}(\phi(o_j)) = \text{cent}(i_b)$. Lemma 7 shows the assignment cost increases by at most $3c_j^* + c_j$. This can only happen if $s_j \in \{i_r, i_b\}$ and $\phi(o_j) = i_b$.

Combining these observations and using slight overestimates, we see

$$0 \le \sum_{j \in N^*(\{i_b^*, \text{cent}(i_b)\})} (c_j^* - c_j) + \sum_{\substack{j \in N(\{i_b, i_r\}) \\ \phi(o_j) \ne i_b}} 2c_j^* + \sum_{\substack{j \in N(\{i_b, i_r\}) \\ \phi(o_j) = i_b}} (3c_j^* + c_j). \tag{1}$$

Now, if the random choice for i_b in the swap has $\deg(i_b) = 0$, then swapping $\{i_b\}$ out and $\{i_b^*\}$ in generates an even simpler inequality:

$$0 \leq \sum_{j \in N^*(i_b^*)} (c_j^* - c_j) + \sum_{j \in N(i_b)} 2c_j^*. \tag{2}$$

To see this, just reassign each $j \in N^*(i_b^*)$ from s_j to o_j and reassign the remaining $j \in N(i_b)$ from s_j to $\phi(o_j)$ (which remains open because $\deg(i_b) = 0$) and use Lemma 6.

Consider the expected inequality that is generated for this fixed i_b^*. We start with some useful facts that follow straight from the definitions and the swap we just performed.

- Any $j \in N^*(\text{cent}(\overline{B}))$ has o_j being opened with probability $\frac{1}{|T_B - \hat{i}|}$.
- Any $j \in C_{bad}$ has s_j being closed with probability $\frac{1}{|T_B - \hat{i}|}$.
- Any $j \in C_{ok} - N(T_R)$ has s_j being closed with probability $\frac{1}{|T_B - \hat{i}|}$. When this happens, if o_j is not opened then $\phi(o_j)$ must be open. That is, $s_j \in C_{ok}$ means $o_j \in T_B^* \cup \text{cent}(\overline{B})$.

  Furthermore, if $o_j \in T_B^*$ then $\phi(o_j) = \hat{i}$ (by the structure of block T) which remains open. If $o_j \in \text{cent}(\overline{B})$ then either $\phi(o_j)$ was not closed, or else $\text{cent}(\phi(o_j)) = o_j$ was opened.
- Any $j \in C_{ok} \cap N(T_R)$ has s_j being closed with probability $\frac{|\overline{B}|}{|T_B - \hat{i}|} \cdot \frac{1}{|T_R - \hat{i}|}$. If o_j and $\phi(o_j)$ are closed, then we move j to $\text{cent}(\phi(o_j))$. However, this can only happen with probability $\frac{1}{|T_B - \hat{i}|} \cdot \frac{1}{|T_R - \hat{i}|}$ since it must be that $\phi(o_j)$ is the blue facility that was randomly chosen to be closed.

Averaging (1) over all random choices and using some slight overestimates we see

$$\begin{aligned} 0 \quad \leq \quad & \sum_{j \in N^*(i_b^*)} (c_j^* - c_j) + \frac{1}{|T_B - \hat{i}|} \cdot \sum_{j \in N^*(\text{cent}(\overline{B}))} (c_j^* - c_j) \\ & + \frac{1}{|T_B - \hat{i}|} \left[\sum_{j \in C_{bad}} (3c_j^* + c_j) + \sum_{j \in C_{ok} - N(T_R)} 2c_j^* \right] \\ & + \frac{1}{|T_B - \hat{i}|} \cdot \frac{1}{|T_R - \hat{i}|} \sum_{j \in C_{ok} \cap N(T_R)} (|\overline{B}| 2c_j^* + 3c_j^* + c_j). \end{aligned}$$

Summing over all $i_b^* \in T_B^*$ (i.e. over all swaps of the first type) shows

$$\begin{aligned} 0 \quad \leq \quad & \sum_{j \in N^*(T_B^*)} (c_j^* - c_j) + \beta_B \cdot \sum_{j \in N^*(\text{cent}(\overline{B}))} (c_j^* - c_j) \\ & + \beta_B \cdot \left[\sum_{j \in C_{bad}} (3c_j^* + c_j) + \sum_{j \in C_{ok} - N(T_R)} 2c_j^* \right] \\ & + \frac{\beta_B}{|T_R - \hat{i}|} \cdot \sum_{j \in C_{ok} \cap N(T_R)} ((2|\overline{B}| + 3)c_j^* + c_j). \end{aligned} \tag{3}$$

Next, consider the second type of swap that swaps in some $i_r^* \in T_R^* - \text{cent}(\overline{B})$ and swaps out some randomly chosen $i_r \in T_R - \hat{i}$. Over all such swaps, the expected number of times each $i_r \in T_R - \hat{i}$ is swapped out is $\frac{|T_R^*| - |\overline{B}|}{|T_R - \hat{i}|} = \beta_R - \frac{|\overline{B}|}{|T_R - \hat{i}|}$. In each such swap, we reassign $j \in N^*(i_r^*)$ from s_j to o_j and every other $j \in N(i_r)$ from j to $\phi(o_j)$ which is still open because $\deg(i_r) = 0$. Thus,

$$0 \leq \sum_{j \in N^*(T_R^* - \text{cent}(\overline{B}))} (c_j^* - c_j) + \left(\beta_R - \frac{|\overline{B}|}{|T_R - \hat{i}|} \right) \cdot \sum_{j \in C_{ok} \cap N(T_R)} 2c_j^* \,.$$

Scaling this bound by β_B, adding it to (3), and recalling $|T_R| \geq t^2$ shows

$$\begin{aligned} 0 \quad \leq \quad & \sum_{j \in N^*(T_B^*)} (c_j^* - c_j) + \beta_B \cdot \sum_{j \in N^*(T_R^*)} (c_j^* - c_j) \\ & + \beta_B \cdot \left[\sum_{j \in C_{bad}} (3c_j^* + c_j) + \sum_{j \in C_{ok} - N(T_R)} 2c_j^* + \right] \\ & + \beta_B \cdot \sum_{j \in C_{ok} \cap N(T_R)} \left[\left(2\beta_R + \frac{3}{t^2} \right) \cdot c_j^* + \frac{1}{t^2} \cdot c_j \right]. \end{aligned}$$

Recall that $\beta_B, \beta_R \leq \frac{t+1}{t}$ and also $\beta_B \cdot \beta_R \leq \frac{t+1}{t}$ to complete the proof of Lemma 9. ◀

Our next step is to cancel terms of the form $+c_j$ in the bound from Lemma 9 for $j \in C_{bad}$. To do this, we again perform the second type of swap for each $i \in T_R^* - \text{cent}(\overline{B})$ but reassign clients a bit differently in the analysis.

▶ **Lemma 10.**

$$0 \leq \sum_{j \in C_{bad}} (c_j^* - c_j) + \frac{t+1}{t} \cdot \sum_{j \in C_{ok} \cap N(T_R)} 2c_j^* .$$

Proof. For each $i_r^* \in T_R^* - \text{cent}(\overline{B})$, swap i_r^* in and a randomly chosen $i_r \in T_r - \hat{i}$. Rather than reassigning all $j \in N^*(i_r^*)$ to i_r^*, we only reassign those in $C_{bad} \cap N^*(i_r^*)$. Since $\deg(i_r) = 0$ then any other $j \in N(i_r)$ can be reassigned to $\phi(o_j)$ and which increases the cost by at most $2c_j^*$.

Summing over all i_r^*, observing that $C_{bad} \subseteq T_R^* - \text{cent}(\overline{B})$, and also observing that each $j \in C_{ok}$ has s_j closed at most $\beta_R \leq \frac{t+1}{t}$ times in expectation, we derive the inequality stated in Lemma 10. ◀

Adding the bounds stated in Lemmas 9 and 10 shows that Theorem 8 holds in this case.

3.3 Case $|T_R^*| \geq t^2 + 1, |T_B^*| \leq t$

In this case, we start by swapping in all of T_B^* and swapping out all of T_B (including, perhaps, $\hat{i}$ if it is blue). In the same swap, we also swap in $\text{cent}(T_B)$ and swap out a random subset of the appropriate number of facilities in $T_R - \hat{i}$. This is possible as $|T_R - \hat{i}| \geq t \geq |\text{cent}(T_B)|$. By random subset, we mean among all subsets of $T_r - \hat{i}$ of the necessary size, choose one uniformly at random.

As with Section 3.2, we begin with a definition of bad clients that is specific to this case:

$$C_{bad} := N(T_B) \cap N^*(T_R^* - \text{cent}(T_B)) .$$

Clients $j \in C_{bad}$ have both s_j and $\phi(o_j)$ being closed yet o_j is not opened and we cannot make this negligible with an averaging argument.

▶ **Lemma 11.**

$$0 \quad \leq \quad \sum_{j \in N^*(T_B^* \cup \text{cent}(T_B))} (c_j^* - c_j) + \frac{1}{t} \sum_{j \in N(T_R)} (3c_j^* + c_j) + \sum_{j \in C_{bad}} (3c_j^* + c_j)$$

Proof. After the swap, reassign every $j \in N^*(T_B^* \cup \text{cent}(T_B))$ from s_j to o_j, for a cost change of $c_j^* - c_j$. Every other j that has s_j being closed is first reassigned to $\phi(o_j)$. If this is not open, then further move j to $\text{cent}(o_j)$ which must be open because the only facilities $i \in T_R \cup T_B$ with $\deg(i) > 0$ that were closed lie in T_B and we opened $\text{cent}(T_B)$.

If $j \in N(T_B) - C_{bad}$ then $o_j \in T_B^* \cup \text{cent}(T_B)$ and we have already assigned j to o_j. If $j \in C_{bad}$ then we have moved j to $\text{cent}(\phi(o_j))$ and the cost change is $3c_j^* + c_j$ by Lemma 7.

Finally, if $j \in N(T_R)$ then we either move j to $\phi(o_j)$ or to $\text{cent}(\phi(o_j))$ if $\phi(o_j)$ is not open. The worst-case bound on the reassignment cost is $3c_j^* + c_j$ by Lemmas 6 and 7. However, note that $s_j \in T_R$ is closed with probability at most $1/t$ because we closed a random subset of $T_r - \hat{i}$ of size at most t and $|T_r - \hat{i}| \geq t^2$. ◀

We still need to swap in $T_R^* - \text{cent}(T_B)$. For each such facility i_r^*, swap in i_r^* and swap out a randomly chosen $i_r \in T_R - \hat{i}$. The analysis of these swaps is nearly identical to analysis of the second type of swaps in Section 3.2, so we omit it and merely summarize what we get by combining the resulting inequalities with the inequality from Lemma 11.

▶ **Lemma 12.**

$$0 \leq \sum_{j \in N^*(T_R^* \cup T_B^*)} (c_j^* - c_j) + \sum_{j \in N(T_R)} \left(\frac{t^2+1}{t^2} \cdot 2c_j^* + \frac{1}{t} \cdot (3c_j^* + c_j) \right) + \sum_{j \in C_{bad}} (3c_j^* + c_j) \,.$$

We cancel the $+c_j$ terms for $j \in C_{bad}$ with one further collection of swaps. For each $i_r^* \in T_R^* - \text{cent}(T_B)$ we swap in i_r^* and a randomly chosen $i_r \in T_R - \hat{i}$. The following lemma summarizes a bound we can obtain from these swaps. It is proven in essentially the same way as Lemma 10.

▶ **Lemma 13.**

$$0 \leq \sum_{j \in C_{bad}} (c_j^* - c_j) + \frac{t^2+1}{t^2} \cdot \sum_{j \in N(T_R)} 2c_j^* \,.$$

Adding this to the bound from Lemma 12 and recalling $C_{bad} \subseteq N(T_B)$ shows

$$0 \leq \sum_{j \in N^*(T_R^* \cup T_B^*)} (c_j^* - c_j) + \sum_{j \in N(T_R \cup T_B)} \left(\frac{t^2+3t+1}{t^2} \cdot 4c_j^* + \frac{1}{t} \cdot c_j \right) .$$

3.4 Case $|T_R^*| \leq t^2, |T_B^*| \geq t+1$

Because $\phi^{-1}(i) \subseteq T_R^*$ and $\deg(i) > 0$ for each $i \in \overline{B}$, then $|\overline{B}| \leq t^2$ as well. We will swap all of T_R^* for all of T_R, but we will also swap some blue facilities at the same time. Let $B' = \overline{B}$ and let $\overline{B}'$ be an arbitrary subset of T_B^* of size $|\overline{B}|$.

If $\hat{i} \notin T_R \cup B'$ then add $\hat{i}$ to B'. If $\text{cent}(\hat{i}) \notin T_R^* \cup \overline{B}'$ then add $\text{cent}(\hat{i})$ to $\overline{B}'$. At this point, $\left| |\overline{B}'| - |B'| \right| \leq 1$ Add an arbitrary $i_b^* \in T_B^* - \overline{B}'$ to $\overline{B}'$ or $i_b \in T_B - B'$ to B' to ensure $|\overline{B}'| = |B'|$.

We begin by swapping out $T_R \cup B'$ and swapping in $T_R^* \cup \overline{B}'$. The following list summarizes the important properties of this selection, the first point emphasizes that this swap will not improve the objective function since S is a locally optimum solution for the p-swap heuristic where $p = t^2 + 1$.

- $|B'| = |\overline{B}'| \leq t^2 + 1$ and $|T_R^*| \leq t^2$.
- T_R^* was swapped in and T_R was swapped out.
- For each $i \in T_R \cup T_B$ with $\deg(i) > 0$, i was swapped out and $\text{cent}(i)$ was swapped in.

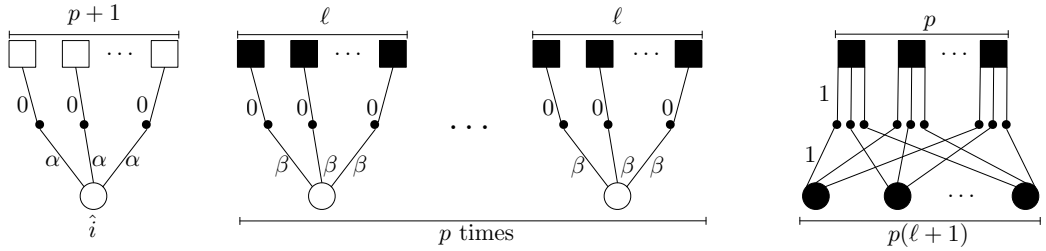


Figure 2 Illustration of the bad locality gap. Blue facilities are depicted with black and red facilities are depicted with white. The top facilities are the global optimum and the bottom are the local optimum (all of $\mathcal{R}$ and $\mathcal{B}$ is depicted in the picture). Each client is represented by a small black dot. There are $p^2 \cdot (\ell + 1)$ clients in the right group, one for each pair of local and global optimum facilities in the group. The metric is the shortest path metric of the presented graph, if two locations are not connected in the picture then their distance is a very large value. Every edge in the right-most group with $p^2(\ell + 1)$ clients has length 1. Recall $\beta = 2p$ and $\alpha = (\ell - p)2p$.

The following describes precisely the clients j that will be moved to $\text{cent}(\phi(o_j))$ in our analysis.

$$C_{bad} := [N(T_R \cup B') - N^*(T_R^* \cup \overline{B}')] \cap \{j : \phi(o_j) \in T_R \cup B'\}.$$

The following bound is generated from swapping out $T_R \cup B'$ and swapping in $T_R^* \cup \overline{B}'$ and follows from the same arguments we have been using throughout the paper.

▶ **Lemma 14.**

$$0 \le \sum_{j \in N^*(T_R^* \cup \overline{B}')} (c_j^* - c_j) + \sum_{j \in N(T_R \cup B') - C_{bad}} 2c_j^* + \sum_{j \in C_{bad}} (3c_j^* + c_j).$$

Next, let $\kappa_B : (T_B^* - \overline{B}') \to (T_B - B')$ be an arbitrary bijection of the remaining blue facilities that were not swapped. For every $i_b^* \in T_B^* - \overline{B}'$, consider the effect of swapping in i_b^* and swapping out $\kappa_B(i_b^*)$. Note that every facility i_b swapped out in this way has $\deg(i_b) = 0$. So we can derive two possible inequalities from such swaps.

$$0 \le \sum_{j \in N^*(i_b^*)} (c_j^* - c_j) + \sum_{j \in N(\kappa_B(i_b^*))} 2c_j^* \quad \text{and} \quad 0 \le \sum_{\mathrm{j} \in \mathrm{N}^*(\mathrm{i}_\mathrm{b}^*) \cap \mathrm{C}_{bad}} (\mathrm{c}_\mathrm{j}^* - \mathrm{c}_\mathrm{j}) + \sum_{\mathrm{j} \in \mathrm{N}(\kappa_\mathrm{B}(\mathrm{i}_\mathrm{b}^*))} 2\mathrm{c}_\mathrm{j}^*. \quad (4)$$

The second inequality follows from only reassigning clients $j \in N^*(i_b^*) \cap C_{bad}$ from s_j to o_j.

Adding the bound in Lemma 14 to the sum of both inequalities over all $i_b^* \in T_B^* - \overline{B}'$ and noting that $\kappa_B(T_B^* - \overline{B}) \cap (T_R \cup B') = \emptyset$, we see

$$0 \le \sum_{j \in N^*(T_R^* \cup T_B^*)} (c_j^* - c_j) + \sum_{j \in N(T_R \cup T_B)} 4c_j^*.$$

4 Locality Gaps

Here we prove Theorem 3. Let p, ℓ be integers satisfying $p \ge 1$ and $\ell \ge 2p$. Consider the instance with $k_r = p + 1$ and $k_b = p(\ell + 1)$ depicted in Figure 2. Here, $\beta = 2p$ and $\alpha = \beta \cdot (\ell - p)$.

The cost of the local optimum solution is $\alpha \cdot (p+1) + \beta \cdot p \cdot \ell + p^2(\ell+1)$ and the cost of the global optimum solution is simply $p^2(\ell+1)$. Through some careful simplification, we see the local optimum solution has cost at least $5 + \frac{2}{p} - \frac{10p}{\ell+1}$ times the global optimum solution.

To complete the proof of Theorem 3, we must verify that the presented local optimum solution indeed cannot be improved by swapping up to p facilities of each colour. The straightforward details appear in the full version of this paper.

5 Conclusion

We have demonstrated that a natural p-swap local search procedure for Budgeted Red-Blue Median is a $(5 + O(1/\sqrt{p}))$-approximation. This guarantees a better approximation ratio than the single-swap heuristic from [9], which we showed may find solutions whose cost is $(7-\epsilon) \cdot OPT$ for arbitrarily small ϵ. Our analysis is essentially tight in that the p-swap heuristic may find solutions whose cost is $(5 + \frac{2}{p} - \epsilon) \cdot OPT$.

More generally, one can ask about the p-swap heuristic for the generalization where there are many different facility colours. If the number of colours is part of the input then any local search procedure that swaps only a constant number of facilities in total cannot provide good approximation guarantees [10]. However, if the number of different colours is bounded by a constant, then perhaps one can get better approximations through multiple-swap heuristics.

However, generalizing the approaches taken with Budgeted Red-Blue Median to this setting seems more difficult; one challenge is that it is not possible to get such nicely structured blocks. It would also be interesting to see what other special cases of Matroid Median admit good local-search based approximations. For example, the Mobile Facility Location problem studied in [2] is another special case of Matroid Median that admits a $(3+\epsilon)$-approximation through local search.

Finally, the locality gap of the p-swap heuristic for k-Median is known to be $3 + \frac{2}{p}$ [3] and we have just shown it is at least $5 + \frac{2}{p}$ for Budgeted Red-Blue Median. Even if the multiple-swap heuristic for the generalization to a constant number of colours can provide a constant-factor approximation, this constant may be worse than the alternative 8-approximation obtained through Swamy's general Matroid Median approximation [15].

References

1. Ankit Aggarwal, Anand Louis, Manisha Bansal, Naveen Garg, Neelima Gupta, Shubham Gupta, and Surabhi Jain. A 3-approximation for facility location with uniform capacities. In *Proc. of IPCO*, pages 149–162, 2010.
2. Sara Ahmadian, Zachary Friggstad, and Chaitanya Swamy. Local-search based approximation algorithms for mobile facility location problems. In *Proc. of SODA*, pages 1607–1621, 2013.
3. Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM J. Comput.*, 33(3):544–562, 2004.
4. Manisha Bansal, Naveen Garg, and Neelima Gupta. A 5-approximation for capacitated facility location. In *Proc. of ESA*, pages 133–144, 2012.
5. Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median, and positive correlation in budgeted optimization. In *Proc. of SODA*, pages 737–756, 2015.
6. Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.*, 34(4):803–824, 2005.

7 Inge Li Gørtz and Viswanath Nagarajan. Locating depots for capacitated vehicle routing. In *Proc. of APPROX*, pages 230–241, 2011.

8 Anupam Gupta and Kanat Tangwongsan. Simpler analyses of local search algorithms for facility location. *CoRR*, abs/0809.2554, 2008.

9 MohammadTaghi Hajiaghayi, Rohit Khandekar, and Guy Kortsarz. Local search algorithms for the red-blue median problem. *Algorithmica*, 63(4):795–814, 2012.

10 Ravishankar Krishnaswamy, Amit Kumar, Viswanath Nagarajan, Yogish Sabharwal, and Barna Saha. The matroid median problem. In *Proc. of SODA*, pages 1117–1130, 2011.

11 Shi Li and Ola Svensson. Approximating k-median via pseudo-approximation. In *Proc. of STOC*, pages 901–910, 2013.

12 Mohammad Mahdian and Martin Pál. Universal facility location. In *Proc. of ESA*, pages 409–421, 2003.

13 Martin Pál, Éva Tardos, and Tom Wexler. Facility location with nonuniform hard capacities. In *Proc. of FOCS*, pages 329–338, 2001.

14 Zoya Svitkina and Éva Tardos. Facility location with hierarchical facility costs. *ACM Trans. Algorithms*, 6(2), 2010.

15 Chaitanya Swamy. Improved approximation algorithms for matroid and knapsack median problems and applications. In *Proc. of APPROX*, pages 403–418, 2014.

Improved Reduction from the Bounded Distance Decoding Problem to the Unique Shortest Vector Problem in Lattices*†

Shi Bai[1], Damien Stehlé[2], and Weiqiang Wen[3]

1 **ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), Lyon, France**
shi.bai@ens-lyon.fr

2 **ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), Lyon, France**
damien.stehle@ens-lyon.fr

3 **ENS de Lyon, Laboratoire LIP (U. Lyon, CNRS, ENSL, INRIA, UCBL), Lyon, France**
weiqiang.wen@ens-lyon.fr

Abstract

We present a probabilistic polynomial-time reduction from the lattice Bounded Distance Decoding (BDD) problem with parameter $1/(\sqrt{2}\cdot\gamma)$ to the unique Shortest Vector Problem (uSVP) with parameter γ for any $\gamma > 1$ that is polynomial in the lattice dimension n. It improves the BDD to uSVP reductions of [Lyubashevsky and Micciancio, CRYPTO, 2009] and [Liu, Wang, Xu and Zheng, Inf. Process. Lett., 2014], which rely on Kannan's embedding technique. The main ingredient to the improvement is the use of Khot's lattice sparsification [Khot, FOCS, 2003] before resorting to Kannan's embedding, in order to boost the uSVP parameter.

1998 ACM Subject Classification F.2.1 Numerical Algorithms and Problems

Keywords and phrases Lattices, Bounded Distance Decoding Problem, Unique Shortest Vector Problem, Sparsification

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.76

1 Introduction

A (full-rank) lattice $\mathcal{L}$ in dimension n is the set of all integer linear relations of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Q}^n$. The minimum $\lambda_1(\mathcal{L})$ quantifies the discreteness of $\mathcal{L}$: the smallest Euclidean distance between two distinct lattice vectors is $\lambda_1(\mathcal{L})$. A standard computational problem on lattices is the so-called Bounded Distance Decoding problem (BDD_α): Given as inputs a basis $\mathbf{B} = (\mathbf{b}_i)_i$ of a lattice $\mathcal{L}$ and a vector $\mathbf{t} \in \mathbb{Q}^n$ (called target vector) within distance $\alpha \cdot \lambda_1(\mathcal{L})$ of $\mathcal{L}$, the goal is to find a vector $\mathbf{b} \in \mathcal{L}$ closest to $\mathbf{t}$. Here $\alpha > 0$ is a problem parameter, which may be a function of the lattice dimension n. The hardness of BDD was initially studied in the context of linear codes by Vardy in [22], and later in the context of lattices by Liu *et al.* in [14].

In communications theory, BDD models the task of decoding in the context of continuous channels with white Gaussian noise [6]. The information to be transmitted is stored in a

* A full version of the paper is available at http://eprint.iacr.org/2016/753.

† This work has been supported by ERC Starting Grant ERC-2013-StG-335086-LATTAC.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 76; pp. 76:1–76:12

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

lattice vector, and the receiver should recover this vector from a noisy version thereof. The knowledge of the signal-to-noise ratio implies a bound on the distance from the noisy vector to the lattice. Decoding a white Gaussian noise channel can be seen as a version of BDD in which the distance to the lattice follows a prescribed distribution.

In cryptography, BDD is closely related to the Learning With Errors problem (LWE) [19], which serves as a security foundation for numerous cryptographic primitives. When the number of requested LWE samples is bounded (which is most often the case in cryptographic constructions), LWE may be viewed as a variant of BDD in which the offset from $\mathbf{t}$ to $\mathcal{L}$ is Gaussian (like in the decoding context), and $\mathcal{L}$ is randomly sampled.

A common approach to solve BDD is via Kannan's embedding technique [9, Se. 6]. The principle is to map the offset between $\mathbf{t}$ and a closest lattice vector to $\mathbf{t}$, to a shortest non-zero vector in an $(n+1)$-dimensional lattice. Lattice reduction [12, 20] and short lattice vector enumeration [8, 7] may then be used to find shortest non-zero vectors in the $(n+1)$-dimensional lattice. Formally, Kannan's embedding technique is a reduction from BDD to a variant of the Shortest Vector Problem (SVP) in which the pair of shortest non-zero vectors in the lattice under scope are known to be much shorter than any other lattice vector not parallel to them. For a lattice $\mathcal{L}$, we define the second minimum $\lambda_2(\mathcal{L})$ as the minimal radius of a zero-centered ball that contains two or more linearly independent vectors from $\mathcal{L}$. The unique Shortest Vector Problem (uSVP_γ) of parameter $\gamma \geq 1$ consists in finding a shortest non-zero vector in a lattice $\mathcal{L}$ described by an input basis $\mathbf{B} = (\mathbf{b}_i)_i$, under the promise that $\lambda_2(\mathcal{L}) \geq \gamma \cdot \lambda_1(\mathcal{L})$. This reduction was analyzed by Lyubashevsky and Micciancio in [15], who showed that $\text{BDD}_{1/(2\gamma)}$ reduces to uSVP_γ for any $\gamma \geq 1$. Later, Liu *et al.* [13] refined the analysis of Lyubashevsky and Micciancio and proved that BDD_{1/γ_1} reduces to uSVP_γ with $\gamma_1 = \sqrt{3/(4-\gamma^2)}\gamma + 1$, for any $\gamma \in (1, 1.9318)$. It is folklore [3, 17] that the analysis can be tightened even more, resulting in a proof that BDD_{1/γ_1} reduces to uSVP_γ with $\gamma_1 = (2\gamma^2 + 2\lfloor\gamma\rfloor\lfloor\gamma+1\rfloor)/(2\lfloor\gamma\rfloor+1)$, for any $\gamma \geq 1$. For the sake of completeness, we give a proof in Appendix A.1 of the full version. Note that in the case of $\gamma = 1$ (and in fact all integral γ), all three results are identical: $\text{BDD}_{1/2}$ reduces to uSVP_1.

Our result. We give a probabilistic polynomial-time reduction from $\text{BDD}_{1/(\sqrt{2}\gamma)}$ to uSVP_γ, for any $\gamma \geq 1$ that is polynomially bounded as a function of n. As clearly visible in Figure 1, this reduction supersedes all prior results with respect to the BDD problem parameter. In particular, we reduce $\text{BDD}_{1/\sqrt{2}}$ to uSVP_1. Our improvement comes with two weaknesses: the reduction is probabilistic and restricted to polynomially bounded γ. Like prior reductions, the dimension of the uSVP instance is only one more than the dimension of the BDD instance.

Technical overview. We illustrate our improvement with the case of $\text{BDD}_{1/2}$. Given the $\text{BDD}_{1/2}$ instance $(\mathbf{B}, \mathbf{t})$, Kannan's embedding consists in constructing the following uSVP_1 instance:

$$\mathbf{B}' = \begin{pmatrix} \mathbf{B} & \mathbf{t} \\ \mathbf{0} & d \end{pmatrix} \in \mathbb{Q}^{n+1},$$

with $d = \text{dist}(\mathbf{t}, \mathcal{L}) \leq \lambda_1(\mathcal{L})/2$, where $\mathcal{L}$ is the lattice spanned by $\mathbf{B}$ (in fact, the reduction does not know d, but this is a mere technical problem which can be handled easily, as explained in Section 2). If $\mathbf{c}$ denotes a closest vector to $\mathbf{t}$ in $\mathcal{L}$ then it may be proved that the vector $\mathbf{s}' = ((\mathbf{c}-\mathbf{t})^{\mathrm{T}}, -d)^{\mathrm{T}}$ is a shortest non-zero vector of lattice $\mathcal{L}'$ of basis $\mathbf{B}'$. Now, let $\mathbf{s}$ denote a shortest non-zero vector in $\mathcal{L}$ and assume that $\mathbf{t}$ is exactly halfway between $\mathbf{c}$ and $\mathbf{c}+\mathbf{s}$. Then both $\mathbf{s}'$ and $\mathbf{s}' + (\mathbf{s}^{\mathrm{T}}, 0)^{\mathrm{T}}$ in $\mathcal{L}'$ have norm $\sqrt{2} \cdot d$ but are linearly independent.

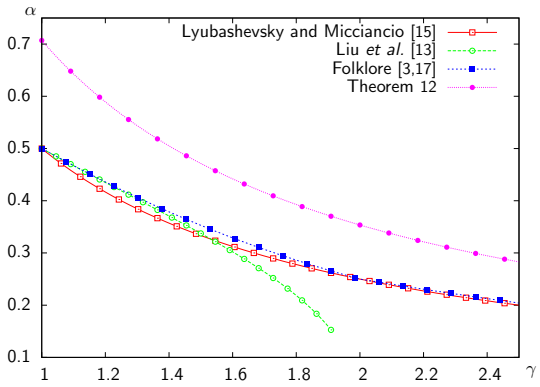


Figure 1 Comparison between prior reductions from BDD_α to uSVP_γ, and ours.

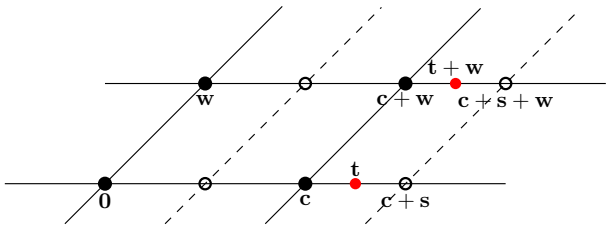


Figure 2 An example of sparsification for $\mathrm{BDD}_{1/2}$ (here $\mathbf{w} \in \mathcal{L}_{p,\mathbf{z}}/\mathcal{L}$).

This shows that we can have $\lambda_2(\mathcal{L}') = \lambda_1(\mathcal{L}')$ and obtain a uSVP_γ instance with $\gamma = 1$. This is a limitation of Kannan's embedding and hence of its analyzes.

We modify the reduction to increase the ratio $\lambda_2(\mathcal{L}')/\lambda_1(\mathcal{L}')$. To achieve this, we use lattice sparsification on $\mathcal{L}$. It provides a full-rank sublattice $\mathcal{L}_{\mathrm{sparse}} \subseteq \mathcal{L}$ that still contains a closest vector $\mathbf{c} \in \mathcal{L}$ to $\mathbf{t}$, but no other close-by vector. We consider the vectors of $\mathcal{L}$ whose coordinates with respect to a basis $\mathbf{B}$ satisfy a linear equation modulo some prime integer p: $\mathcal{L}_{\mathrm{sparse}} = \mathcal{L}_{p,\mathbf{z}} = \{\mathbf{b} \in \mathcal{L} : \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{b} \rangle = 0 \bmod p\}$, for some vector $\mathbf{z} \in \mathbb{Z}_p^n$. This technique was first introduced by Khot in [10, 11]. To guarantee that vectors in $\mathcal{L}$ remain in the sparsified set with probability close to $1/p$, a uniform coset of $\mathcal{L}_{p,\mathbf{z}}$ (modulo $\mathcal{L}$) was considered in [4, 5]. Technically, we use the formulation from [21] of the latter variant.

The aim of sparsification in our context is to keep a closest vector $\mathbf{c} \in \mathcal{L}$ to $\mathbf{t}$, and remove as many as nearby vectors of $\mathcal{L}$ as possible. After sparsification, vector $\mathbf{c}$ remains in the sparse lattice (with non-negligible probability), and all other remaining vectors are much further away from $\mathbf{t}$. For $\mathrm{BDD}_{1/2}$, a simple example of sparsification is shown in Figure 2: there are two points simultaneously closest to the target point; then sparsification is used to remove one of the closest points (either is fine); in the sparse lattice, the closest vector is much closer than any other lattice vector. In Figure 2, after sparsification, all the lattice vectors labelled with filled dots are kept, e.g., vector $\mathbf{c} + \mathbf{w}$, and other vectors labelled with hollow dots are removed, e.g., vector $\mathbf{c} + \mathbf{s} + \mathbf{w}$.

The probability of keeping a vector of $\mathcal{L}$ in the sparsified set is essentially $1/p$. As we want the probability of keeping $\mathbf{c}$ to be non-negligible, we are hence restricted to taking $p \leq \mathrm{poly}(n)$. As a result, we cannot remove more than polynomially many close-by vectors, because each

one individually is removed with probability $\approx 1 - 1/p$ (a precise statement is given in Lemma 8). To assess the limitation of our reduction, we are hence interested in the largest value of α such that for any lattice $\mathcal{L}$ and any vector $\mathbf{t}$, there are at most $\mathrm{poly}(n)$ vectors within distance $\alpha \cdot \lambda_1(\mathcal{L})$ from $\mathbf{t}$. The quantity $\alpha \cdot \lambda_1(\mathcal{L})$ can be viewed as the worst-case list-decoding radius. Interestingly, this problem was studied by Ajtai [1] and Micciancio [16] in the context of proving hardness of SVP. Proofs of the following two statements may be found in [18, Chap. 5]:

- For any lattice $\mathcal{L}$ and vector $\mathbf{t}$, there are $\leq 2n$ vectors of $\mathcal{L}$ within distance $\lambda_1(\mathcal{L})/\sqrt{2}$ from $\mathbf{t}$.
- For any $\alpha > 1/\sqrt{2}$, there exists $\varepsilon > 0$ such that for any sufficiently large n we can find an n-dimensional lattice $\mathcal{L}$ and a vector $\mathbf{t}$ such that there are $\geq 2^{n^\varepsilon}$ vectors of $\mathcal{L}$ within distance $\alpha \cdot \lambda_1(\mathcal{L})$ from $\mathbf{t}$.

The overall reduction consists in first sparsifying $\mathcal{L}$ to $\mathcal{L}_{p,\mathbf{z}}$ and shifting $\mathbf{t}$ (as we use a coset of $\mathcal{L}_{p,\mathbf{z}}$), and then resorting to Kannan's embedding. To increase the ratio $\lambda_2(\mathcal{L}')/\lambda_1(\mathcal{L}')$, we decrease the bottom-right entry in $\mathbf{B}'$ from d to $k \cdot d$ for some $k < 1$. Geometrically, this has the effect of limiting the contribution of the extra dimension. This idea was already used in [13], but we decrease k even further, to $1/\mathrm{poly}(n)$. An additional difficulty, related to this decrease of k, is that short vectors in $\mathcal{L}'$ may be obtained by using multiples of $\mathbf{t}$. Let $m \geq 2$ and $\mathbf{d} \in \mathcal{L}$ closest to $m\mathbf{t}$. Then, vector $((\mathbf{d} - m\mathbf{t})^{\mathrm{T}}, mkd)^{\mathrm{T}}$ may be very short (if very unlucky, it has norm mkd). We remove such annoying vectors with sparsification.

Open problems. In [15], Lyubashevsky and Micciancio considered the relative hardness of BDD and uSVP. They obtained a reduction from $\mathrm{BDD}_{1/(2\gamma)}$ to uSVP_γ, and a reduction from uSVP_γ to $\mathrm{BDD}_{1/\gamma}$ (for all $\gamma \geq 1$). This led them to conjecture that it may be possibly to show that (i)- $\mathrm{uSVP}_{\gamma/2}$ reduces to $\mathrm{BDD}_{1/\gamma}$, or (ii)- $\mathrm{BDD}_{1/\gamma}$ reduces to uSVP_γ, or (iii)- uSVP_γ reduces to $\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$ and $\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$ reduces to uSVP_γ. By showing the second half of (iii), (i) becomes very unlikely.

Independently, it would be interesting to make our reduction deterministic and let it work even for parameters γ that are not $\leq \mathrm{poly}(n)$.

Notation. For a lattice $\mathcal{L}$, a point $\mathbf{t}$, a radius r, we define $\mathcal{B}(\mathbf{t}, r) = \{\mathbf{x} : \|\mathbf{x} - \mathbf{t}\| \leq r\}$. We let $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$ denote the distance between $\mathbf{t}$ and lattice $\mathcal{L}(\mathbf{B})$. We always represent the basis of lattice in column form. If S is a finite set, we let $\#S$ denote its cardinality.

2 Reminders

In this section, we recall basic facts on lattices and lattice problems. We then consider lattice sparsification and its use in the context of BDD instances.

2.1 Lattice problems

We refer the reader to [18] for an introduction to the computational aspects of lattices.

▶ **Definition 1** (Lattice). An n-dimensional lattice $\mathcal{L} \subseteq \mathbb{Q}^m$ $(m \geq n)$ is a discrete additive subgroup of $\mathbb{R}^m$. The lattice $\mathcal{L}$ is the set of all integral linear combinations of n linearly independent basis vectors $\mathbf{B} = \{\mathbf{b}_1, \cdots, \mathbf{b}_n\} \subseteq \mathbb{Q}^m$. In other words, we have

$$\mathcal{L}(\mathbf{B}) = \left\{ \sum_{i \in [n]} u_i \mathbf{b}_i : \ \mathbf{u} \in \mathbb{Z}^n \right\}.$$

▶ **Definition 2** (Successive minima)**.** For any lattice $\mathcal{L}$, the i-th minimum $\lambda_i(\mathcal{L})$ is the radius of the smallest ball with center $\mathbf{0}$ and containing i linearly independent lattice vectors:

$$\lambda_i(\mathcal{L}) = \inf\{r : \dim(\mathrm{span}(\mathcal{L} \cap \mathcal{B}(\mathbf{0}, r))) \geq i\}.$$

In this work, we investigate the respective hardness of uSVP_γ and BDD_α defined below, when the lattice dimension n goes to infinity. The problem parameters γ and α can be functions of n.

▶ **Definition 3** (Unique Shortest Vector Problem (uSVP_γ))**.** Let $\gamma \geq 1$. Given as input a lattice basis $\mathbf{B}$ such that $\lambda_2(\mathbf{B}) \geq \gamma \cdot \lambda_1(\mathbf{B})$, the goal is to find a non-zero vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ of norm $\lambda_1(\mathcal{L}(\mathbf{B}))$. The Shortest Vector Problem (SVP) corresponds to $\gamma = 1$.

In the literature (in [15], for example), uSVP is sometimes be defined with a strict lower bound on $\lambda_2(\mathbf{B})$. We allow equality (as in [13]), as it is more convenient in our proofs. Note that Lemma 5 below implies that these two variants are equivalent.

▶ **Definition 4** (Bounded Distance Decoding (BDD_α))**.** Let $\alpha > 0$. Given as inputs a lattice basis $\mathbf{B}$ and a vector $\mathbf{t}$ such that $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \leq \alpha \cdot \lambda_1(\mathbf{B})$, the goal is to find a lattice vector $\mathbf{v} \in \mathcal{L}(\mathbf{B})$ closest to $\mathbf{t}$.

Note that in some works, the range of α is restricted to $(0, 1/2)$. This is to guarantee that there is exactly one element of $\mathcal{L}$ in the ball of radius $\alpha \cdot \lambda_1(\mathcal{L})$ centered on $\mathbf{t}$. The problem is well-defined even for large α, and in this work we actually consider $\alpha \geq 1/2$.

In the next lemma, it is stated that BDD_α is equivalently hard for any parameter α' that is within a factor $(1 - 1/n)^c$ of α, for any constant c.

▶ **Lemma 5** ([15, Cor. 2])**.** *For any $\alpha > 0$, any constant $c > 0$, there is a polynomial-time reduction from BDD_α to $\mathrm{BDD}_{\alpha(1-1/n)^c}$.*

2.2 Approximation results

Given as input an n-dimensional lattice basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$, it is possible to find a non-zero vector that has norm at most $2^{n/2} \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$ in time polynomial in n and also the bit-sizes of the entries of $\mathbf{B}$, by using the LLL algorithm [12]. Further, by using the Babai round-off algorithm [2] with inputs an n-dimensional lattice basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ and a target vector $\mathbf{t} \in \mathbb{Q}^n$, one obtains an approximation of the distance between $\mathbf{t}$ and $\mathcal{L}(\mathbf{B})$ within a factor $2^{n/2}$ in time polynomial in n and also the bit-sizes of the entries of $\mathbf{B}$ and $\mathbf{t}$.

▶ **Lemma 6** ([12, Prop. 1.6])**.** *There exists a polynomial-time algorithm that, given as input an n-dimensional lattice basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$, outputs $\ell \in \lambda_1(\mathcal{L}) \cdot [1, 2^{n/2})$.*

▶ **Lemma 7** ([2, Thm. 3.1])**.** *There exists a polynomial-time algorithm that, given as input an n-dimensional lattice basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ and a target vector $\mathbf{t} \in \mathbb{Q}^n$, outputs $d \in \mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \cdot [1, 2^{n/2})$.*

We will rely on much tighter approximations to $\lambda_1(\mathcal{L}(\mathbf{B}))$ (resp. $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$) than provided by Lemmata 6 and 7. We explain here why we may assume that we know $\ell \in \lambda_1(\mathcal{L}(\mathbf{B})) \cdot [1, 1/(1 - 1/n))$ (resp. $d \in \mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B})) \in [1, 1/(1 - 1/n))$).

Our reduction is from BDD, whose candidate solutions can be compared in polynomial time. Assume the reduction finds the optimal solution in one case among polynomially many, but that we do not know which one. Then we may call the reduction this polynomially many times, and keep a best solution among the returned ones. Concretely, our reduction will be

proved correct if we know a tight approximation to $\lambda_1(\mathcal{L}(\mathbf{B}))$ and $\mathrm{dist}(\mathbf{t}, \mathcal{L}(\mathbf{B}))$, where $(\mathbf{B}, \mathbf{t})$ is the BDD instance. We can assume without loss of generality that we have these tight approximations, as the interval $[1, 2^{n/2})$ may be covered by polynomially many intervals of the form $x \cdot [1, 1/(1-1/n))$ for well-chosen rational x's.

2.3 Lattice sparsification

Our techniques rely on lattice sparsification, and, more concretely, on the following lemma.

▶ **Lemma 8** ([21, Cor. 2.16]). *For any prime p, collection of vectors $\mathbf{v}_1, \cdots, \mathbf{v}_N \in \mathbb{Z}_p^n \setminus \{\mathbf{0}\}$, and $\mathbf{x} \notin \{\mathbf{v}_i\}_{i \le N}$, we have*

$$\frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}} \le \Pr_{\mathbf{z},\mathbf{u} \hookleftarrow U(\mathbb{Z}_q^n)} \left[\begin{array}{ll} \forall i, & \langle \mathbf{z}, \mathbf{v}_i + \mathbf{u} \rangle \neq 0 \bmod p \\ & \langle \mathbf{z}, \mathbf{x} + \mathbf{u} \rangle = 0 \bmod p \end{array} \right] \le \frac{1}{p} + \frac{1}{p^n}.$$

The upper bound in Lemma 8 is not used in this work, but we keep it to show that the difference between the upper and lower bound is small, and thus that the lower bounds is almost tight. Lemma 8 leads to the definition of a sublattice that will be used in our reduction from BDD to uSVP. The lemma below explains that we can efficiently compute a basis of the sublattice.

▶ **Lemma 9.** *There exists a polynomial-time algorithm which, given as inputs a basis $\mathbf{B} \in \mathbb{Q}^{n \times n}$ of an n-dimensional lattice $\mathcal{L}$, an integer p and a vector $\mathbf{z} \in \mathbb{Z}_p^n$, outputs a basis $\mathbf{B}_{p,\mathbf{z}}$ of the lattice $\mathcal{L}_{p,\mathbf{z}} = \{\mathbf{x} \in \mathcal{L} \mid \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{x} \rangle = 0 \bmod p\}$.*

Proof. According to the definition of the lattice $\mathcal{L}_{p,\mathbf{z}}$, we have

$$\langle \mathbf{z}, \mathbf{y} \rangle = 0 \bmod p,$$

where $\mathbf{y} = \mathbf{B}^{-1}\mathbf{x}$ and $\mathbf{x} \in \mathcal{L}_{p,\mathbf{z}}$. We can obtain a basis $\mathbf{S}$ of the kernel $\mathbf{y}$ over $\mathbb{Z}^n$. We compute the column Hermite normal form of $\begin{bmatrix} \mathbf{S} & p\mathbf{I}_n \end{bmatrix} \in \mathbb{Z}^{n \times 2n}$; and obtain the nonzero columns $\mathbf{S}' \in \mathbb{Z}^{n \times n}$. The columns of $\mathbf{S}'$ generate the lattice orthogonal to $\mathbf{z} \pmod p$. In the end, we compute $\mathbf{B}_{p,\mathbf{z}} = \mathbf{B}\mathbf{S}'$, which is a basis for the lattice $\mathcal{L}_{p,\mathbf{z}}$. ◀

Below, we state that for any two lattice vectors in $\mathcal{L}$ with distance smaller than $p \cdot \lambda_1(\mathcal{L})$ where p is an integer, the coordinates of these two lattice vectors differ modulo p.

▶ **Lemma 10.** *For any basis $\mathbf{B}$, any integer p and any pair of lattice vectors $\mathbf{x} \neq \mathbf{v}$ with $\|\mathbf{x} - \mathbf{v}\| < p \cdot \lambda_1(\mathcal{L}(\mathbf{B}))$, we have that $\mathbf{B}^{-1}\mathbf{x} \neq \mathbf{B}^{-1}\mathbf{v} \bmod p$.*

Proof. For all lattice vector $\mathbf{a}$, we let $\widetilde{\mathbf{a}}$ denote its coordinate vector under the basis $\mathbf{B}$. Assume by contradiction that $\widetilde{\mathbf{x}} = \widetilde{\mathbf{v}} \bmod p$. Equivalently, we have $\mathbf{x} - \mathbf{v} \in p \cdot \mathcal{L}(\mathbf{B})$.

Combined with $\mathbf{x} \neq \mathbf{v}$, we have $\|\mathbf{x} - \mathbf{v}\| \ge p \cdot \lambda_1(\mathcal{L})$, which is in contradiction with $\|\mathbf{x} - \mathbf{v}\| < p \cdot \lambda_1(\mathcal{L})$. As a result, we have $\widetilde{\mathbf{x}} \neq \widetilde{\mathbf{v}} \bmod p$. ◀

The proof from [18] of the lemma below is by induction. It goes fast over a subtle counting argument when reducing the problem in dimension $n+1$ to dimension n. We briefly recall the proof and give more explanations on the counting argument in Appendix A.2 of the full version.

▶ **Lemma 11** ([18, Thm. 5.2]). *For any n-dimensional lattice $\mathcal{L}$ and any vector $\mathbf{t} \in \mathbb{Q}^n$, we have $\#\mathcal{L} \cap \mathcal{B}(\mathbf{t}, \lambda_1(\mathcal{L})/\sqrt{2}) \le 2n$.*

We will use the above three lemmas in the following way to tackle BDD. Consider the coordinate vectors (with respect to some arbitrary basis) of all lattice vectors in $\mathcal{B}(\mathbf{t}, r)$ with $r = \lambda_1(\mathcal{L})/\sqrt{2}$ and some arbitrary target vector $\mathbf{t}$. First, according to Lemma 10 with $p > 2\sqrt{2}$, we can obtain that one of the coordinate vectors differs from all the others modulo p. Further, by Lemma 8, a uniformly chosen vector $\mathbf{z}$ over $\mathbb{Z}_p$ is orthogonal to exactly one of the coordinate vectors (shifted by another uniformly chosen vector $\mathbf{u}$) with non-negligible probability. Assume that this orthogonal coordinates vector is the coordinates vector of a closest lattice vector to $\mathbf{t}$: this occurs with non-negligible probability as $\#\mathcal{L} \cap \mathcal{B}(\mathbf{t}, r) \leq 2n$. We can consider the sublattice $\mathcal{L}_{p,\mathbf{z}}$, which contains just this BDD solution and none of the other vectors of $\mathcal{L} \cap \mathcal{B}(\mathbf{t}, r)$. This will help us ensuring a large gap between the first two minima of the uSVP lattice in the BDD to uSVP reduction.

Note that $\mathbf{u}$ is necessary, as otherwise some superfluous vectors (including vector $\mathbf{0}$) could be multiples of the solution vector and hence always stay in $\mathcal{L}_{p,\mathbf{z}}$ if the solution vector does.

3 Reducing $\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$ to u$\mathrm{SVP}_{\gamma(1+\varepsilon)}$

In this section, we use a $\mathrm{uSVP}_{\gamma(1+\varepsilon)}$ solver with $\varepsilon = \Omega(1/n)$ to solve $\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$.

▶ **Theorem 12.** *Let $\gamma(n) \leq \mathrm{poly}(n)$. There is a probabilistic polynomial-time reduction from $\mathrm{BDD}_{1/(\sqrt{2}\gamma)}$ to $\mathrm{uSVP}_{\gamma(1+\varepsilon)}$, where $\varepsilon = \Omega(1/n)$.*

Thanks to Lemma 5, it suffices to reduce $\mathrm{BDD}_{(1-1/n)/(\sqrt{2}\gamma)}$ to $\mathrm{uSVP}_{\gamma(1+\varepsilon)}$. Let us first describe the reduction.

Algorithm 1. The $\mathrm{BDD}_{(1-1/n)/(\sqrt{2}\gamma)}$ to $\mathrm{uSVP}_{\gamma(1+\varepsilon)}$ reduction.

Input: a basis $\mathbf{B} = \{\mathbf{b}_i\}_{i\in[n]}$ of an n-dimensional lattice $\mathcal{L} \subseteq \mathbb{Q}^n$, and a target point $\mathbf{t} \in \mathbb{Q}^n$.

Output: a lattice point $\mathbf{c}$ such that $\|\mathbf{c} - \mathbf{t}\| = \mathrm{dist}(\mathbf{t}, \mathcal{L})$.

0. Guess $d_0 \in [d, d/(1-1/n))$ and $\ell_0 \in [\ell, \ell/(1-1/n))$, where $d = \mathrm{dist}(\mathbf{t}, \mathcal{L})$ and $\ell = \lambda_1(\mathcal{L})$ (see Section 2.2).

1. Compute p the smallest prime greater than $4\gamma n^2$.
Sample $\mathbf{z}, \mathbf{u}$ uniformly and independently in $\mathbb{Z}_p^n$.
Compute $\mathbf{w} = \mathbf{B}\bar{\mathbf{u}} \in \mathcal{L}$, such that $\bar{\mathbf{u}} = \mathbf{u} \bmod p$ and $\|\mathbf{t} + \mathbf{w}\| \geq (n+1)\ell_0/\sqrt{2}$.
Use the algorithm of Lemma 9 to compute a basis $\mathbf{B}_{p,\mathbf{z}}$ of $\mathcal{L}_{p,\mathbf{z}} = \{\mathbf{b} \in \mathcal{L} : \langle \mathbf{z}, \mathbf{B}^{-1}\mathbf{b}\rangle = 0 \bmod p\}$.

2. Set $k = 1/(n-1)$. Define

$$\mathbf{B}' = \begin{pmatrix} \mathbf{B}_{p,\mathbf{z}} & \mathbf{t}+\mathbf{w} \\ \mathbf{0} & kd_0 \end{pmatrix}.$$

3. Run the $\mathrm{uSVP}_{\gamma(1+\varepsilon)}$ solver on input $\mathbf{B}'$. Let $\mathbf{s}' = ((\mathbf{s}'_1)^{\mathrm{T}}, s'_2)^{\mathrm{T}}$ be its output. Output $\mathbf{s}'_1 + \mathbf{t}$.

It may be checked that the above algorithm runs in polynomial time. The rest of the section is devoted to proving its correctness.

In this reduction, we are given a $\mathrm{BDD}_{(1-1/n)/(\sqrt{2}\gamma)}$ instance $(\mathbf{B}, \mathbf{t})$. Let $\mathbf{c} \in \mathcal{L}$ be a closest vector to $\mathbf{t}$. In order to construct a uSVP instance, our strategy is to use lattice sparsification to keep only one closest vector $\mathbf{c} + \mathbf{w}$ for some lattice shift $\mathbf{w}$ closest to $\mathbf{t} + \mathbf{w}$ (the shift vector $\mathbf{w}$ comes from Lemma 8). As the sparsification results in a lattice, we not only keep $\mathbf{c} + \mathbf{w}$, but also the $m \cdot (\mathbf{c} + \mathbf{w})$'s for all $m \leq \gamma n$. Simultaneously, all other vectors inside the balls with centers $\{m \cdot (\mathbf{t} + \mathbf{w})\}_{m\leq\gamma n}$ and radius $\lambda_1(\mathcal{L})/\sqrt{2}$ are regarded

as superfluous vectors and removed through sparsification. For the first γ balls, we have $m \cdot (\mathbf{c}+\mathbf{w}) \in \mathcal{B}(m \cdot (\mathbf{t}+\mathbf{w}), \lambda_1(\mathcal{L})/\sqrt{2})$. We can keep exactly one vector inside every ball with sparsification over these balls. However, for $m > \gamma$, all closest points to $\mathbf{t}$ may fall out of the corresponding ball, but may end up in another relevant ball: vector $i \cdot (\mathbf{c}+\mathbf{w})$ may belong to $\mathcal{B}(j \cdot (\mathbf{t}+\mathbf{w}), \lambda_1(\mathcal{L})/\sqrt{2})$ for some $j \neq i$. As a consequence, there can be more than one lattice vector inside a ball, which may result in no gap between first two minima of the uSVP oracle input lattice. In order to avoid this, we make every two balls far away from each other by choosing $\mathbf{w}$ such that $\mathbf{t}+\mathbf{w}$ is long.

▶ **Lemma 13.** *Consider a basis $\mathbf{B}$ of an n-dimensional lattice $\mathcal{L}$, a vector $\mathbf{c} \in \mathcal{L}$ and a vector $\mathbf{t} \in \mathbb{R}^n$ such that $\|\mathbf{c}-\mathbf{t}\| \leq r = \lambda_1(\mathcal{L})/(\sqrt{2}\gamma)$ for some $\gamma > 0$. Let p prime with $p \geq n+1$. For any $\mathbf{z} \in \mathbb{Z}_p^n$, We have*

$$\Pr_{\mathbf{u},\mathbf{z} \hookleftarrow U(\mathbb{Z}_p^n)} \left[\begin{array}{ccc} \mathbf{c}+\mathbf{w} & \in & \mathcal{L}_{p,\mathbf{z}} \cap \mathcal{B}(\mathbf{t}+\mathbf{w}, \gamma \cdot r) \\ \mathbb{Z} \cdot (\mathbf{c}+\mathbf{w}) & \supseteq & \mathcal{L}_{p,\mathbf{z}} \cap \bigcup\limits_{i \leq \gamma n} \mathcal{B}(i \cdot (\mathbf{t}+\mathbf{w}), \gamma \cdot r) \end{array} \right] \geq \frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}},$$

where $\mathbf{w}$ is arbitrary such that $\mathbf{B}^{-1}\mathbf{w} = \mathbf{u} \bmod p$ and $N = \#\mathcal{L} \cap \cup_{i \leq \gamma n} \mathcal{B}(i \cdot (\mathbf{t}+\mathbf{w}), \gamma \cdot r)$.

Further, if $\mathbf{w}$ is chosen such that $\|\mathbf{t}+\mathbf{w}\| > \gamma(n+1)r$, we have, for all $i \in [\gamma n]$,

$$i \cdot (\mathbf{c}+\mathbf{w}) \notin \bigcup_{j \neq i, j \in [\gamma n]} \mathcal{B}\left(j \cdot (\mathbf{t}+\mathbf{w}), \gamma \cdot r\right).$$

Proof. For $i \in [\gamma n]$, we define $N_i = \#\mathcal{L} \cap \mathcal{B}(i \cdot \mathbf{t}, \gamma \cdot r) \setminus \{i \cdot \mathbf{c}\}$ and $\{\mathbf{v}_{ij}\}_{j \in [N_i]} = (\mathcal{L} \cap \mathcal{B}(i \cdot \mathbf{t}, \gamma \cdot r)) \setminus \{i \cdot \mathbf{c}\}$. For any $\mathbf{v} \in \mathcal{L}$, we use $\widetilde{\mathbf{v}}$ to denote the coordinate of $\mathbf{v}$ under the basis $\mathbf{B}$. We claim that, with probability $\geq 1/p - N/p^2 - N/p^{n-1}$, the vector $\mathbf{z}$ is orthogonal (modulo p) to $\widetilde{\mathbf{c}}$, and at the same time not orthogonal to any $\widetilde{\mathbf{v}}_{ij}$ for $i \in [\gamma n]$ and $j \in [N_i]$.

We have, for all $i \in [\gamma n]$ and $j \in [N_i]$,

$$\|i \cdot \mathbf{c} - \mathbf{v}_{ij}\| \leq i \cdot \|\mathbf{c}-\mathbf{t}\| + \|i \cdot \mathbf{t} - \mathbf{v}_{ij}\| \leq (n+1)(\gamma r) = \frac{n+1}{\sqrt{2}} \cdot \lambda_1(\mathcal{L}).$$

By choice of p, this is smaller than $p \cdot \lambda_1(\mathcal{L})$. Thanks to Lemma 10, we have $i \cdot \widetilde{\mathbf{c}} \neq \widetilde{\mathbf{v}}_{ij} \bmod p$. Moreover, as p is prime and $p \geq \gamma n + 1 > i$, we have $\widetilde{\mathbf{c}} \neq \frac{1}{i} \cdot \widetilde{\mathbf{v}}_{ij} \bmod p$.

Now we apply Lemma 8 with $\widetilde{\mathbf{c}}$ and $\{\frac{1}{i} \cdot \widetilde{\mathbf{v}}_{ij}\}_{i \in [\gamma n], j \in [N_i]}$. We have

$$\Pr_{\mathbf{z},\mathbf{u} \hookleftarrow U(\mathbb{Z}_p^n)} \left[\begin{array}{lcl} \forall i,j: & \langle \mathbf{z}, \frac{1}{i} \cdot \widetilde{\mathbf{v}}_{ij} + \mathbf{u} \rangle & \neq 0 \bmod p \\ & \langle \mathbf{z}, \widetilde{\mathbf{c}} + \mathbf{u} \rangle & = 0 \bmod p \end{array} \right] \geq \frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}}.$$

As p is prime and sufficiently large, the inequality $\langle \mathbf{z}, \frac{1}{i} \cdot \widetilde{\mathbf{v}}_{ij} + \mathbf{u} \rangle \neq 0 \bmod p$ is equivalent to $\langle \mathbf{z}, \widetilde{\mathbf{v}}_{ij} + i \cdot \mathbf{u} \rangle \neq 0 \bmod p$. Therefore

$$\Pr_{\mathbf{z},\mathbf{u} \hookleftarrow U(\mathbb{Z}_p^n)} \left[\begin{array}{lcl} \forall i,j: & \langle \mathbf{z}, \widetilde{\mathbf{v}}_{ij} + i \cdot \mathbf{u} \rangle & \neq 0 \bmod p \\ & \langle \mathbf{z}, \widetilde{\mathbf{c}} + \mathbf{u} \rangle & = 0 \bmod p \end{array} \right] \geq \frac{1}{p} - \frac{N}{p^2} - \frac{N}{p^{n-1}}.$$

This proves the first claim of the lemma.

Let $i \neq j \leq \gamma n$. Then, by the triangle inequality and the assumption on $\mathbf{w}$, we have:

$$\|i \cdot (\mathbf{c}+\mathbf{w}) - j \cdot (\mathbf{t}+\mathbf{w})\| \geq |j-i| \cdot \|\mathbf{t}+\mathbf{w}\| - i\|\mathbf{c}-\mathbf{t}\| > \gamma(n+1)r - (\gamma n)r = \gamma r.$$

This completes the proof of the lemma. ◀

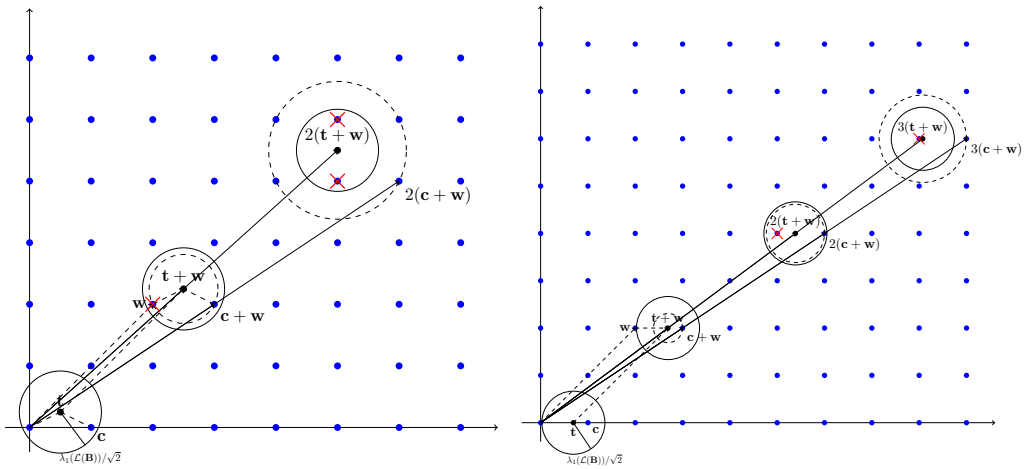


Figure 3 Sparsification for a $\mathrm{BDD}_{1/\sqrt{2}}$ instance (left) and for a $\mathrm{BDD}_{1/(2\sqrt{2})}$ instance (right).

As we have $p > 4\gamma n^2 \geq 2N$ (thanks to Lemma 11), with non-negligible probability, none of the vectors of $\mathcal{L}$ belonging to the γn balls is in the sparser lattice $\mathcal{L}_{p,\mathbf{z}}$, except possibly those in $\{i \cdot (\mathbf{c}+\mathbf{w})\}_{i\in[\gamma n]}$. In the rest of the reduction analysis, we assume that we are in this situation and do not repeatedly state that this occurs with non-negligible probability.

As an illustration of Lemma 13, we include Figure 3. In the case of $\gamma = 1$ (left subfigure), there are several plain balls with radius $\lambda_1(\mathcal{L})$, centered in $\mathbf{t}$, $\mathbf{t}+\mathbf{w}$ and $2(\mathbf{t}+\mathbf{w})$. The dashed balls illustrate the distance between $i \cdot (\mathbf{c}+\mathbf{w})$ and $i \cdot (\mathbf{t}+\mathbf{w})$ for all $i \in [\gamma n]$. We can see that $\mathbf{c}+\mathbf{w}$ (within the dashed ball) is inside the plain ball, and $2 \cdot (\mathbf{c}+\mathbf{w})$ (within the dashed ball) is outside of its corresponding plain ball. Similarly, in the case of $\gamma = 2$ (right subfigure), vector $i \cdot (\mathbf{c}+\mathbf{w})$ is outside of its corresponding plain ball only when $i > 2$. Note in particular that in the case of $\gamma = 2$, vector $\mathbf{0}$ is not the closest point to the target vector, but belongs to the plain ball with center $\mathbf{t}$. Thus vector $\mathbf{0}$ should be removed via sparsification. As it is kept in any sparsified lattice, this is impossible to achieve. This illustrates why center $\mathbf{t}$ is shifted to a new point $\mathbf{t}+\mathbf{w}$ (then the shift $\mathbf{w}$ of $\mathbf{0}$ may be removed via sparsification). In both figures, the red crosses denote the points that are removed from the lattice via sparsification.

In Step 2 of the reduction, we construct a basis $\mathbf{B}'$ of an $(n+1)$-dimensional lattice $\mathcal{L}'$ by using Kannan's embedding technique. In Step 3, we call the $\mathrm{uSVP}_{\gamma(1+\varepsilon)}$ oracle with input basis $\mathbf{B}'$. The correctness of the reduction is provided by Lemmata 14, 15 and 16.

Any vector in lattice $\mathcal{L}'$ can be written as $\mathbf{b}' = ((\mathbf{b}+m(\mathbf{t}+\mathbf{w}))^{\mathrm{T}}, mkd_0)^{\mathrm{T}}$ with $\mathbf{b} \in \mathcal{L}_{p,\mathbf{z}}$ and $m \in \mathbb{Z}$. We claim that the vector $\mathbf{s}' = (((\mathbf{c}+\mathbf{w})-(\mathbf{t}+\mathbf{w}))^{\mathrm{T}}, -kd_0)^{\mathrm{T}} = ((\mathbf{c}-\mathbf{t})^{\mathrm{T}}, -kd_0)^{\mathrm{T}}$ is a shortest non-zero vector in $\mathcal{L}'$ and also that $\lambda_2(\mathcal{L}')/\lambda_1(\mathcal{L}') = \gamma(1+\Omega(1/n))$. Thus $\pm\mathbf{s}'$ will be output by the $\mathrm{uSVP}_{\gamma(1+\varepsilon)}$ oracle. We can then obtain the vector $\mathbf{c} = (\mathbf{c}+\mathbf{w})-(\mathbf{t}+\mathbf{w})+\mathbf{t} \in \mathcal{L}$. In the following, we give lower bounds for the norm of $\mathbf{b}' = ((\mathbf{b}+m(\mathbf{t}+\mathbf{w}))^{\mathrm{T}}, mkd_0)^{\mathrm{T}}$ not parallel to $\mathbf{s}'$, which depend on the value of m. Without loss of generality, we restrict ourselves to $m \geq 0$.

The following lemma is analogous to the '$m = 0$ case' of the Lyubashevsky-Micciancio reduction [15]. Note that the lower bound in the statement is essentially $2\gamma^2$.

▶ **Lemma 14.** *If $m = 0$ and $\mathbf{b}' \neq \mathbf{0}$, then $\|\mathbf{b}'\|^2/\|\mathbf{s}'\|^2 \geq 2\gamma^2/(1+1/(n-1)^2)$.*

Proof. As $m = 0$ and $\mathbf{b}' \neq \mathbf{0}$, we must have $\mathbf{b} \neq \mathbf{0}$. As a result, we have

$$\|\mathbf{b}'\|^2 \;=\; \|\mathbf{b}\|^2 \;\geq\; \lambda_1^2(\mathcal{L}) \;\geq\; \frac{2\gamma^2 d^2}{(1-\frac{1}{n})^2} \;\geq\; 2\gamma^2 d_0^2.$$

Thus, in this case, we have the gap

$$\frac{\|\mathbf{b}'\|^2}{\|\mathbf{s}'\|^2} \geq \frac{2\gamma^2 d_0^2}{d^2 + d_0^2 k^2} \geq \frac{2\gamma^2}{1+k^2} = \frac{2\gamma^2}{1+\frac{1}{(n-1)^2}}.$$

◀

The second lemma bounds the gap for small m's. It is where our improvement over prior reductions stems from. Note that the lower bound in the statement is essentially γ^2.

▶ **Lemma 15.** *If $m \leq \gamma n$ and $\mathbf{b}'$ is linearly independent with $\mathbf{s}'$, then $\|\mathbf{b}'\|^2/\|\mathbf{s}'\|^2 \geq (\gamma^2 + 1/n^2)/((1-1/n)^2 + 1/(n-1)^2)$.*

Proof. By Lemma 13, we have

$$\mathbf{c} + \mathbf{w} \;\in\; \mathcal{L}_{p,\mathbf{z}} \bigcap \bigcup_{i \leq \gamma n} \mathcal{B}\left(i \cdot (\mathbf{t}+\mathbf{w}), \frac{\lambda_1(\mathcal{L})}{\sqrt{2}}\right) \;\subseteq\; \mathbb{Z} \cdot (\mathbf{c}+\mathbf{w}).$$

Thus, as $\mathbf{b} \notin \mathbb{Z} \cdot (\mathbf{c}+\mathbf{w})$ (by assumption), we have

$$\|\mathbf{b}'\|^2 = \|\mathbf{b} - m \cdot (\mathbf{t}+\mathbf{w})\|^2 + m^2 d_0^2 k^2 \geq \frac{\lambda_1^2(\mathcal{L})}{2} + m^2 d_0^2 k^2 \geq \left(\frac{d\gamma}{1-\frac{1}{n}}\right)^2 + m^2 d_0^2 k^2.$$

Thus, in this case, we have the gap

$$\frac{\|\mathbf{b}'\|^2}{\|\mathbf{s}'\|^2} \geq \frac{(\frac{d\gamma}{1-\frac{1}{n}})^2 + m^2 d_0^2 k^2}{d^2 + d_0^2 k^2} \geq \frac{(\frac{d\gamma}{1-\frac{1}{n}})^2 + m^2 d^2 k^2}{d^2 + (\frac{d}{1-\frac{1}{n}})^2 k^2} = \frac{\gamma^2 + \frac{m^2}{n^2}}{(1-\frac{1}{n})^2 + \frac{1}{(n-1)^2}}.$$

The gap is an increasing function in m and hence it suffices to consider $m = 1$. ◀

The third lemma bounds the gap for larger m's. This corresponds to the 'large m case' of the Lyubashevsky-Micciancio reduction. As in the previous case, the lower bound in the statement is essentially γ^2.

▶ **Lemma 16.** *If $m > \gamma n$, then $\|\mathbf{b}'\|^2/\|\mathbf{s}'\|^2 \geq \gamma^2/((1-1/n)^2 + 1/(n-1)^2)$.*

Proof. For any $\mathbf{b} \in \mathcal{L}$, we have $\|\mathbf{b}'\|^2 \geq m^2 k^2 d_0^2$. Thus, in this case, we have the gap

$$\frac{\|\mathbf{b}'\|^2}{\|\mathbf{s}'\|^2} \geq \frac{m^2 k^2 d_0^2}{d^2 + k^2 d_0^2} \geq \frac{m^2 k^2 d^2}{d^2 + (\frac{d}{1-\frac{1}{n}})^2 k^2} = \frac{m^2}{(n-1)^2 + \frac{n}{(n-1)^2}}.$$

The gap is an increasing function in m and hence it suffices to consider the $m = \gamma n$. ◀

Now, we complete the proof of Theorem 12. According to Lemmata 14, 15 and 16, the uSVP gap satisfies, for large enough n

$$\frac{\lambda_2^2(\mathcal{L}')}{\lambda_1^2(\mathcal{L}')} \geq \min\left(\frac{2\gamma^2}{1+\frac{1}{(n-1)^2}}, \frac{\gamma^2+\frac{1}{n^2}}{(1-\frac{1}{n})^2+\frac{1}{(n-1)^2}}, \frac{\gamma^2}{(1-\frac{1}{n})^2+\frac{1}{(n-1)^2}}\right) \geq \gamma^2\left(1 + \Omega\left(\frac{1}{n}\right)\right).$$

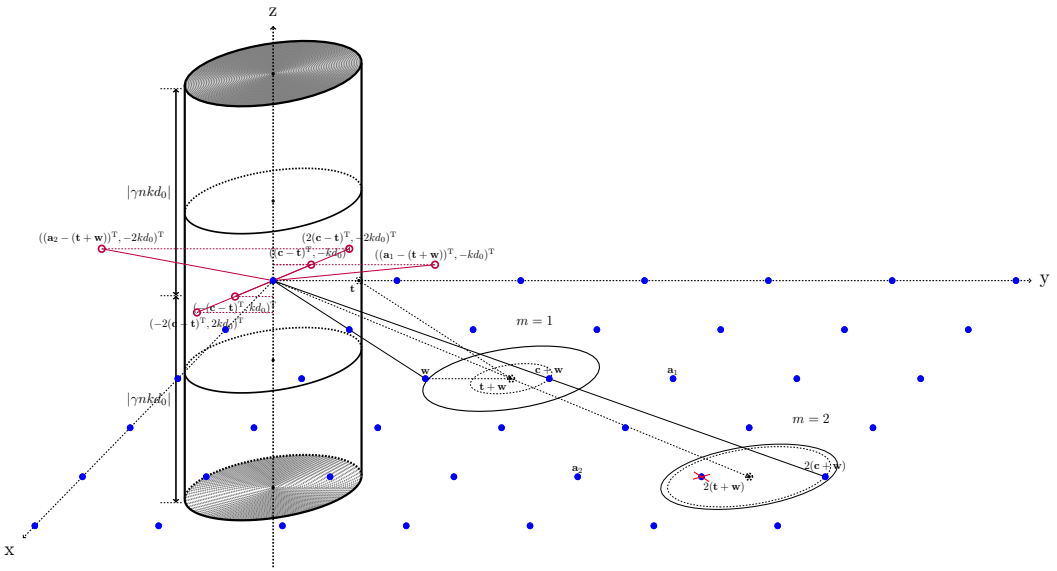


Figure 4 Geometric illustration of the reduction.

We include Figure 4 to geometrically illustrate the overall reduction. For convenience, we take $k = -1/(n-1)$ in the figure. We use filled dots to label points of 2-dimensional lattice $\mathcal{L}$, and hollow dots to label points of 3-dimensional lattice $\mathcal{L}'$ that are not in $\mathcal{L}$ (recall that $\mathcal{L} \subseteq \mathcal{L}'$). With Kannan's embedding technique, the offset between the vectors of $\mathcal{L}$ (e.g., $\mathbf{c}+\mathbf{w}$) and the shifted target $\mathbf{t}+\mathbf{w}$ are mapped to $\mathcal{L}'$ (e.g., $((\mathbf{c}-\mathbf{t})^{\mathrm{T}}, -kd_0)^{\mathrm{T}}$). Thanks to sparsification, all the points of $\mathcal{L}'$ belonging to the drawn cylinder (of height $|2\gamma nkd_0|$) are multiples of the shortest non-zero vector $\mathbf{s}' = ((\mathbf{c}-\mathbf{t})^{\mathrm{T}}, -kd_0)^{\mathrm{T}}$, e.g., $\pm((\mathbf{c}-\mathbf{t})^{\mathrm{T}}, -kd_0)^{\mathrm{T}}$ and $\pm(2(\mathbf{c}-\mathbf{t})^{\mathrm{T}}, -2kd_0)^{\mathrm{T}}$. All other points in $\mathcal{L}'$ that are linearly independent from $\mathbf{s}'$ lie outside of the cylinder, e.g., $((\mathbf{a}_1-(\mathbf{t}+\mathbf{w}))^{\mathrm{T}}, -kd_0)^{\mathrm{T}}$ and $((\mathbf{a}_2-(\mathbf{t}+\mathbf{w}))^{\mathrm{T}}, -2kd_0)^{\mathrm{T}}$. This cylinder forces the second minimum $\lambda_2(\mathcal{L}')$ to be large, and, more concretely, larger than $\gamma\lambda_1(\mathcal{L}')$. This corresponds to Lemma 15 (Lemma 14 handles the points of $\mathcal{L}$ and Lemma 16 handles the points of $\mathcal{L}'$ whose z-component is large).

Acknowledgments. We thank Steven Galbraith, Daniele Micciancio and Jinming Wen for helpful discussions.

References

1 M. Ajtai. The shortest vector problem in l_2 is NP-hard for randomized reductions (extended abstract). In *Proc. of STOC*, pages 284–293. ACM, 1998.
2 L. Babai. On Lovász lattice reduction and the nearest lattice point problem. *Combinatorica*, 6:1–13, 1986.
3 S. Bai and S. Galbraith. Private communication, 2015.
4 D. Dadush and G. Kun. Lattice sparsification and the approximate closest vector problem. In *Proc. of SODA*, pages 1088–1102. SIAM, 2013.
5 D. Dadush, O. Regev, and N. Stephens-Davidowitz. On the closest vector problem with a distance guarantee. In *Proc. of CCC*, pages 98–109. IEEE Computer Society Press, 2014.
6 R. de Buda. The upper error bound of a new near-optimal code. *IEEE Trans. on Information Theory*, 21(4):441–445, 1975.

7 U. Fincke and M. Pohst. A procedure for determining algebraic integers of given norm. In *Proc. of EUROCAL*, volume 162 of *LNCS*, pages 194–202, 1983.

8 R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. of STOC*, pages 99–108. ACM, 1983.

9 R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12(3):415–440, 1987.

10 S. Khot. Hardness of approximating the shortest vector problem in high L_p norms. In *Proc. of FOCS*, pages 290–297. IEEE Computer Society Press, 2003.

11 S. Khot. Hardness of approximating the shortest vector problem in lattices. *J. ACM*, 52(5):789–808, 2005.

12 A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann*, 261:515–534, 1982.

13 M. Liu, X. Wang, G. Xu, and X. Zheng. A note on BDD problems with λ_2-gap. *Inf. Process. Lett.*, 114(1-2):9–12, January 2014.

14 Y. K. Liu, V. Lyubashevsky, and D. Micciancio. On bounded distance decoding for general lattices. In *Proc. of RANDOM*, volume 4110 of *LNCS*, pages 450–461. Springer, 2006.

15 V. Lyubashevsky and D. Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In *Proc. of CRYPTO*, pages 577–594, 2009.

16 D. Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM J. Comput*, 30(6):2008–2035, 2001.

17 D. Micciancio. Private communication, 2015.

18 D. Micciancio and S. Goldwasser. *Complexity of Lattice problem: A Cryptography Perspective*. Kluwer, 2009.

19 O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.

20 C. P. Schnorr. A hierarchy of polynomial lattice basis reduction algorithms. *Theor. Comput. Science*, 53:201–224, 1987.

21 N. Stephens-Davidowitz. Discrete Gaussian sampling reduces to CVP and SVP. In *Proc. of SODA*, pages 1748–1764. SIAM, 2016.

22 A. Vardy. Algorithmic complexity in coding theory and the minimum distance problem. In *Proc. of STOC*, pages 92–109. ACM, 1997.

A Parallel Repetition Theorem for All Entangled Games*

Henry Yuen†

Massachusetts Institute of Technology, Cambridge, MA, USA
hyuen@mit.edu

Abstract

The behavior of games repeated in parallel, when played with quantumly entangled players, has received much attention in recent years. Quantum analogues of Raz's classical parallel repetition theorem have been proved for many special classes of games. However, for general entangled games no parallel repetition theorem was known.

We prove that the entangled value of a two-player game G repeated n times in parallel is at most $c_G n^{-1/4} \log n$ for a constant c_G depending on G, provided that the entangled value of G is less than 1. In particular, this gives the first proof that the entangled value of a parallel repeated game must converge to 0 for all games whose entangled value is less than 1. Central to our proof is a combination of both classical and quantum correlated sampling.

1998 ACM Subject Classification F.1.2. Modes of Computation

Keywords and phrases parallel repetition, direct product theorems, entangled games, quantum games

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.77

1 Introduction

A two-player one-round game G is played between a referee and two isolated players (who we will call Alice and Bob), who communicate only with the referee and not between themselves. The referee first samples a question pair (x, y) from some distribution μ and sends x to Alice and y to Bob. Alice and Bob respond with answers a and b respectively, and they win if $V(x, y, a, b) = 1$ for some predicate V.

The maximum winning probability of Alice and Bob in a game G is a quantity that depends on what resources they are allowed to use. If their answers are a deterministic function of their received question (and perhaps some public random string), then we call their maximum winning probability the *classical value* of G, denoted by $\mathrm{val}(G)$. However quantum mechanics allows Alice and Bob to share a resource called *entanglement*, which gives rise to correlations that cannot be reproduced with public randomness only. When Alice and Bob make use of entanglement to play a game G, we call their maximum winning probability the *entangled value* of G, denoted by $\mathrm{val}^*(G)$. For all games, the classical value is at most the entangled value. Cast in the language of games, the famous Bell's Theorem states that there exist games G where those values are different: $\mathrm{val}^*(G) > \mathrm{val}(G)$ [3].

The Parallel Repetition Question is the following natural and basic question: given a game G with value less than 1, what is the value of the game G^n, wherein Alice and Bob

* The full version of this paper can be found on the arXiv at http://arxiv.org/abs/1604.04340.
† This work was supported by Simons Foundation grant 360893 and National Science Foundation Grant 1218547.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 77; pp. 77:1–77:13


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

play n independent instances of G played *in parallel*? More formally, in the game G^n, the referee samples n independent question pairs $(x_1, y_1), \ldots, (x_n, y_n)$ from μ, and sends $(x_1, \ldots, x_n)$ to Alice, and sends $(y_1, \ldots, y_n)$ to Bob. Alice responds with answer tuple $(a_1, \ldots, a_n)$, Bob responds with $(b_1, \ldots, b_n)$, and the players win if for all coordinates $i \in [n]$, $V(x_i, y_i, a_i, b_i) = 1$.

The difficulty in relating $\mathrm{val}(G^n)$ with $\mathrm{val}(G)$ and n is that even though each of the n instances of G in G^n are independent, Alice and Bob need not play each instance independently. For example, since Alice receives $(x_1, \ldots, x_n)$ all at once, she can use some question x_j to answer the i'th game, and Bob can do something similar. Because of such strategies, for every k there are games G such that $\mathrm{val}(G^k) = \mathrm{val}(G) < 1$. This shows that the naive expectation that $\mathrm{val}(G^n) = \mathrm{val}(G)^n$ is false.

The naive expectation is not too far from the truth, however: Raz's Parallel Repetition Theorem [19] states that

$$\mathrm{val}(G^n) \leq (1 - (1 - \mathrm{val}(G))^3)^{c_G n},$$

where c_G is a constant depending on G. In particular, as n goes to infinity, the classical success probability goes to 0 exponentially fast in n (provided that $\mathrm{val}(G) < 1$). The proof is highly nontrivial, although it has been simplified and improved upon in recent years [12, 4]. Raz's Parallel Repetition Theorem has heavily influenced complexity theory, most notably in the areas of hardness of approximation [11] and communication complexity [13, 5].

One open question, which we call the Quantum Parallel Repetition Conjecture, asks whether an analogue of Raz's Parallel Repetition Theorem holds in the setting of entangled players. The Quantum Parallel Repetition Conjecture has been resolved for many special cases of games, including free games [6, 14, 7], projection games [9], XOR games [8], unique games [15], anchored games [1], and fortified games [2]. However, the general case has remained elusive. Not only do we not know of a quantum analogue of Raz's Parallel Repetition Theorem, it hasn't even been shown that if $\mathrm{val}^*(G) < 1$, then $\mathrm{val}^*(G^n)$ goes to 0 as n goes to infinity! Could quantum entanglement allow players to counteract the value-decreasing effect of parallel repetition?

In this paper we prove that for all nontrivial entangled games G (i.e. $\mathrm{val}^*(G) < 1$), the entangled value of G^n must converge to 0. This resolves a weaker version of the Quantum Parallel Repetition Conjecture for general games. Quantitatively, our result is the following:

▶ **Theorem 1** (Main Theorem). *Let G be a game involving two entangled players with $\mathrm{val}^*(G) = 1 - \varepsilon$. Then for all integer $n > 0$,*

$$\mathrm{val}^*(G^n) \leq c \cdot \frac{s_G \log n}{\varepsilon^{17} n^{1/4}}$$

where c is a universal constant and s_G is the bit-length of the players' answers in G.

This shows that the entangled value of G^n must decay at a polynomial rate with n. The full Quantum Parallel Repetition Conjecture states that the rate of decay is in fact exponential, and this remains an important open problem.

1.1 Previous work

There has been extensive work on the parallel repetition of entangled games. As stated earlier, past results have applied to various special classes of games, but there was no result that covered *all* games.

The results coming closest to the Quantum Parallel Repetition Conjecture are the work of Kempe and Vidick [16] and Bavarian, Vidick, and Yuen [1, 2]. Rather than proving parallel repetition theorems for general games, these works prove general *gap amplification* theorems, which are closely related. Instead of showing that for games G where $\mathrm{val}^*(G) < 1$ that $\mathrm{val}^*(G^n)$ goes to 0 with n, the game G is first converted to another game H where analyzing $\mathrm{val}^*(H^n)$ is much more tractable. Gap amplification is a technique used in complexity theory and cryptography to amplify the difference between two cases of a problem (usually called the *completeness* and *soundness* cases).

Kempe and Vidick showed that given an arbitrary game G, one can efficiently transform it to another game H with the following properties: if the *classical* value of G is 1 (meaning that there is a perfect deterministic strategy), then $\mathrm{val}(H^n) = 1$ (and thus $\mathrm{val}^*(H^n) = 1$). If the *entangled* value of G is less than 1, then the entangled value of H^n decays at a polynomial rate $n^{-\Omega(1)}$. In this tranformed game H, in addition to playing the game G, the referee will randomly choose to ask "consistency" questions to check that the players give the same answers on the same questions[1]. Thus [16] prove gap amplification for general games – with a caveat. Because of the random consistency checks in the game H, the "quantum completeness" is not preserved: even if $\mathrm{val}^*(G) = 1$, it is not necessarily the case that $\mathrm{val}^*(H) = 1$.

More recently, Bavarian, Vidick, and Yuen [1, 2] gave better gap amplification results for entangled games[2]. They showed that for general games G, one can apply a simple transformation to obtain another game H with the following properties:

1. If $\mathrm{val}^*(G) = 1$, then $\mathrm{val}^*(H^n) = 1$.
2. If $\mathrm{val}^*(G) < 1$, then $\mathrm{val}^*(H^n) \leq \exp(-\Omega(n))$.

Note that the transformation from G to H preserves quantum completeness, and that when $\mathrm{val}^*(G) < 1$, the entangled value of the repeated game decays *exponentially*. Like [16], the transformations of [1, 2] construct H by adding auxiliary questions to the game G. The transformation given in [1] is called *anchoring*, and the trasformation in [2] is called *fortification*. The latter transformation gives a quantum generalization of the fortification technique of [17] for *classical games*. The quantitative aspects of repeated anchored games are different from those of fortified games, but both yield general gap amplification theorems for entangled games.

The results of Bavarian, Vidick and Yuen show that, while we do not know if the Quantum Parallel Repetition Conjecture holds for all games G, we *do* know that it holds for a class of games that effectively captures the general case, in fact with exponential decay similar to Raz's theorem. Since the main application of parallel repetition in complexity theory and quantum information is gap amplification, the results of [1, 2] effectively settle the Quantum Parallel Repetition Conjecture – as far as applications are concerned.

But as a scientific question, the original Quantum Parallel Repetition Conjecture is a fundamental and basic problem about the power of entanglement in games. Prior to this work, one might have wondered whether there exists a game G such that $\mathrm{val}^*(G) < 1$, but there is some constant δ such that for infinitely many n there is a nefarious entangled strategy for G^n with success probability at least δ? Here we prove that this cannot happen.

1.2 Proof overview

Theorem 1 is proved via reduction: if $\mathrm{val}^*(G^n)$ is too large, then from an optimal entangled strategy for G^n we can construct an entangled strategy for the single-shot game G that wins

[1] This transformation is to due to Feige and Kilian [10], who proved a similar result for classical games.
[2] They also obtain general gap amplification results for games with more than two players.

with probability strictly greater than $\mathrm{val}^*(G)$, which would be a contradiction.

In more detail, suppose that $\mathrm{val}^*(G) = 1 - \varepsilon$. If the success probability of the players in G^n is dramatically larger than our target bound (which in our case is $\sim n^{-O(1)}$), then we can identify a set of coordinates $C \subseteq [n]$ that is not too large, but has the property that for a uniformly random coordinate $i \in [n] - C$,

$$\Pr(\text{Win game } i \mid \text{Win games in } C) > 1 - \varepsilon/2 \tag{1}$$

where here the probability is both over the randomness of the questions in G^n, the randomness of the players' entangled strategy, and the randomly chosen index i. Thus it would be advantageous if Alice and Bob could play the single-shot game G by "embedding" it in a randomly chosen ith coordinate of G^n, and playing G^n *conditioned* on the event that the games indexed by C have been won. If they could do this, then by (1), the probability they win the ith coordinate of G^n, and hence the original game G, is at least $1 - \varepsilon/2 > \mathrm{val}^*(G)$, which would be a contradiction.

If the players are classical (i.e. use deterministic strategies), this embedding is performed in the following way. Alice and Bob are first given questions (X_i, Y_i) for the i'th game. Based on their received question, Alice and Bob jointly sample a *dependency-breaking variable* R. The essential features of this dependency-breaking variable are:

1. **Usefulness:**[3] $\mathsf{P}_{A_iB_i|RX_iY_iW_C} = \mathsf{P}_{A_i|RX_iW_C} \cdot \mathsf{P}_{B_i|RY_iW_C}$
2. **Sampleability:** $\mathsf{P}_{R|X_iY_iW} \approx \mathsf{P}_{R|X_iW_C} \approx \mathsf{P}_{R|Y_iW_C}$

where "$\approx$" means closeness in statistical distance. Here, W_C denotes the event that the players win all the games in C. $\mathsf{P}_{A_iB_i|RX_iY_iW_C}$ denotes the probability distribution of Alice's and Bob's answers in the ith coordinate when playing G^n, conditioned on the dependency-breaking variable R, their received questions for the ith game (X_i, Y_i), *and* the event W_C. The "Usefulness property" states that, the players' answers in the ith round are independent of each other, conditioned on R, their own questions, and W_C. Thus, given R distributed according to $\mathsf{P}_{R|X_iY_iW_C}$, Alice can sample A_i on her own, because she possesses R and X_i, and similarly Bob can sample B_i on his own, because he possesses knowledge of R and Y_i. By (1), the probability that $V(X_i, Y_i, A_i, B_i) = 1$ will be strictly greater than $\mathrm{val}^*(G)$, wherein we would arrive at a contradiction.

As the name suggests, the "sampleability property" implies that Alice and Bob can (approximately) jointly sample the variable R. Even though the distribution $\mathsf{P}_{R|X_iY_iW_C}$ may depend on both players' questions, the sampleability property shows R, up to some error, only depends on X_i or Y_i, but not both. Using the *correlated sampling procedure* of [12], Alice and Bob can jointly sample R from $\mathsf{P}_{R|X_iY_iW}$ with high probability.

At a high level, the proof of our quantum parallel repetition theorem is similar. However instead of sampling a dependency-breaking variable R, the players will need to sample a *dependency-breaking state*. It is an entangled state $|\Psi_{x_iy_i}\rangle$ that depends on both Alice's and Bob's questions (x_i, y_i), and satisfies similar Usefulness and Sampleability properties:

1. **Usefulness:** The distribution of measurement outcomes by making local measurements on $|\Psi_{X_iY_i}\rangle$ is equal to $\mathsf{P}_{A_iB_i|X_iY_iW_C}$.
2. **Sampleability:** There exist states $|\Phi_{X_i}\rangle$ and $|\Gamma_{Y_i}\rangle$ such that $|\Psi_{X_iY_i}\rangle \approx |\Phi_{X_i}\rangle \approx |\Gamma_{Y_i}\rangle$

where "$\approx$" means closeness in ℓ_2 distance, and the statements hold on average over X_iY_i.

[3] We will let P denote the probability distribution that describes the joint distribution of the random variables relevant in an execution of the strategy for G^n, including the players' questions $X_1, \ldots, X_n, Y_1, \ldots, Y_n$, the players' answers $A_1, \ldots, A_n, B_1, \ldots, B_n$, and the dependency-breaking variable R.

The Usefulness property states that if on input (x_i, y_i), Alice and Bob were to share the entangled state $|\Psi_{x_i y_i}\rangle$, then they could make local measurements to obtain outcomes distributed according to $\mathsf{P}_{A_i B_i | X_i Y_i W_C}$, which would mean that their success probability would be $\Pr(\text{Win } i \mid \text{Win } C)$, which is greater than $\text{val}^*(G)$, an impossibility.

The Sampleability property implies that on input (x_i, y_i) Alice and Bob are actually able to approximately prepare the state $|\Psi_{x_i y_i}\rangle$. This is because of the *quantum correlated sampling procedure* of Dinur, Steurer, and Vidick, who used it to prove a parallel repetition theorem for entangled projection games [9]. It is entirely analogous to Holenstein's correlated sampling procedure: Alice has a description of a state $|\Phi_{X_i}\rangle$ that's close to $|\Psi_{X_i Y_i}\rangle$, and Bob has a description of a state $|\Gamma_{Y_i}\rangle$ that is also close to $|\Psi_{X_i Y_i}\rangle$. Via local transformations on preshared quantum entanglement, Alice and Bob can generate an approximation of $|\Psi_{X_i Y_i}\rangle$. Combined with the Usefulness property, Alice and Bob are then able to win the ith game with too high probability.

It is not difficult to define states that satisfy the Usefulness property. Consider an execution of the entangled strategy for G^n. In the beginning, the players share some entangled state $|\psi\rangle$, and upon obtaining questions $(x_1, \ldots, x_n)$ and $(y_1, \ldots, y_n)$, the players apply local measurements depending on these questions to $|\psi\rangle$ to obtain answer tuples $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$. One can define an ensemble of states $\{|\Psi_{x_i, y_i}\rangle\}$ that are, roughly speaking, derived from the post-measurement state of the players *conditioned* on the players having won all the games in C (that is, conditioned on the event W_C), *and* having received a specific question pair (x_i, y_i) in the i'th coordinate. Such an ensemble of states would satisfy the Usefulness property.

However, the primary challenge is achieving Sampleability property, that is, to show the states $|\Psi_{x_i, y_i}\rangle$ only depend on one player's question, but not both. One major obstacle to proving the Sampleability property is the following: in the players' strategy for G^n, Bob (say) may elect to "print" his entire vector of questions $(y_1, \ldots, y_n)$ into the entangled state $|\psi\rangle$. He can do this by applying a local unitary operation controlled on his questions on some ancilla qubits in $|\psi\rangle$. We cannot say he does not do this, because the shared entangled state $|\psi\rangle$ and the players' measurements are completely arbitrary. But this implies that we cannot hope to prove that the post-measurement state is independent of y_i, conditioned on x_i.

Despite such barriers, we are able to define the $|\Psi_{x_i, y_i}\rangle$ in such a way that removes such adversarial dependencies on the players' questions. Assuming (for contradiction) that the players' probability of success is at least $n^{-O(1)}$, then we are able to prove that these states satisfy the Sampleability property. We build upon many previous works: we use the information theoretic framework of [6, 14], carefully combined with the operator analysis techniques from [9]. The definition of the dependency-breaking states $|\Psi_{x_i, y_i}\rangle$ includes the classical dependency-breaking variables of [12] used to prove Raz's parallel repetition theorem. Our final constructed strategy for the single-shot game G uses both classical and quantum correlated sampling procedures.

2 Preliminaries

2.1 Probability distributions

We largely adopt the notational conventions from [12] for probability distributions. We let capital letters denote random variables and lower case letters denote specific samples. We will use subscripted sets to denote tuples, e.g., $X_{[n]} := (X_1, \ldots, X_n)$, $x_{[n]} = (x_1, \ldots, x_n)$, and if $C \subset [n]$ is some subset then X_C will denote the sub-tuple of $X_{[n]}$ indexed by C. We use P_X to denote the probability distribution of random variable X, and $\mathsf{P}_X(x)$ to

denote the probability that $X = x$ for some value x. For multiple random variables, e.g., X, Y, Z, $\mathsf{P}_{XYZ}(x, y, z)$ denotes their joint distribution with respect to some probability space understood from context.

We use $\mathsf{P}_{Y|X=x}(y)$ to denote the conditional distribution $\mathsf{P}_{YX}(y, x)/\mathsf{P}_X(x)$, which is defined when $\mathsf{P}_X(x) > 0$. When conditioning on many variables, we usually use the shorthand $\mathsf{P}_{X|y,z}$ to denote the distribution $\mathsf{P}_{X|Y=y,Z=z}$. For example, we write $\mathsf{P}_{V|\omega_{-i},x_i,y_i}$ to denote $\mathsf{P}_{V|\Omega_{-i}=\omega_{-i},X_i=x_i,Y_i=y_i}$. For an event W we let $\mathsf{P}_{XY|W}$ denote the distribution conditioned on W. We use the notation $\mathbb{E}_X f(x)$ and $\mathbb{E}_{\mathsf{P}_X} f(x)$ to denote the expectation $\sum_x \mathsf{P}_X(x) f(x)$.

Let P_{X_0} be a distribution of $\mathcal{X}$, and for every x in the support of P_{X_0}, let $\mathsf{P}_{Y|X_1=x}$ be a conditional distribution defined over $\mathcal{Y}$. We define the distribution $\mathsf{P}_{X_0}\mathsf{P}_{Y|X_1}$ over $\mathcal{X} \times \mathcal{Y}$ as

$$(\mathsf{P}_{X_0}\mathsf{P}_{Y|X_1})(x, y) := \mathsf{P}_{X_0}(x) \cdot \mathsf{P}_{Y|X_1=x}(y).$$

Additionally, we write $\mathsf{P}_{X_0Z}\mathsf{P}_{Y|X_1}$ to denote the distribution $(\mathsf{P}_{X_0Z}\mathsf{P}_{Y|X_1})(x, z, y) := \mathsf{P}_{X_0Z}(x, z) \cdot \mathsf{P}_{Y|X_1=x}(y)$.

For two random variables X_0 and X_1 over the same set $\mathcal{X}$, we use

$$\|\mathsf{P}_{X_0} - \mathsf{P}_{X_1}\| := \frac{1}{2} \sum_{x \in \mathcal{X}} |\mathsf{P}_{X_0}(x) - \mathsf{P}_{X_1}(x)|,$$

to denote the total variation distance between P_{X_0} and P_{X_1}.

2.2 Quantum information theory

For comprehensive references on quantum information we refer the reader to [18, 21].

For a vector $|\psi\rangle$, we use $\||\psi\rangle\|$ to denote its Euclidean length. For a matrix A, we will use $\|A\|_1$ to denote its *trace norm* $\mathrm{Tr}(\sqrt{AA^\dagger})$, and $\|A\|_F$ to denote its *Frobenius norm* $\sqrt{\mathrm{Tr}(AA^\dagger)}$. A density matrix is a positive semidefinite matrix with trace 1. The *fidelity* between two density matrices ρ and σ is defined as $F(\rho, \sigma) = \|\sqrt{\rho}\sqrt{\sigma}\|_1$. For Hermitian matrices A, B we write $A \preceq B$ to indicate that $A - B$ is positive semidefinite. We use $\mathbb{I}$ to denote the identity matrix. A *positive operator valued measurement* (POVM) with outcome set $\mathcal{A}$ is a set of positive semidefinite matrices $\{E^a\}$ labeled by $a \in \mathcal{A}$ that sum to the identity.

We will use the convention that, when $|\psi\rangle$ is a pure state, ψ refers to the rank-1 density matrix $|\psi\rangle\langle\psi|$. We use subscripts to denote system labels; so ρ_{AB} will denote the density matrix on the systems A and B. A *classical-quantum* state ρ_{XE} is classical on X and quantum on E if it can be written as $\rho_{XE} = \sum_x p(x)|x\rangle\langle x|_X \otimes \rho_{E|X=x}$ for some probability measure $p(\cdot)$. The state $\rho_{E|X=x}$ is by definition the E part of the state ρ_{XE}, conditioned on the classical register $X = x$. We write $\rho_{XE|X=x}$ to denote the state $|x\rangle\langle x|_X \otimes \rho_{E|X=x}$. We often write expressions such as $\rho_{E|x}$ as shorthand for $\rho_{E|X=x}$ when it is clear from context which registers are being conditioned on. This will be useful when there are many classical variables to be conditioned on.

2.3 Classical and quantum correlated sampling

Correlated sampling is a key component of Holenstein's proof of the classical parallel repetition theorem.

▶ **Lemma 2** (Classical correlated sampling [12]). *Let* P *and* Q *be two probability distributions over a universe* $\mathcal{U}$ *such that* $\|\mathsf{P} - \mathsf{Q}\|_1 \leq \varepsilon < 1$*. Then there exists a zero communication two-player protocol using shared randomness where the first player outputs an element* $p \in \mathcal{U}$ *distributed according to* P*, the second player samples an element* $q \in \mathcal{U}$ *distributed according to* Q*, and with probability at least* $1 - O(\varepsilon)$*, the two elements are identical (i.e.* $p = q$*).*

We call the protocol in the Lemma above the *classical correlated sampling procedure*. The next lemma is the quantum extension of the correlated sampling lemma, proved by [9] in order to obtain a parallel repetition theorem for entangled projection games, a class of two-player games. Their lemma is a robust version of the quantum state embezzlement procedure of [20].

▶ **Lemma 3** (Quantum correlated sampling [9]). *Let d be an integer. Then there exists an integer d' and a collection of unitaries V_ψ, W_ψ acting on $\mathbb{C}^{dd'}$ for every state $|\psi\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d$, such that the following holds: for any two states $|\varphi\rangle, |\theta\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d$,*

$$\|\overline{V}_\varphi \otimes W_\theta |E_{dd'}\rangle - |\varphi\rangle|E_{d'}\rangle\| \leq O(\| \, |\varphi\rangle - |\theta\rangle \, \|^{1/6})$$

where $|E_d\rangle \propto \sum_{j=1}^d \frac{1}{\sqrt{j}}|j\rangle|j\rangle$ is the d-dimensional embezzlement state.

We shall call the protocol in the Lemma above the *quantum correlated sampling procedure*.

3 Proof of the Main Theorem

Let G be a two-player one-round game with question distribution μ and referee predicate $V(x, y, a, b)$. Let $\mathcal{A}$ and $\mathcal{B}$ denote the alphabets of Alice's and Bob's answers, respectively. Let $\mathrm{val}^*(G) = 1 - \varepsilon$.

Consider an optimal entangled strategy for G^n, which consists of a shared entangled state $|\psi\rangle^{E_A E_B} \in \mathbb{C}^d \otimes \mathbb{C}^d$ and measurement POVMs for Alice and Bob, $\{A^{a_{[n]}}_{x_{[n]}}\}$ and $\{B^{b_{[n]}}_{y_{[n]}}\}$ respectively. We will assume that $|\psi\rangle$ is symmetric; i.e., $|\psi\rangle = \sum_i \sqrt{\lambda_i}|v_i\rangle|v_i\rangle$ for some orthonormal basis $\{|v_i\rangle\}$. This is without loss of generality, as we can always rotate (say) Bob's basis vectors to match Alice's basis vectors, and fold the unitary rotation into Bob's measurements. For $i \in [n]$, let W_i denote the event that the players win coordinate i using this optimal strategy. Let $W = W_1 \wedge \cdots \wedge W_n$ denote the event that the players win all coordinates. For a set $C \subseteq [n]$, let $W_C = \wedge_{i \in C} W_i$.

▶ **Proposition 4.** *Suppose that $\log 1/\Pr(W) \leq \varepsilon n/16 - \log 4/\varepsilon$. Then there exists a set $C \subseteq [n]$ of size at most $t = \frac{8}{\varepsilon}(\log 4/\varepsilon + \log 1/\Pr(W))$ such that*

$$\Pr_{i \notin C}(W_i|W_C) \geq 1 - \varepsilon/2.$$

where i is chosen uniformly from $[n] - C$.

Proof. Set $\delta = \varepsilon/8$. Let $W_{>1-\delta}$ denote the event that the players won more than $(1-\delta)n$ rounds. To show existence of such a set C, we will show that $\mathbb{E}_C \Pr(\neg W_i|W_C) \leq \varepsilon/2$, where C is a (multi)set of t independently chosen indices in $[n]$. This implies that there exists a particular set C such that $\Pr(\neg W_i|W_C) \leq \varepsilon/2$, which concludes the claim.

First we write, for a fixed C, $\Pr(\neg W_i|W_C) = \Pr(\neg W_i|W_C, W_{>1-\delta}) \Pr(W_{>1-\delta}|W_C) + \Pr(\neg W_i|W_C, \neg W_{>1-\delta}) \Pr(\neg W_{>1-\delta}|W_C)$. Observe that $\Pr(\neg W_i|W_C \wedge W_{>1-\delta})$ is the probability that, conditioned on winning all rounds in C, the randomly selected coordinate $i \in [n]-C$ happens to be one of the (at most) δn lost rounds. This is at most $\delta n/(n-t) \leq \varepsilon/4$, where we use our assumption on t from the Proposition statement. Now observe that

$$\mathop{\mathbb{E}}_C \Pr(\neg W_{>1-\delta}|W_C) \leq \mathop{\mathbb{E}}_C \frac{\Pr(W_C|\neg W_{>1-\delta})}{\Pr(W_C)} \leq \frac{1}{\Pr(W)}(1-\delta)^t \leq \varepsilon/4$$

where in the second line we used the fact that $\Pr(W_C) \geq \Pr(W)$. ◀

For the rest of the proof we will fix a set C given by Proposition 4.

3.1 Dependency-breaking variables

We introduce the random variables that play an important role in the proof of Theorem 1. Let $C \subseteq [n]$ be as given by Proposition 4. We fix $C = \{m+1, m+2, \ldots, n\}$, where $m = n - |C|$, as this will easily be seen to hold without loss of generality. Let $(X_{[n]}, Y_{[n]})$ be distributed according to $\mu_{[n]}$ and $(A_{[n]}, B_{[n]})$ be defined from $X_{[n]}$ and $Y_{[n]}$ as follows:

$$\mathsf{P}_{A_{[n]}B_{[n]}|x_{[n]},y_{[n]}}(a_{[n]}, b_{[n]}) = \langle\psi| A^{a_{[n]}}_{x_{[n]}} \otimes B^{b_{[n]}}_{y_{[n]}} |\psi\rangle.$$

Let (X_C, Y_C) and $\mathrm{Z} = (A_C, B_C)$ be random variables that denote the players' questions and answers respectively associated with the coordinates indexed by C.

We use the random variables Ω and R that are crucially used in Holenstein's proof of Raz's parallel repetition theorem. Let $D_1, \ldots, D_m$ be independent and uniformly distributed in $\{Alice, Bob\}$. Let $M_1, \ldots, M_m$ be independent random variables defined in the following way: for each $i \in [m]$,

$$M_i = \begin{cases} X_i & \text{if } D_i = Alice \\ Y_i & \text{if } D_i = Bob \end{cases}$$

Now for $i \in [m]$, we define $\Omega_i := (D_i, M_i)$. We say that Ω_i *fixes Alice's input* if $D_i = Alice$, and otherwise Ω_i fixes Bob's input. We write Ω to denote the random variable $(\Omega_1, \ldots, \Omega_m, X_C, Y_C)$, where $X_C Y_C$ are Alice and Bob's questions in the coordinates indexed by C. For $i \in [m]$ we write Ω_{-i} to denote the random variable Ω with Ω_i omitted.

▶ **Proposition 5.** *Conditioned on Ω, $X_{[n]}$ and $Y_{[n]}$ are independent.*

Finally, we will define a *dependency-breaking variable* $R := (\Omega, A_C, B_C)$, where A_C and B_C are the players' answers in the coordinates indexed by C. For $i \notin C$, we let $R_{-i} := (\Omega_{-i}, A_C, B_C)$. R_i will refer to Ω_i. We will use lowercase letters to denote instantiations of these random variables: e.g., r_{-i}, x_i, and y_i refer to specific values of R_{-i}, X_i, and Y_i.

Throughout our proofs, all expectations are implicitly over the measure defined by P. For example, the expectation $\mathbb{E}_{\Omega_{-i}Z|x_i,y_i}$ indicates $\sum_{\omega_{-i},a_C,b_C} \mathsf{P}_{\Omega_{-i}A_CB_C|x_i,y_i}(\omega_{-i}, a_C, b_C)$. Given an event such as W (winning all the coordinates) or W_C (winning all the coordinates in C), $\mathsf{P}(W)$ and $\mathsf{P}(W_C)$ will mean the probability of these events with respect to the distribution P.

The following Lemma expresses the idea that, because W_C is an event that occurs with not-too-small probability, conditioning on it cannot skew the distribution of variables corresponding to an average coordinate by too much. This Lemma follows in a straightforward manner from the [12].

▶ **Lemma 6.** *The following statements hold on, average over i chosen uniformly in $[m]$:*
1. $\mathbb{E}_i \left\| \mathsf{P}_{R_i X_i Y_i | W_C} - \mathsf{P}_{R_i X_i Y_i} \right\|_1 \leq O(\sqrt{\delta})$
2. $\mathbb{E}_i \left\| \mathsf{P}_{X_i Y_i R_{-i} | W_C} - \mathsf{P}_{X_i Y_i} \cdot \mathsf{P}_{R_{-i}|X_i W_C} \right\|_1 \leq O(\sqrt{\delta})$
3. $\mathbb{E}_i \left\| \mathsf{P}_{X_i Y_i R_{-i} | W_C} - \mathsf{P}_{X_i Y_i} \cdot \mathsf{P}_{R_{-i}|Y_i W_C} \right\|_1 \leq O(\sqrt{\delta})$

where $\delta := \frac{1}{m}\left(\log 1/\mathsf{P}(W_C) + |C| \log |\mathcal{A}||\mathcal{B}|\right)$.

3.2 Two key Lemmas, and proof of the Main Theorem

For every $i \in [n] - C$, we will construct a collection of bipartite states $\{|\Psi_{r_{-i},x_i,y_i}\rangle\} \subseteq \mathbb{C}^d \otimes \mathbb{C}^d$, which we call dependency-breaking states, that are indexed by the dependency-breaking variable r_{-i} defined above, and questions (x_i, y_i). The following lemmas state the important properties of this collection of states:

▶ **Lemma 7** (Usefulness Lemma). *For all r_{-i}, x_i, y_i, there exist POVMs $\{\widehat{A}^{a_i}_{r_{-i},x_i}\}$ and $\{\widehat{B}^{b_i}_{r_{-i},y_i}\}$ acting on $\mathbb{C}^d$ such that*

$$\mathsf{P}_{A_iB_i|r_{-i},x_i,y_i}(a_i,b_i) = \mathrm{Tr}\left(\widehat{A}^{a_i}_{r_{-i},x_i} \otimes \widehat{B}^{b_i}_{r_{-i},y_i} \Psi_{r_{-i},x_i,y_i}\right).$$

▶ **Lemma 8** (Sampleability Lemma). *There exists an integer $d' \geq d$ such that for every i, r_{-i}, x_i, y_i, there exist local unitaries $U_{r_{-i},x_i}, V_{r_{-i},y_i}$ acting on $\mathbb{C}^{d'}$ such that*

$$\mathop{\mathbb{E}}_{i} \mathop{\mathbb{E}}_{X_iY_i} \left[\mathop{\mathbb{E}}_{R_{-i}|x_i,y_i,W_C} \left\| U_{r_{-i},x_i} \otimes V_{r_{-i},y_i} |E_{dd'}\rangle - |\Psi_{r_{-i},x_i,y_i}\rangle |E_{d'}\rangle \right\| \right] \leq O((\delta^{1/4}/\mathsf{P}(W_C))^{1/12})$$

where $|E_{dd'}\rangle$ and $|E_{d'}\rangle$ are dd' and d'-dimensional embezzlement states, respectively, and δ is defined to be $\frac{1}{m}(\log 1/\mathsf{P}(W_C) + |C| \log |\mathcal{A}||\mathcal{B}|)$.

Lemma 7 shows that the states $|\Psi_{r_{-i},x_i,y_i}\rangle$ are *useful* to have; they allow Alice and Bob to produce answers in the i'th coordinate whose statistics are consistent with the dependency-breaking variable r_{-i} and their inputs (x_i, y_i). Lemma 8 shows that these states are *locally generatable* by Alice and Bob, when given joint access to preshared entanglement, the dependency-breaking variable r_{-i} and their own inputs x_i and y_i respectively.

Using these two Lemmas we can prove the Main Theorem.

Proof of the Main Theorem. Consider the following strategy for the game G. Alice and Bob share beforehand the embezzlement state $|E_{dd'}\rangle$ of dimension dd' given by Lemma 8, and they also have access to shared randomness. Given inputs (x_i, y_i) distributed according to $\mathsf{P}_{X_iY_i} = \mu$:

1. Alice and Bob jointly sample a uniformly random $i \in [n] - C$.
2. Alice and Bob jointly, approximately sample R_{-i} from $\mathsf{P}_{R_{-i}|x_i,y_i,W_C}$ using the classical correlated sampling procedure.
3. Alice applies U_{r_{-i},x_i} to her side of $|E_{dd'}\rangle$
4. Bob applies V_{r_{-i},y_i} to his side of $|E_{dd'}\rangle$
5. Alice measures her side of the entanglement using $\{\widehat{A}^{a_i}_{r_{-i},x_i}\}$ and outputs the outcome a_i
6. Bob measures his side of the entanglement using $\{\widehat{B}^{b_i}_{r_{-i},y_i}\}$ and outputs the outcome b_i

We now analyze the success probability of this strategy. We will use $\widetilde{\mathsf{P}}$ to denote the distribution of variables in the probability space associated with an execution of this strategy. For example, we will write $\widetilde{\mathsf{P}}_{R_{-i}|X_iY_i}$ to denote the distribution of R_{-i} conditioned on X_iY_i that is sampled in Step 1. From Lemma 6 we have that on average over i, $\mathsf{P}_{X_iY_iR_{-i}|W_C} \approx \mathsf{P}_{X_iY_i} \cdot \mathsf{P}_{R_{-i}|X_iW_C} \approx \mathsf{P}_{X_iY_i} \cdot \mathsf{P}_{R_{-i}|Y_iW_C}$, where "$\approx$" means closeness in statistical distance. By invoking the classical correlated sampling procedure of Lemma 2, we get

$$\mathop{\mathbb{E}}_{i} \|\mathsf{P}_{X_iY_i} \cdot \widetilde{\mathsf{P}}_{R_{-i}|X_iY_i} - \mathsf{P}_{X_iY_iR_{-i}|W_C}\|_1 \leq O(\sqrt{\delta}).$$

After Step 3, Alice and Bob will possess a state $|\Lambda_{r_{-i},x_i,y_i}\rangle$ such that

$$\mathop{\mathbb{E}}_{i} \mathop{\mathbb{E}}_{X_iY_i} \left[\mathop{\mathbb{E}}_{R_{-i}|x_i,y_i,W_C} \|\Lambda_{r_{-i},x_i,y_i} - \Psi_{r_{-i},x_i,y_i}\|_1 \right] \leq \eta$$

where $\eta = O((\delta^{1/4}/\mathsf{P}(W_C))^{1/12})$. Consider the measurement process in Steps 4 and 5. Let $\widetilde{\mathsf{P}}_{A_iB_i|r_{-i},x_i,y_i}$ denote the distribution of measurement outcomes in this strategy, conditioned

on their inputs and a sampled value of r_{-i}. By Lemma 7 and the fact that the trace norm is nonincreasing under quantum operations, we have that

$$\mathop{\mathbb{E}}_{i} \mathop{\mathbb{E}}_{X_iY_i} \left[\mathop{\mathbb{E}}_{R_{-i}|x_i,y_i,W_C} \|\widetilde{\mathsf{P}}_{A_iB_i|x_i,y_i,r_{-i}} - \mathsf{P}_{A_iB_i|x_i,y_i,r_{-i}}\|_1 \right] \leq \eta$$

or equivalently

$$\mathop{\mathbb{E}}_{i} \|\mathsf{P}_{X_iY_i} \cdot \widetilde{\mathsf{P}}_{R_{-i}|X_iY_iW_C} \cdot \widetilde{\mathsf{P}}_{A_iB_i|x_i,y_i,r_{-i}} - \mathsf{P}_{X_iY_i} \cdot \mathsf{P}_{R_{-i}|X_iY_iW_C} \cdot \mathsf{P}_{A_iB_iR_{-i}|X_iY_iW_C}\|_1 \leq \eta.$$

By Lemma 6 we have $\mathbb{E}_i \|\mathsf{P}_{X_iY_i|W_C} - \mathsf{P}_{X_iY_i}\| \leq \sqrt{\delta}$. By triangle inequality and that $\widetilde{\mathsf{P}}_{X_iY_i} = \mathsf{P}_{X_iY_i}$, we have

$$\mathop{\mathbb{E}}_{i} \|\widetilde{\mathsf{P}}_{X_iY_iR_{-i}A_iB_i} - \mathsf{P}_{X_iY_iR_{-i}A_iB_i|W_C}\|_1 \leq O(\eta).$$

Note that $\widetilde{\mathsf{P}}_{X_iY_iR_{-i}A_iB_i}$ represents the probability distribution of all the variables present in the strategy above. Let W_i denote the probability the players win the ith coordinate. Thus we get

$$\mathop{\mathbb{E}}_{i} |\widetilde{\mathsf{P}}(W_i) - \mathsf{P}(W_i|W_C)| \leq O(\eta). \tag{2}$$

Assume that

$$\mathsf{P}(W) \geq \frac{cs \log n}{\varepsilon^{17} n^{1/4}}$$

where $c > 0$ is a universal constant, and s is the bit-length of the players' answers. Since $\mathsf{P}(W_C) \geq \mathsf{P}(W)$, and using our bound on $|C|$ (from Proposition 4) and our bound on δ (from Lemma 6), this implies that the right hand side of (2) is at most $\varepsilon/4$ (for an appropriate choice of c). This implies that

$$\mathop{\mathbb{E}}_{i} \widetilde{\mathsf{P}}(W_i) \geq \mathop{\mathbb{E}}_{i} \mathsf{P}(W_i|W_C) - \varepsilon/4 \geq 1 - \varepsilon/2 - \varepsilon/4 > \mathrm{val}^*(G)$$

where in the second line we used the bound from Proposition 4. However, this implies that there exists an i such that $\widetilde{\mathsf{P}}(W_i) > \mathrm{val}^*(G)$, which is a contradiction. Therefore $\mathsf{P}(W) \leq \frac{cs \log n}{\varepsilon^{17} n^{1/4}}$. ◀

Now we turn to defining the states and operators promised in the two key lemmas above, as well as giving an intuition for them.

3.3 Quantum states and operators

In this subsection we define the states $|\Psi_{r_{-i},x_i,y_i}\rangle$ and measurement operators $\{\widehat{A}^{a_i}_{r_{-i},x_i}\}$ and $\{\widehat{B}^{b_i}_{r_{-i},y_i}\}$. Recall that the dependency-breaking variable R consists of the set of fixed questions $\Omega = (X_C, Y_C, \Omega_1, \ldots, \Omega_m)$ and fixed answers $Z = (A_C, B_C)$ for the coordinates in C.

Coarse-grained measurements. We first *coarsen* the measurement POVMs $\{A^{a_{[n]}}_{x_{[n]}}\}$ and $\{B^{b_{[n]}}_{y_{[n]}}\}$ that constitute Alice and Bob's strategy in G^n to construct a set of *intermediate measurements*, which essentially produce answers for the games in set C, conditioned on a setting of Ω.

Fix i, ω, a_C, b_C, x_i, y_i. Define $A^{a_C}_{\omega_{-i},x_i} = \sum_{a_{[n]}|a_C} \mathbb{E}_{X_{[n]}|\omega_{-i},x_i} A^{a_{[n]}}_{x_{[n]}}$, and $B^{b_C}_{\omega_{-i},y_i} = \sum_{b_{[n]}|b_C} \mathbb{E}_{Y_{[n]}|\omega_{-i},y_i} B^{b_{[n]}}_{y_{[n]}}$ where $a_{[n]}|a_C$ (resp. $b_{[n]}|b_C$) indicates summing over all tuples $a_{[n]}$ consistent with the suffix a_C (resp. $b_{[n]}$ consistent with suffix b_C) and recall that $\mathbb{E}_{X_{[n]}|\omega_{-i},x_i}$ is shorthand for $\sum_{x_{[n]}} \mathsf{P}_{X_{[n]}|\Omega_{-i}=\omega_{-i},X_i=x_i}(x_{[n]})$. We also define $A^{a_C}_{\omega} = \mathbb{E}_{X_{[n]}|\omega} A^{a_C}_{x_{[n]}}$ and $B^{b_C}_{\omega} = \mathbb{E}_{Y_{[n]}|\omega} B^{b_C}_{y_{[n]}}$.

Let ρ denote the reduced density matrix of $|\psi\rangle$ on Alice's side. Since we have assumed that $|\psi\rangle$ is symmetric, ρ is also the reduced density matrix on Bob's side. For all i, ω, x_i, y_i, a_C, b_C, let U_{ω_{-i},x_i,a_C}, U_{ω,a_C}, V_{ω_{-i},y_i,b_C}, and V_{ω,b_C} be unitaries such that

$$U_{\omega_{-i},x_i,a_C}(A^{a_C}_{\omega_{-i},x_i})^{1/2}\sqrt{\rho} \qquad V_{\omega_{-i},y_i,b_C}(B^{b_C}_{\omega_{-i},y_i})^{1/2}\sqrt{\rho}$$
$$U_{\omega,a_C}(A^{a_C}_{\omega})^{1/2}\sqrt{\rho} \qquad V_{\omega,b_C}(B^{b_C}_{\omega})^{1/2}\sqrt{\rho}$$

are positive semidefinite. Such unitaries can be found via singular value decompositions. For notational convenience, let

$$S_{\omega_{-i},x_i,a_C} = U_{\omega_{-i},x_i,a_C}(A^{a_C}_{\omega_{-i},x_i})^{1/2} \qquad T_{\omega_{-i},y_i,b_C} = V_{\omega_{-i},y_i,b_C}(B^{b_C}_{\omega_{-i},y_i})^{1/2}$$
$$S_{\omega,a_C} = U_{\omega,a_C}(A^{a_C}_{\omega})^{1/2} \qquad T_{\omega,b_C} = V_{\omega,b_C}(B^{b_C}_{\omega})^{1/2}$$

Fine-grained measurements. Now we can define the *fine-grained measurements* that Alice and Bob can apply to obtain answers for the i'th game. Define

$$\widehat{A}^{a_i}_{r_{-i},x_i} = S^{-1}_{\omega_{-i},x_i,a_C} A^{a_C,a_i}_{\omega_{-i},x_i} S^{-1}_{\omega_{-i},x_i,a_C} \qquad \widehat{B}^{b_i}_{r_{-i},y_i} = T^{-1}_{\omega_{-i},y_i,b_C} B^{b_C,b_i}_{\omega_{-i},y_i} T^{-1}_{\omega_{-i},y_i,b_C}$$

where

$$A^{a_C,a_i}_{\omega_{-i},x_i} = \sum_{a_{[n]}|a_C,a_i} \mathbb{E}_{X_{[n]}|\omega_{-i},x_i} A^{a_{[n]}}_{x_{[n]}} \qquad B^{b_C,b_i}_{\omega_{-i},y_i} = \sum_{b_{[n]}|b_C,b_i} \mathbb{E}_{Y_{[n]}|\omega_{-i},y_i} B^{b_{[n]}}_{y_{[n]}}$$

and $a_{[n]}|a_C,a_i$ (resp. $b_{[n]}|b_C,b_i$) denotes summing over all $a_{[n]}$ consistent with a_C and a_i (resp. all $b_{[n]}$ consistent with b_C and b_i). It is easy to verify that the sets $\{\widehat{A}^{a_i}_{r_{-i},x_i}\}_{a_i\in\mathcal{A}}$ and $\{\widehat{B}^{b_i}_{r_{-i},y_i}\}_{b_i\in\mathcal{B}}$ form POVMs. Here, for a square matrix A, A^{-1} denotes its generalized inverse.

States. Now we are ready to define the states. Fix i, $r_{-i} = (\omega_{-i}, a_C, b_C)$, and x_i, y_i. Then let

$$|\Psi_{r_{-i},x_i,y_i}\rangle = \frac{S_{\omega_{-i},a_C,x_i} \otimes T_{\omega_{-i},b_C,y_i}|\psi\rangle}{\left\|S_{\omega_{-i},a_C,x_i} \otimes T_{\omega_{-i},b_C,y_i}|\psi\rangle\right\|}.$$

Observe that the normalization $\left\|S_{\omega_{-i},a_C,x_i} \otimes T_{\omega_{-i},b_C,y_i}|\psi\rangle\right\|^2$ is equal to $\mathsf{P}_{A_C B_C|\omega_{-i},x_i,y_i}(a_C, b_C)$.

3.4 Proof of Usefulness Lemma (Lemma 7)

This Lemma follows from a simple calculation: for every $x_i, y_i, a_i, b_i, r_{-i}$:

$$\begin{aligned}
&\mathrm{Tr}\left(\widehat{A}^{a_i}_{r_{-i},x_i} \otimes \widehat{B}^{b_i}_{r_{-i},y_i} \Psi_{r_{-i},x_i,y_i}\right) \\
&= \frac{1}{\left\| S_{\omega_{-i},a_C,x_i} \otimes T_{\omega_{-i},b_C,y_i} |\psi\rangle \right\|^2} \mathrm{Tr}\left(A^{a_C,a_i}_{\omega_{-i},x_i} \otimes B^{b_C,b_i}_{\omega_{-i},y_i} |\psi\rangle\langle\psi|\right) \\
&= \frac{1}{\mathsf{P}_{A_C B_C|\omega_{-i},x_i,y_i}(a_C,b_C)} \sum_{a_{[n]}|a_C,a_i} \sum_{b_{[n]}|b_C,b_i} \mathop{\mathbb{E}}_{X_{[n]}Y_{[n]}|\omega_{-i},x_i,y_i} \mathrm{Tr}\left(A^{a_{[n]}}_{x_{[n]}} \otimes B^{b_{[n]}}_{y_{[n]}} |\psi\rangle\langle\psi|\right) \\
&= \frac{\mathsf{P}_{A_i B_i A_C B_C|\omega_{-i},x_i,y_i}(a_i,b_i,a_C,b_C)}{\mathsf{P}_{A_C B_C|\omega_{-i},x_i,y_i}(a_C,b_C)} \\
&= \mathsf{P}_{A_i B_i|r_{-i},x_i,y_i}(a_i,b_i).
\end{aligned}$$

In the second equality we used that conditioned on Ω, $X_{[n]}$ and $Y_{[n]}$ are independent, so therefore $\mathbb{E}_{X_{[n]}|\omega_{-i},x_i} \mathbb{E}_{Y_{[n]}|\omega_{-i},y_i} = \mathbb{E}_{X_{[n]}Y_{[n]}|\omega_{-i},x_i,y_i}$. In the last equality we used that $r_{-i} = (\omega_{-i}, a_C, b_C)$. This concludes the Usefulness Lemma.

3.5 Proof of the Sampleability Lemma (Lemma 8)

Overview. Here we give some intuition. We first analyze an ensemble of states $\{|\Gamma_{x_i,x_C,a_C}\rangle\}$ (for now we omit mention of the dependency-breaking variable R for simplicity). These are indexed by Alice's questions in the i'th coordinate, her questions in the C coordinates, as well as her answers in the C coordinates. The state $|\Gamma_{x_i,x_C,a_C}\rangle$ roughly represents the state of the players where only Alice has applied her measurements – Bob hasn't done anything yet.

Fix a y_i, x_C, a_C. For average x_i, x_i' that are independently sampled from the marginal distribution $\mathsf{P}_{X_i|Y_i=y_i}$, we will show that $\||\Gamma_{x_i,x_C,a_C}\rangle - |\Gamma_{x_i',x_C,a_C}\rangle\| \sim \frac{1}{n}$. To handle issues such as Alice "printing" her input onto the state $|\psi\rangle$ (as discussed in the introduction), the definition of $|\Gamma_{x_i,x_C,a_C}\rangle$ requires local unitaries that "undo" such overt actions of Alice and Bob – this is accomplished by the unitaries U and V defined in Section 3.3.

Then, we consider what happens when we apply Bob's measurement to both states $|\Gamma_{x_i,x_C,a_C}\rangle$ and $|\Gamma_{x_i',x_C,a_C}\rangle$, and condition on obtaining answers b_C for the C coordinates. His measurement will depend on the questions y_i and y_C. The post-measurement states will be precisely $|\Psi_{x_i,y_i,x_C,y_C,a_C,b_C}\rangle$ and $|\Psi_{x_i',y_i,x_C,y_C,a_C,b_C}\rangle$. The distance between these states will be, roughly speaking, the distance between $|\Gamma_{x_i,x_C,a_C}\rangle$ and $|\Gamma_{x_i',x_C,a_C}\rangle$ divided by the probability of Bob obtaining outcome b_C conditioned on Alice obtaining a_C. If we average this distance over all choices of x_C, y_C, a_C, b_C that imply the event W_C, we get that the average distance between $|\Psi_{x_i,y_i,x_C,y_C,a_C,b_C}\rangle$ and $|\Psi_{x_i',y_i,x_C,y_C,a_C,b_C}\rangle$ is approximately $\frac{1}{n\mathsf{P}(W_C)}$. If $\mathsf{P}(W)$ is much greater than $1/n$, then this distance is small. We then invoke quantum correlated sampling (Lemma 3), and that proves the Sampleability Lemma.

Because of space constraints, we omit the proof and refer the reader to the full version at `http://arxiv.org/abs/1604.04340`.

Acknowledgments. The author thanks both the Institute of Mathematical Sciences at the National University of Singapore, and the Weizmann Institute of Science for hospitable stays during which this research was conducted. The author also thanks Corinna Li, Mohammad Bavarian, Govind Ramnarayan, and anonymous referees for helpful feedback and discussions.

References

1 Mohammad Bavarian, Thomas Vidick, and Henry Yuen. Anchoring games for parallel repetition. *arXiv preprint arXiv:1509.07466*, 2015.

2 Mohammad Bavarian, Thomas Vidick, and Henry Yuen. Parallel repetition via fortification: analytic view and the quantum case. *arXiv preprint arXiv:1603.05349*, 2016.

3 John S Bell. On the Einstein-Podolsky-Rosen paradox. *Physics*, 1(3), 1964.

4 Mark Braverman and Ankit Garg. Small value parallel repetition for general games. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC)*, 2015.

5 Mark Braverman, Anup Rao, Omri Weinstein, and Amir Yehudayoff. Direct products in communication complexity. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 746–755. IEEE, 2013.

6 André Chailloux and Giannicola Scarpa. Parallel repetition of entangled games with exponential decay via the superposed information cost. In *Automata, Languages, and Programming*, pages 296–307. Springer, 2014.

7 Kai-Min Chung, Xiaodi Wu, and Henry Yuen. Parallel repetition for entangled k-player games via fast quantum search. In *the 30th Conference on Computational Complexity (CCC)*, pages 512–536, 2015.

8 Richard Cleve, William Slofstra, Falk Unger, and Sarvagya Upadhyay. Perfect parallel repetition theorem for quantum xor proof systems. *Computational Complexity*, 17(2):282–299, 2008.

9 Irit Dinur, David Steurer, and Thomas Vidick. A parallel repetition theorem for entangled projection games. In *the 29th Conference on Computational Complexity (CCC)*, pages 197–208, 2014.

10 Uriel Feige and Joe Kilian. Two-prover protocols – low error at affordable rates. *SIAM Journal on Computing*, 30(1):324–346, 2000.

11 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4), 2001.

12 Thomas Holenstein. Parallel repetition: Simplification and the no-signaling case. *Theory of Computing*, 5(8):141–172, 2009. doi:10.4086/toc.2009.v005a008.

13 Rahul Jain. New strong direct product results in communication complexity. *Electronic Colloquium on Computational Complexity (ECCC)*, 18(24):2, 2011.

14 Rahul Jain, Attila Pereszlényi, and Penghui Yao. A parallel repetition theorem for entangled two-player one-round games under product distributions. In *Proceedings of 29th Conference on Computational Complexity (CCC)*, pages 209–216, 2014.

15 Julia Kempe, Oded Regev, and Ben Toner. Unique games with entangled provers are easy. In *Proceedings of Foundations of Computer Science (FOCS)*, 2008.

16 Julia Kempe and Thomas Vidick. Parallel repetition of entangled games. In *Proceedings of the forty-third annual ACM symposium on Theory of computing (STOC)*, pages 353–362, 2011.

17 Dana Moshkovitz. Parallel repetition from fortification. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 414–423. IEEE, 2014.

18 Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information*. Cambridge university press, 2010.

19 Ran Raz. A parallel repetition theorem. *SIAM Journal on Computing*, 27(3):763–803, 1998.

20 Wim van Dam and Patrick Hayden. Universal entanglement transformations without communication. *Physical Review A*, 67(6):060302, 2003.

21 Mark M Wilde. *Quantum information theory*. Cambridge University Press, 2013.

Tight Sum-Of-Squares Lower Bounds for Binary Polynomial Optimization Problems*

Adam Kurpisz[1], Samuli Leppänen[2], and Monaldo Mastrolilli[3]

1 **IDSIA, Manno, Switzerland**
adam@idsia.ch
2 **IDSIA, Manno, Switzerland**
samuli@idsia.ch
3 **IDSIA, Manno, Switzerland**
monaldo@idsia.ch

Abstract

We give two results concerning the power of the Sum-Of-Squares(SoS)/Lasserre hierarchy. For binary polynomial optimization problems of degree $2d$ and an odd number of variables n, we prove that $(n+2d-1)/2$ levels of the SoS/Lasserre hierarchy are necessary to provide the exact optimal value. This matches the recent upper bound result by Sakaue, Takeda, Kim and Ito.

Additionally, we study a conjecture by Laurent, who considered the linear representation of a set with no integral points. She showed that the Sherali-Adams hierarchy requires n levels to detect the empty integer hull, and conjectured that the SoS/Lasserre rank for the same problem is $n-1$. We disprove this conjecture and derive lower and upper bounds for the rank.

1998 ACM Subject Classification Optimization, Convex programming, Integer programming

Keywords and phrases SoS/Lasserre hierarchy, lift and project methods, binary polynomial optimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.78

1 Introduction

In this paper we are concerned with the unconstrained *binary polynomial optimization problems* (BPOP):

$$\min_{x\in\{0,1\}^n} f(x)$$

where $f(x)$ is a multivariate polynomial. Many basic optimization problems are special cases of this general problem. Prominent examples include the MaxCut problem and the boolean Max k-CSP. For these problems the polynomials have at most degree 2 and k, respectively.

The Sum-of-Squares (SoS)/Lasserre hierarchy of semidefinite (SDP) relaxations [14, 21] is one of the most studied solution methods for general polynomial optimization problems (POP) including BPOP. The hierarchy is parameterized by a parameter t called the *relaxation level* and larger levels correspond to tighter relaxations. At level t, the relaxation consists of $n^{O(t)}$ variables and constraints, and it is thus solvable in time $n^{O(t)}$ using for example the ellipsoid method. At level n the SOS hierarchy finds the exact optimal value of an arbitrary constrained BPOP (but not a general POP).

* Supported by the Swiss National Science Foundation project 200020-144491/1 "Approximation Algorithms for Machine Scheduling Through Theory and Experiments".

For *quadratic* BPOP, Laurent [16] conjectured that at level $\lceil \frac{n}{2} \rceil$ the relaxation provides the exact optimal value. She also provided a matching lower bound showing that $\lfloor \frac{n}{2} \rfloor$ levels are not enough for finding the integer cut polytope of the complete graph with n nodes, when n is odd (the result was preceded by a similar lower bound by Grigoriev [9] for the KNAPSACK problem). The conjecture was proved by Fawzi, Saunderson and Parrilo [7] while showing that $\lceil \frac{n}{2} \rceil$ rounds are enough to exactly solve *any* unconstrained BPOP of degree 2. Very recently, Sakaue, Takeda, Kim and Ito [22] extended the result of [7] and showed that the SoS hierarchy requires at most $\lceil (n+r-1)/2 \rceil$ rounds to find the exact optimal value of an unconstrained BPOP of degree r with n variables. Note that the two upper bounds [7, 22] coincide when n is odd and $r = 2$, whereas for even n there is a difference of 1 (although [22] show also that if the optimized polynomial consists of only even degree monomials, the bound reduces to $\lceil (n+r-2)/2 \rceil$, matching the bound of [7] for example for the MAXCUT problem for every n). Furthermore, Sakaue et al. [22] numerically confirmed that for some degrees their bound is tight for certain instances of unconstrained BPOPs with 8 variables.

In a recent breakthrough Lee, Raghavendra and Steurer [17] proved that for the class of MAX-CSPs the SoS relaxation yields the "optimal" SDP approximation, meaning that SDPs of polynomial-size are equivalent in power to those arising from $O(1)$ rounds of the SoS relaxations. This result implies that known lower bound for SoS SDP relaxations translates to corresponding lower bounds on the size of any SDP formulations. With this aim, they build on the work of Grigoriev/Laurent [9, 16] to show that, for odd n, any sum of squares of degree $\lfloor n/2 \rfloor$ polynomials has ℓ_1-error at least $2^{n-2}/\sqrt{n}$ in approximating the following quadratic function

$$f(x) = (\|x\|_1 - \lfloor n/2 \rfloor)(\|x\|_1 - \lfloor n/2 \rfloor - 1) \tag{1}$$

This result is shown to imply lower bounds on the semidefinite extension complexity of the *correlation polytope* (which is isomorphic to the cut polytope and sometimes also called boolean quadric polytope). By reduction, the latter in turn implies exponential lower bounds for the integer cut, TSP and stable set polytopes. In [18] Lee, Prakash, de Wolf and Yuen proved that these lower bounds cannot be improved by showing better ℓ_1-approximations of $f(x)$.

Our Results

In this paper we give two results concerning the power of the SoS hierarchy. Our first result shows that the bound given by Sakaue et al. [22] is tight for polynomials with even degree and an odd number of variables. More precisely, we consider BPOPs of the form $\min_{x \in \{0,1\}^n} f_d(x)$ where $f_d(x)$ is a degree $2d$ (for $d \geq 1$) polynomial defined as follows:

$$f_d(x) = (\|x\|_1 - \lfloor n/2 \rfloor + d - 1)^{\underline{2d}} \tag{2}$$

where $k^{\underline{r}} = k(k-1)\cdots(k-r+1)$ denotes the falling factorial. For $d = 1$ we have $f_1(x) = f(x)$, where $f(x)$ is the polynomial defined in (1) and considered in [17, 18]. We show that for odd $n = 2m+1$, the SoS relaxation allows negative values for polynomial $f_d(x)$ that is non-negative over $\{0,1\}^n$, even at level $\lceil \frac{n+2d-1}{2} \rceil - 1 = m + d - 1$.

Our second result concerns comparing the SoS hierarchy to other lift and project methods. A commonly used benchmark for comparing hierarchies is to find the smallest level at which they find the convex hull of a given set of integral points P,[1] usually given as an intersection

[1] The smallest such level is called the *rank* of P, and it is always smaller or equal to n for the usually studied hierarchies.

of the set $\{0,1\}^n$ and a polytope. Examples of such results include [8, 9, 10, 11, 16, 19, 23]. In [15], Laurent shows that the Sherali-Adams hierarchy detects that the set

$$K = \{0,1\}^n \cap \left\{ x \in [0,1]^n \mid \sum_{r \in R} x_r + \sum_{r \in R \setminus N} (1 - x_r) \geq \frac{1}{2} \text{ for all } R \subseteq N \right\} \tag{3}$$

is empty only after n levels. She then conjectures the SoS rank of K is $n-1$. The polytope K has been used earlier to show that n iterations are needed also for the following procedures: the Lovász-Schrijver N_+ operator (with positive semidefiniteness) [8], the Lovász-Schrijver N_+ operator combined with taking Chvátal cuts [4], and the N_+ operator combined with taking Gomory mixed integer cuts (equivalent to disjunctive cuts) [5]. In this paper we disprove Laurent's conjecture, and show that indeed the SoS rank of K is bounded between $\Omega(\sqrt{n})$ and $n - \Omega(n^{1/3})$.

Interestingly, Au [1] and the authors of this paper [12] independently considered the rank of a variation of the set K where on the right hand side of the inequalities there is an exponentially small constant instead of $\frac{1}{2}$. Both works show that the rank of the modified K is exactly n.

In our proofs we demonstrate the use of a recent theorem of the authors [13] that simplifies the positive semidefiniteness (PSD) condition of the SoS hierarchy when the problem formulation is highly symmetric (as noted in [18], Blekherman [3] has also obtained a similar result that is still in preparation).

Our first result is obtained by showing that a certain conical combination of solutions with non-integral relaxation value to the SoS relaxation for the function (1) gives a negative SoS relaxation value for the polynomials (2) of degree $2d$. Then, for the first and the second result, we apply the theorem in [13] to reduce the PSDness condition into showing that a particular inequality is satisfied for every polynomial with a certain form. Showing that the inequality is satisfied (lower bounds) or cannot be satisfied (upper bounds) then boils down to evaluating or approximating a certain combinatorial sum. Our results also answer the question in [18] regarding the applications of the theorem of [3, 13].

2 The Sum-of-Squares hierarchy

In this paper we consider the SoS hierarchy when applied to (i) unconstrained 0/1 polynomial optimization problems, and (ii) approximating the convex hull of the set

$$P = \{x \in \{0,1\}^n \mid g_\ell(x) \geq 0, \forall \ell \in [p]\} \tag{4}$$

where $g_\ell(x)$ are linear constraints and p a positive integer. The form of the SoS hierarchy we use in this paper is equivalent to the one used in literature (see e.g. [2, 14, 15]) and follows from applying a change of basis to the dual certificate of the refutation of the proof system (see [13] for the details on the change of basis and [20] for discussion on the connection to the proof system). We use this change of basis in order to obtain a useful decomposition of the moment matrices as a sum of rank one matrices of special kind.

For any $I \subseteq N = \{1, \ldots, n\}$, let x_I denote the 0/1 solution obtained by setting $x_i = 1$ for $i \in I$, and $x_i = 0$ for $i \in N \setminus I$. For a function $f : \{0,1\}^n \to \mathbb{R}$, we denote by $f(x_I)$ the value of the function evaluated at x_I. In the SoS hierarchy defined below there is a variable y_I^N that can be interpreted as the "relaxed" indicator variable for the solution x_I. We point out that in this formulation of the hierarchy the number of variables $\{y_I^N : I \subseteq N\}$ is exponential in n, but this is not a problem in our context since we are interested in proving lower and upper bounds rather than solving an optimization problem.

Let $\mathcal{P}_t(N)$ be the collection of subsets of N of size at most $t \in \mathbb{N}$. For every $I \subseteq N$, the q-zeta vector $Z_I \in \mathbb{R}^{\mathcal{P}_q(N)}$ is a 0/1 vector with J-th entry ($|J| \leq q$) equal to 1 if and only if $J \subseteq I$.[2] Note that $Z_I Z_I^\top$ is a rank one matrix and the matrices considered in Definitions 1 and 2 are linear combinations of these rank one matrices.

To simplify the presentation we define the SoS hierarchy separately for polynomial optimization problems and for the integer hull approximation.

▶ **Definition 1.** The t-th round SoS hierarchy relaxation of $\min_{x\in\{0,1\}^n} f(x)$, denoted by $\text{SoS}_t(f)$, is the optimization problem with variables $\{y_I^N \in \mathbb{R} : \forall I \subseteq N\}$ of the form

$$\min_{y^N \in \mathbb{R}^{2^n}} \sum_{I\subseteq N} y_I^N f(x_I) \tag{5}$$

$$\text{s.t.} \sum_{I\subseteq N} y_I^N = 1, \tag{6}$$

$$\sum_{I\subseteq N} y_I^N Z_I Z_I^\top \succeq 0, \text{ where } Z_I \in \mathbb{R}^{\mathcal{P}_t(N)} \tag{7}$$

▶ **Definition 2.** The t-th round SoS hierarchy relaxation for the set P as given in (4), denoted by $\text{SoS}_t(P)$, is the set of variables $\{y_I^N \in \mathbb{R} : \forall I \subseteq N\}$ that satisfy

$$\sum_{I\subseteq N} y_I^N = 1, \tag{8}$$

$$\sum_{I\subseteq N} y_I^N Z_I Z_I^\top \succeq 0, \text{ where } Z_I \in \mathbb{R}^{\mathcal{P}_{t+1}(N)} \tag{9}$$

$$\sum_{I\subseteq N} g_\ell(x_I) y_I^N Z_I Z_I^\top \succeq 0, \ \forall \ell \in [p], \text{ where } Z_I \in \mathbb{R}^{\mathcal{P}_t(N)} \tag{10}$$

It is straightforward to see that the SoS hierarchy formulation given in Definition 2 is a relaxation of the integral polytope. Indeed consider any feasible integral solution $x_I \in P$ and set $y_I^N = 1$ and the other variables to zero. This solution clearly satisfies (8) and (9) because the rank one matrix $Z_I Z_I^\top$ is positive semidefinite (PSD), and (10) since $x_I \in P$.

For a set $Q \subseteq [0,1]^n$, we define the projection from $\text{SoS}_t(Q)$ to $\mathbb{R}^n$ as $x_i = \sum_{i\in I\subseteq N} y_I^N$ for each $i \in \{1, ..., n\}$. The *SoS rank* of Q, $\rho(Q)$, is the smallest t such that $\text{SoS}_t(Q)$ projects exactly to the convex hull of $Q \cap \{0,1\}^n$.

2.1 Using symmetry to simplify the PSDness conditions

In this section we present a theorem given in [13] that can be used to simplify the PSDness conditions (7), (9) and (10) when the problem formulation is very symmetric. More precisely, the theorem can be applied whenever the solutions and constraints are symmetric in the sense that $w_I^N = w_J^N$ whenever $|I| = |J|$ where w_I^N is understood to denote either y_I^N or $g_\ell(x_I) y_I^N$. In what follows we denote by $\mathbb{R}[x]$ the ring of polynomials with real coefficients and by $\mathbb{R}[x]_d$ the polynomials in $\mathbb{R}[x]$ with degree less or equal to d.

[2] In order to keep the notation simple, we do not emphasize the parameter q as the dimension of the vectors should be clear from the context.

▶ **Theorem 3** ([13]). *For any $t \in \{1,\dots,n\}$, let $\mathcal{S}_t$ be the set of univariate polynomials $G_h(k) \in \mathbb{R}[k]$, for $h \in \{0,\dots,t\}$, that satisfy the following conditions:*

$$G_h(k) \in \mathbb{R}[k]_{2t} \tag{11}$$
$$G_h(k) = 0 \qquad \text{for } k \in \{0,\dots,h-1\} \cup \{n-h+1,\dots,n\}, \text{ when } h \geq 1 \tag{12}$$
$$G_h(k) \geq 0 \qquad \text{for } k \in [h-1, n-h+1] \tag{13}$$

For any fixed set of values $\{w_k^N \in \mathbb{R} : k = 0,\dots,n\}$, if the following holds

$$\sum_{k=h}^{n-h} \binom{n}{k} w_k^N G_h(k) \geq 0 \qquad \forall G_h(k) \in \mathcal{S}_t \tag{14}$$

then

$$\sum_{k=0}^{n} w_k^N \sum_{\substack{I \subseteq N \\ |I|=k}} Z_I Z_I^\top \succeq 0$$

where $Z_I \in \mathbb{R}^{\mathcal{P}_t(N)}$.

Note that polynomial $G_h(k)$ in (13) is nonnegative in a *real interval*, and in (12) it is zero over a *set of integers*. Moreover, constraints (14) are trivially satisfied for $h > \lfloor n/2 \rfloor$.

3 Tightness of the SoS upper bounds for unconstrained BPOPs

In [22] it is shown that the SoS hierarchy exactly solves any unconstrained BPOP of degree r with n variables after $\lceil \frac{n+r-1}{2} \rceil$ levels. We show that this bound is tight for certain values of n and r, by giving a polynomial of degree $r = 2d$ for $d \geq 1$ that is non-negative over the hypercube, and show that when $n = 2m+1$, $m \geq d$, the SoS relaxation of the corresponding BPOP attains a negative value at level $t = \lceil \frac{n+2d-1}{2} \rceil - 1 = m + d - 1$.

More precisely, we consider the degree $2d$ polynomial

$$f_d(x) = (\|x\|_1 + d - m - 1)^{\underline{2d}} \tag{15}$$

where $k^{\underline{r}} = k(k-1)\cdots(k-r+1)$ denotes the falling factorial and $\|x\|_1 = \sum_i x_i$. For the sake of convenience, we denote by $f_d(k)$ the univariate polynomial evaluated at any point x with $\sum_i x_i = k$. We obtain the following result

▶ **Theorem 4.** *For odd n, the SoS relaxation of minimizing f_d requires at least $\lceil \frac{n+2d-1}{2} \rceil$ levels to find the exact optimum.*

3.1 Proof of Theorem 4

The case $d = 1$

The polynomial $f_1(x)$ is connected to the MaxCut problem in the complete graph of $n = 2m+1$ vertices in the following way: Let $x \in \{0,1\}^n$ denote any partition of the vertices into two sets in the natural way. Then, the maximal cut is achieved whenever $\sum_i x_i$ is either m or $m+1$, and $m(m+1) - f_1(x)$ counts the edges in the cut. Therefore, the SoS hierarchy is not able to exactly solve the MaxCut problem if it allows for solutions with negative values of the objective function (5).

It is shown in [13] that[3]

$$y_I^N[\alpha] = (n+1)\binom{\alpha}{n+1}\frac{(-1)^{n-|I|}}{\alpha-|I|} \qquad \forall I \subseteq N \tag{16}$$

is a feasible solution to the SoS hierarchy (as given in Definition 1) at level $\lfloor\alpha\rfloor$ for any non-integer $0 < \alpha < \frac{n}{2}$. Since the value of the solution only depends on the size of the set I, we denote by $y_k^N[\alpha]$ any $y_I^N[\alpha]$ with $|I| = k$. As a consequence of the proof in [13] it follows that for any non-integer $0 < \alpha \leq n$, $\sum_{i=0}^n \binom{n}{k} y_k^N[\alpha] = 1$. Furthermore, it is shown that the objective function attains the value $\sum_{k=0}^n \binom{n}{k} y_k^N[\alpha] f_1(k) = f_1(\alpha)$ and that in particular for $\alpha = \frac{n}{2}$, $f_1(\alpha) = -\frac{1}{4}$ at level $t = m$. Next we generalize this approach to $f_d(x)$.

Polynomials of degree $2d$

Consider the following solution

$$z_k^N = (2d-2)!(n+1)\binom{\frac{n}{2}-d+1}{n+1}\frac{(-1)^{n-k}}{(\frac{n}{2}+d-1-k)^{\underline{2d-1}}} \qquad \forall k \in \{0,\ldots,n\} \tag{17}$$

We show that for this solution, the SoS hierarchy objective (5) attains a negative value (see Lemma 10) and (7) is satisfied. For convenience, we do not actually show that (6) is satisfied and in fact it is not. We show, however, that $\sum_{k=0}^n \binom{n}{k} z_k > 0$, which implies that with proper normalization also (6) can be satisfied (see Lemma 7).

First we prove that the solution z_k^N can be written as a conical combination of the solutions $y_k^N[\cdot]$ in (16). We begin with the following lemma about partial fraction decompositions.

▶ **Lemma 5.** *For any $b \in \mathbb{N}_+$ and $a \in \mathbb{R}$ the following identity holds*

$$\frac{1}{(x-a)^{\underline{b}}} = \sum_{i=0}^{b-1} \frac{(-1)^{b-1-i}}{i!(b-1-i)!}\frac{1}{(x-a-i)}\,.$$

Proof. It is known that given two polynomials $P(x)$ and $Q(x) = (x-a_1)(x-a_2)\cdots(x-a_n)$, where the a_i are distinct constants and $\deg P < n$, the rational polynomial $\frac{P(x)}{Q(x)}$ can be decomposed into

$$\frac{P(x)}{Q(x)} = \sum_{i=1}^n \frac{P(a_i)}{Q'(a_i)}\frac{1}{(x-a_i)}$$

where $Q'(x)$ is the derivative of $Q(x)$. In our case, since $P(x) = 1$ and $Q(x) = \prod_{i=0}^{b-1}(x-a-i)$, we get

$$\frac{1}{(x-a)^{\underline{b}}} = \sum_{i=0}^{b-1} \frac{1}{\prod_{j\neq i}(a+i-(a+j))}\frac{1}{(x-a-i)} = \sum_{i=0}^{b-1} \frac{(-1)^{b-1-i}}{i!(b-1-i)!}\frac{1}{(x-a-i)}\,. \qquad \blacktriangleleft$$

Now we can express the solution z_k^N as a conical combination of the solutions $y_k^N[\cdot]$.

[3] The same solution was earlier considered in different basis by [10, 16] for the Knapsack and MaxCut problems respectively to show that the SoS hierarchy does not exactly solve the aforementioned problems at level $\lfloor\frac{n}{2}\rfloor$.

▶ **Lemma 6.** *The solution* (17) *can be decomposed as a conical combination of* $y_k^N[\cdot]$*:*

$$z_k^N = \sum_{j=0}^{2d-2} a_j y_k^N[n/2+d-1-j] \qquad \forall k \in \{0,\dots,n\}$$

for positive

$$a_j = \binom{2d-2}{j} \frac{(\frac{n}{2}+d-1)^{\underline{j}}}{(\frac{n}{2}-d+1+j)^{\underline{j}}}\,.$$

Proof. By Lemma 5 we get that

$$\frac{1}{(\frac{n}{2}+d-1-k)^{\underline{2d-1}}} = \sum_{j=0}^{2d-2} \frac{(-1)^{2d-2-j}}{j!(2d-2-j)!} \cdot \frac{1}{(\frac{n}{2}+d-1-k-j)}$$

and by writing

$$\binom{\frac{n}{2}-d+1}{n+1} = \frac{(-\frac{n}{2}+d-2-j)^{\underline{2d-2-j}}}{(\frac{n}{2}+d-1-j)^{\underline{2d-2-j}}} \binom{\frac{n}{2}+d-1-j}{n+1}$$

and using raising factorial notation, $(-b)^{\underline{a}} = (-1)^a b^{\overline{a}}$, we get that

$$\begin{aligned} z_k^N &= \sum_{j=0}^{2d-2} \frac{(2d-2)!}{j!(2d-2-j)!} \frac{(-\frac{n}{2}+d-2-j)^{\underline{2d-2-j}}}{(\frac{n}{2}+d-1-j)^{\underline{2d-2-j}}} \\ &\quad \cdot (n+1)\binom{\frac{n}{2}+d-1-j}{n+1} \frac{(-1)^{2d-2-j+n-k}}{(\frac{n}{2}+d-1-k-j)} \\ &= \sum_{j=0}^{2d-2} \binom{2d-2}{j} \frac{(\frac{n}{2}-d+2+j)^{\overline{2d-2-j}}}{(\frac{n}{2}+d-1-j)^{\underline{2d-2-j}}} y_k^N\left[\frac{n}{2}+d-1-j\right] \\ &= \sum_{j=0}^{2d-2} \binom{2d-2}{j} \frac{(\frac{n}{2}+d-1)^{\underline{j}}}{(\frac{n}{2}-d+1+j)^{\underline{j}}} y_k^N\left[\frac{n}{2}+d-1-j\right] \end{aligned}$$

◀

▶ **Lemma 7.** *We have* $\sum_{k=0}^n \binom{n}{k} z_k^N > 0$ *for every odd* n*,* $n = 2m+1$*, and* $d \in [m]$*.*

Proof. The proof follows by recalling that for every $\alpha \in [0,n] \setminus \mathbb{Z}$, $\sum_{i=0}^n \binom{n}{k} y_k^N[\alpha] = 1$ and by the fact that all the coefficients in the decomposition in Lemma 6 are positive. ◀

Now we show that the solution (17) is a feasible solution for the SoS hierarchy at level $t = m+d-1$. The solution (17) is symmetric, and so by Theorem 3 (see (14)) is enough to prove that for $t = m+d-1$,

$$\sum_{k=0}^n \binom{n}{k} z_k^N G_h(k) \geq 0 \qquad \forall G_h(k) \in \mathcal{S}_t\,.$$

We first note that the solution (17) attains positive values for every integer $k \in \{m-d+1,\dots,m+d\}$. Indeed, for $k = m-d+1+p$ for $p = \{0,\dots,2d-1\}$, since

$$\left(\frac{n}{2}-d+1\right)^{\underline{n+1}} = \left(\frac{n}{2}-d+1\right)^{\underline{m-d+2}} \left(\frac{1}{2}\right)^{\overline{m+d}} (-1)^{m+d}$$

and for $0 \leq p \leq 2d-1$

$$\left(\frac{n}{2}+d-1-k\right)^{\underline{2d-1}} = \left(2d-\frac{3}{2}-p\right)^{\underline{2d-1-p}} \left(\frac{1}{2}\right)^{\overline{p}} (-1)^p$$

the only, not obviously, non-negative part of z_k^N is

$$\frac{(-1)^{m+d}(-1)^{m+d-p}}{(-1)^p}$$

which is always positive. Thus the above (see (14)) is always satisfied whenever $h \geq m-d+1$ by the definition of the polynomials $G_h \in \mathcal{S}_t$.

It follows that it is enough to prove that the above is satisfied for $h \leq m-d$ which is implied if the following is true

$$\sum_{k=0}^{n} \binom{n}{k} z_k^N P(k) \geq 0$$

for every polynomial $P(x) \in \mathbb{R}[x]_{2t}$ that is nonnegative in the interval $[m-d+1, m+d]$.

▶ **Lemma 8.** *For any polynomial $P(x) \in \mathbb{R}[x]_{2(m+d-1)}$ we have*

$$\sum_{k=0}^{n} \binom{n}{k} z_k^N P(k) = \sum_{j=0}^{2d-2} a_j P\left(\frac{n}{2}+d-1-j\right).$$

Proof. Let $g(k) = (\frac{n}{2}+d-1-k)^{\underline{2d-1}}$ be the polynomial of degree $2d-1$ that corresponds to the denominator in polynomial in z_k^N (see (17)). By the polynomial remainder theorem, $P(k) = g(k)Q(k) + R(k)$, where the $Q(k)$ is the unique polynomial of degree at most $\deg(P) - \deg(g) \leq n-2$, and for the remainder it holds $R(r) = P(r)$ for all the roots r of polynomial $g(k)$. Then

$$\sum_{k=0}^{n} \binom{n}{k} z_k^N P(k) = \sum_{k=0}^{n} \binom{n}{k} z_k^N g(k)Q(k) + \sum_{k=0}^{n} \binom{n}{k} z_k^N R(k).$$

Here $\sum_{k=0}^{n} \binom{n}{k} z_k^N g(k)Q(k) = 0$, as $\sum_{k=0}^{n} (-1)^k \binom{n}{k} k^c = 0$ for every $c \leq n-1$. We remark here that if the level t is greater than $m+d-1$, then the polynomial Q can be of degree n or more and this reasoning fails.

By Lemma 6 we can write the sum with the remainder polynomial $R(k)$ as

$$\sum_{j=0}^{2d-2} a_j \sum_{k=0}^{n} \binom{n}{k} y_k^N \left[\frac{n}{2}+d-1-j\right] R(k)$$

and, again by the polynomial reminder theorem, for every $j \in \{0, \ldots, 2d-2\}$, $R(k) = (\frac{n}{2}+d-1-j-k)S_j(k) + R(\frac{n}{2}+d-1-j)$ and as before, since the degree of R is less or equal to $2d-2$, we have $\sum_{i=0}^{n} (-1)^k \binom{n}{k} S_j(k) = 0$. Thus, since $R(r) = P(r)$ for all the roots r of the polynomial $g(k)$, the above reduces to

$$\sum_{j=0}^{2d-2} a_j \left(R\left(\frac{n}{2}+d-1-j\right) \underbrace{\sum_{k=0}^{n} \binom{n}{k} y_k^N \left[\frac{n}{2}+d-1-j\right]}_{=1} \right)$$

$$= \sum_{j=0}^{2d-2} a_j P\left(\frac{n}{2}+d-1-j\right)$$

◀

By Lemma 8 we immediately obtain the following corollary.

▶ **Corollary 9.** *For any polynomial* $P(x) \in \mathbb{R}[x]_{2(m+d-1)}$ *such that* $P(x) \geq 0$ *for* $x \in [m-d+1, m+d]$ *we have*

$$\sum_{k=0}^{n} \binom{n}{k} z_k^N P(k) \geq 0\,.$$

Proof. By Lemma 8 we have that

$$\sum_{k=0}^{n} \binom{n}{k} z_k^N P(k) = \sum_{j=0}^{2d-2} a_k P\left(\frac{n}{2} + d - 1 - j\right)$$

which is positive since it is a conical combination of points at which polynomial P is positive. ◀

It remains to show that the objective value of the SoS hierarchy (5) attains a negative value.

▶ **Lemma 10.** *The sum* $\sum_{k=0}^{n} \binom{n}{k} z_k^N f_d(k)$ *is negative for every odd* $n = 2m+1$*, for any positive integer* m *and* $d \in \{1, ..., m\}$*.*

Proof. By Lemma 8, the solution z_k^N is such that

$$\sum_{k=0}^{n} \binom{n}{k} z_k^N f_d(k) = \sum_{j=0}^{2d-2} a_j f_d\left(\frac{n}{2} + d - 1 - j\right)\,.$$

Then, the claim is proved by showing that the following function $g(d,n)$ is negative for every odd $n = 2m+1$, for any positive integer m and $d \in [m]$. Formally, that

$$g(d,n) = \sum_{j=0}^{2d-2} \binom{2d-2}{j} \frac{(\frac{n}{2}+d-1)^{\underline{j}}}{(\frac{n}{2}-d+1+j)^{\underline{j}}} (2d-3/2-j)^{\underline{2d}} < 0$$

More precisely we show that the following identity holds (where !! denotes the double factorial).

$$g(d,n) = (2d-3/2)^{\underline{2d}} \cdot \frac{4^{d-1}(2d-2)!(2d-1)!!}{(d-1)!(4d-3)!!} \cdot \frac{(2m-2d+3)!!(m-1)!}{(m-d)!(2m+1)!!} \tag{18}$$

By simple inspection it is easy to see that (18) is negative and the claim follows.

We start by rewriting $g(d,n)$ by using the following (easy to check) identities:

$$\binom{2d-2}{j}\left(\frac{n}{2}+d-1\right)^{\underline{j}} = \frac{(2-2d)^{\overline{j}}\left(1-d-\frac{n}{2}\right)^{\overline{j}}}{j!}$$

$$\left(\frac{n}{2}-d+1+j\right)^{\underline{j}} = \left(2-d+\frac{n}{2}\right)^{\overline{j}}$$

$$(2d-3/2-j)^{\underline{2d}} = (2d-3/2)^{\underline{2d}} \frac{(3/2)^{\overline{j}}}{(3/2-2d)^{\overline{j}}}$$

By the above identities we have that

$$\begin{aligned} g(d,n) &= \sum_{j=0}^{2d-2} \binom{2d-2}{j} \frac{(\frac{n}{2}+d-1)^{\underline{j}}}{(\frac{n}{2}-d+1+j)^{\underline{j}}} (2d-3/2-j)^{\underline{2d}} \\ &= (2d-3/2)^{\underline{2d}} \sum_{j=0}^{2d-2} \frac{(2-2d)^{\overline{j}}\left(1-d-\frac{n}{2}\right)^{\overline{j}}(3/2)^{\overline{j}}}{\left(2-d+\frac{n}{2}\right)^{\overline{j}}(3/2-2d)^{\overline{j}}} \cdot \frac{1}{j!} \\ &= (2d-3/2)^{\underline{2d}} \sum_{j=0}^{\infty} \frac{(2-2d)^{\overline{j}}\left(1-d-\frac{n}{2}\right)^{\overline{j}}(3/2)^{\overline{j}}}{\left(2-d+\frac{n}{2}\right)^{\overline{j}}(3/2-2d)^{\overline{j}}} \cdot \frac{1}{j!} \\ &= (2d-3/2)^{\underline{2d}} \cdot {}_3F_2\left[\begin{matrix} a & b & c \\ 1+a-b & 1+a-c \end{matrix}; 1\right] \end{aligned} \tag{19}$$

where ${}_3F_2\left[\begin{smallmatrix} a & b & c \\ 1+a-b & 1+a-c \end{smallmatrix}; 1\right] = \sum_{j=0}^{\infty} \frac{(a)^{\overline{j}}(b)^{\overline{j}}(c)^{\overline{j}}}{(1+a-b)^{\overline{j}}(1+a-c)^{\overline{j}}} \cdot \frac{1}{j!}$ is the generalized hypergeometric series with $a = 2-2d$, $b = 1-d-n/2$ and $c = 3/2$.

Note that $1+a/2-b-c = \frac{n-1}{2} > 0$ and by using Dixon's identity [6] for the generalized hypergeometric series ${}_3F_2\left[\begin{smallmatrix} a & b & c \\ 1+a-b & 1+a-c \end{smallmatrix}; 1\right]$ (when $\Re(1+a/2-b-c) > 0$) we have

$$\begin{aligned} {}_3F_2\left[\begin{matrix} a & b & c \\ 1+a-b & 1+a-c \end{matrix}; 1\right] &= \frac{\Gamma(1+\frac{1}{2}a)\Gamma(1+a-b)\Gamma(1+a-c)\Gamma(1+\frac{1}{2}a-b-c)}{\Gamma(1+a)\Gamma(1+\frac{1}{2}a-b)\Gamma(1+\frac{1}{2}a-c)\Gamma(1+a-b-c)} \\ &= \frac{\Gamma(2-d)\Gamma(5/2-d+m)\Gamma(3/2-2d)\Gamma(m)}{\Gamma(3-2d)\Gamma(3/2+m)\Gamma(1/2-d)\Gamma(1-d+m)} \end{aligned}$$

Note that $\Gamma(2-d) = (1-d)\Gamma(1-d)$ and $\Gamma(3-2d) = 2(1-d)(1-2d)\Gamma(1-2d)$. By using the Euler's reflection formula we have that $\frac{\sin(\pi d)}{\pi} = \frac{1}{\Gamma(1-d)\Gamma(d)}$ and $\frac{\sin(\pi 2d)}{\pi} = \frac{1}{\Gamma(1-2d)\Gamma(2d)}$, and by the integrality of d we have that

$$\frac{\Gamma(1-d)}{\Gamma(1-2d)} = \frac{\sin(\pi 2d)(2d-1)!}{\sin(\pi d)(d-1)!} = \frac{2\cos(\pi d)(2d-1)!}{(d-1)!} = \frac{2(-1)^d(2d-1)!}{(d-1)!}.$$

Therefore

$$\frac{\Gamma(2-d)}{\Gamma(3-2d)} = (-1)^{d+1}\frac{(2d-2)!}{(d-1)!}.$$

Recall that for nonnegative integer values of x we have $\Gamma(\frac{1}{2}-x) = \frac{(-2)^x}{(2n-1)!!}\sqrt{\pi}$ and $\Gamma(\frac{1}{2}+x) = \frac{(2x-1)!!}{2^x}\sqrt{\pi}$, and the following holds.

$$\begin{aligned} {}_3F_2\left[\begin{matrix} a & b & c \\ 1+a-b & 1+a-c \end{matrix}; 1\right] &= (-1)^{d+1}\frac{(2d-2)!}{(d-1)!}\frac{(m-1)!}{(m-d)!} \cdot \frac{\Gamma(\frac{5}{2}-d+m)\Gamma(\frac{3}{2}-2d)}{\Gamma(\frac{3}{2}+m)\Gamma(\frac{1}{2}-d)} \\ &= \frac{4^{d-1}(2d-2)!(2d-1)!!}{(d-1)!(4d-3)!!}\frac{(2m-2d+3)!!(m-1)!}{(m-d)!(2m+1)!!} \end{aligned}$$

By simple inspection we see that ${}_3F_2\left[\begin{smallmatrix} a & b & c \\ 1+a-b & 1+a-c \end{smallmatrix}; 1\right]$ is always positive and $g(d,n)$, see (19), is negative as claimed. ◀

4 Rank bounds for detecting a particular empty integral hull

In [15] Laurent considers the representation of the empty set as (3) and shows that the Sherali-Adams procedure requires n levels to detect that $K = \emptyset$. She conjectures that the SoS rank of K is $n-1$. In this section we disprove this conjecture and derive a lower and upper bound for the SoS rank of K.

▶ **Theorem 11.** *The SoS rank of K in (3) can be bounded by $\Omega(\sqrt{n}) \leq \rho(K) \leq n - \Omega(n^{\frac{1}{3}})$.*

Proof.

The upper bound

By symmetry, the solution $y_I^N = \frac{1}{2^n}$ for each $I \subseteq N$ is feasible to $\text{SoS}_t(K)$ unless $\text{SoS}_t(K) = \emptyset$. Let us assume that such a solution is feasible and consider the constraint of K corresponding to $R = N$. Then, $g_R(x_I)$ is negative only when $I = \emptyset$.

To analyse the PSDness, we apply Theorem 3. Notice that in this case we can assume that $P(k)$ is of the form $G_0(k)$, since if $h > 0$, the only negative term in the sum (14) corresponding to $k = 0$ is canceled due to $G_h(0) = 0$, and the inequality holds trivially. Therefore, the PSDness condition (10) reduces to

$$\sum_{k=0}^{n} \binom{n}{k} \frac{1}{2^n} \left(k - \frac{1}{2}\right) P^2(k) \geq 0 \tag{20}$$

for every polynomial P of degree t. Importantly, what is not mentioned in the statement of Theorem 3, in this case the PSDness condition actually becomes an if and only if condition (see Theorem 7 in [13]). Therefore, showing that (20) is not satisfied implies that the PSDness condition (10) is not satisfied.

We now fix the polynomial as $P(k) = \prod_{i=1}^{t}(n - k - i + 1)$, i.e., such that P has the roots at $n, n-1, ..., n-t+1$, and argue that such a polynomial can never satisfy (20) when t is large. Indeed, rewriting the condition using this polynomial, removing the redundant factor $\frac{1}{2^n}$ and moving the negative term to the right hand side, we have the necessary requirement for the positive semidefiniteness that

$$\sum_{k=1}^{n-t} \binom{n}{k} \left(k - \frac{1}{2}\right) \prod_{i=1}^{t}(n - k - i + 1)^2 \geq \frac{1}{2} \prod_{i=1}^{t}(n - i + 1)^2 .$$

Notice that now the sum goes up to $n - t$ only, since all the terms $k > n - t$ are 0 by our choice of the polynomial. By dividing both sides by the positive term $\prod_{i=1}^{t}(n - i + 1)^2$ and observing that $\frac{\prod_{i=1}^{t}(n-k-i+1)}{\prod_{i=1}^{t}(n-i+1)} = \frac{(n-t)^{\underline{k}}}{n^{\underline{k}}}$, the condition further simplifies to

$$\sum_{k=1}^{n-t} \binom{n}{k} \left(k - \frac{1}{2}\right) \left(\frac{(n-t)^{\underline{k}}}{n^{\underline{k}}}\right)^2 \geq \frac{1}{2} \tag{21}$$

Next we upper bound the sum on the left hand side of (21) by considering a generic element for any $1 \leq k \leq n - t$. Any element can be bounded by

$$\binom{n}{k} \left(k - \frac{1}{2}\right) \left(\frac{(n-t)^{\underline{k}}}{n^{\underline{k}}}\right)^2 \leq \frac{n^k e^k}{k^k} k \frac{(n-t)^{2k}}{(n-k)^{2k}} \leq \frac{e^k}{k^{k-1}} \left(\frac{n(n-t)^2}{(n-k)^2}\right)^k .$$

Here we use $\frac{e^k}{k^{k-1}} < 3$ for any k and $\frac{1}{n-k} \leq \frac{1}{t}$ for $k \leq n - t$. Then, for $t \geq n - o(\sqrt{n})$, it holds $\frac{n(n-t)^2}{t^2} < 1$ so we can approximate

$$\frac{e^k}{k^{k-1}} \left(\frac{n(n-t)^2}{(n-k)^2}\right)^k \leq 3 \frac{n(n-t)^2}{t^2} .$$

Now, the sum on the left hand side of (21) is upper bounded by $3(n-t)\frac{n(n-t)^2}{t^2}$ and thus the solution is never feasible to $\text{SoS}_t(K)$ if

$$3 \frac{n(n-t)^3}{t^2} < \frac{1}{2} .$$

Setting $t = n - Cn^{\frac{1}{3}}$ satisfies the inequality asymptotically for an appropriate constant C.

The lower bound

We show that the symmetric solution $y_I^N = \frac{1}{2^n}$ is feasible for $\text{SoS}_t(K)$ when t is $\Omega(\sqrt{n})$. Again, by symmetry it is enough to show that the moment matrix of one constraint is PSD, and again we consider the constraint corresponding to $R = N$. Therefore, we need to show that (20) is satisfied for any choice of the polynomial P with degree less or equal to t. Writing the polynomial P in root form with roots $r_i,\ i = 1, ..., t$, we get similarly as in (21) the condition

$$\sum_{k=1}^{n} \binom{n}{k} \left(k - \frac{1}{2}\right) \prod_{i=1}^{t} \left(\frac{k - r_i}{r_i}\right)^2 \geq \frac{1}{2} \tag{22}$$

Now, we seek for a lower bound for the sum on the left hand side and find the condition on t such that the lower bound still exceeds $\frac{1}{2}$.

One can show (see [13]) that the roots r_i can be assumed to be real and to be located in the interval $[0, n]$. Furthermore, we can assume that the polynomial has degree of exactly t. Then, we look for the worst-case assignment for the roots.

No matter how the roots are located, there exist at least one non-zero point $k \in N$ such that $|k - r_i| \geq \frac{n}{2(t+1)}$ for every root r_i and $k \geq \lfloor \frac{n}{2(t+1)} \rfloor$. In the worst case the smallest of such points is $\lfloor \frac{n}{2(t+1)} \rfloor$. Let $u = \lfloor \frac{n}{2(t+1)} \rfloor$ be this point. Then, (22) is satisfied if we can show that

$$\binom{n}{u} \left(u - \frac{1}{2}\right) \frac{\left(\frac{n}{2(t+1)}\right)^{2t}}{\prod_{i=1}^{t} r_i^2} \geq \frac{1}{2}\,.$$

Next, the worst case of the location for the roots in this formulation is $r_i = n$ for every $i = 1, ..., t$, since all the roots can be assumed to be less or equal to n. We can also get rid of the term $u - \frac{1}{2}$, since it is always greater than 1. We then obtain that (22) holds if

$$\binom{n}{u} \frac{\left(\frac{n}{2(t+1)}\right)^{2t}}{n^{2t}} \geq \frac{1}{2}\,.$$

Here we use the inequality $\binom{n}{u} > \frac{n^u}{u^u}$ and the fact that $\frac{1}{2(t+1)} \geq \frac{1}{4t}$ to get that if $\frac{n^u}{u^u}(4t)^{-2t} \geq \frac{1}{2}$ holds, then the solution is feasible for $\text{SoS}_t(K)$. We have that $n \geq tu$, so the above is satisfied if

$$\frac{(tu)^u}{u^u}(4t)^{-2t} \geq \frac{1}{2} \Leftrightarrow t^u(4t)^{-2t} \geq \frac{1}{2}\,.$$

We have that $u \geq \frac{n}{4t}$, so it is enough to satisfy $t^{\frac{n}{4t}}(4t)^{-2t} \geq \frac{1}{2}$, which is equivalent to $4^{-2t}t^{\frac{n}{4t}-2t} \geq \frac{1}{2}$. If here $t = \frac{\sqrt{n}}{4}$ we need to then satisfy $4^{-\sqrt{n}}\sqrt{n}^{\sqrt{n}/2} \geq \frac{1}{2}$, which holds asymptotically in n. ◀

Open question. We note that applying the theorem of [13], it is possible to perform numerical experiments to test the SoS rank of the polytope K with large number of variables. For a fixed number of variables, the upper bound can be experimented by fixing the polynomial P in (20) in some systematic way and by finding the level t (i.e., the number of roots) for which the expression is positive/negative for that polynomial. For the lower bound, the polynomial in (20) can be expressed in the root form and the PSDness can be tested using a numerical solver to minimize the resulting expression, where the roots are the variables to be minimized.

Based on such experiments, we conjecture that the SoS rank of K is "close" to $\frac{n}{2}$ and suggest that our bounds in Theorem 11 are far from being tight. Therefore we leave it as an open question to improve our bounds for the rank.

Acknowledgements. The authors would like to express their gratitude to Alessio Benavoli for helpful discussions.

References

1 Yu Hin Au. *A Comprehensive Analysis of Lift-and-Project Methods for Combinatorial Optimization*. PhD thesis, University of Waterloo, 2014.

2 Boaz Barak, Fernando G. S. L. Brandão, Aram Wettroth Harrow, Jonathan A. Kelner, David Steurer, and Yuan Zhou. Hypercontractivity, sum-of-squares proofs, and their applications. In *STOC*, pages 307–326, 2012. doi:10.1145/2213977.2214006.

3 Grigoriy Blekherman. Symmetric sums of squares on the hypercube. Manuscript in preparation, 2015.

4 William Cook and Sanjeeb Dash. On the matrix-cut rank of polyhedra. *Mathematics of Operations Research*, 26(1):19–30, 2001.

5 Gérard Cornuéjols and Yanjun Li. On the rank of mixed 0, 1 polyhedra. In *IPCO*, pages 71–77. Springer, 2001.

6 AC Dixon. Summation of a certain series. *Proceedings of the London Mathematical Society*, 1(1):284–291, 1902.

7 Hamza Fawzi, James Saunderson, and Pablo A. Parrilo. Sparse sums of squares on finite abelian groups and improved semidefinite lifts. *Mathematical Programming*, pages 1–43, 2016. doi:10.1007/s10107-015-0977-z.

8 Michel X. Goemans and Levent Tunçel. When does the positive semidefiniteness constraint help in lifting procedures? *Mathematics of Operations Research*, 26(4):796–815, 2001. doi:10.1287/moor.26.4.796.10012.

9 Dima Grigoriev. Complexity of positivstellensatz proofs for the knapsack. *Computational Complexity*, 10(2):139–154, 2001. doi:10.1007/s00037-001-8192-0.

10 Dima Grigoriev. Linear lower bound on degrees of positivstellensatz calculus proofs for the parity. *Theoretical Computer Science*, 259(1-2):613–622, 2001. doi:10.1016/S0304-3975(00)00157-2.

11 Dima Grigoriev and Nicolai Vorobjov. Complexity of null-and positivstellensatz proofs. *Annals of Pure and Applied Logic*, 113(1-3):153–160, 2001. doi:10.1016/S0168-0072(01)00055-0.

12 Adam Kurpisz, Samuli Leppänen, and Monaldo Mastrolilli. On the hardest problem formulations for the 0/1 lasserre hierarchy. In *ICALP*, pages 872–885, 2015. doi:10.1007/978-3-662-47672-7_71.

13 Adam Kurpisz, Samuli Leppänen, and Monaldo Mastrolilli. Sum-of-squares lower bounds for maximally symmetric formulations. In *To appear in IPCO*, 2016. URL: http://arxiv.org/abs/1407.1746.

14 Jean B. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001. doi:10.1137/S1052623400366802.

15 Monique Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28(3):470–496, 2003. doi:10.1287/moor.28.3.470.16391.

16 Monique Laurent. Lower bound for the number of iterations in semidefinite hierarchies for the cut polytope. *Mathematics of Operations Research*, 28(4):871–883, 2003. doi:10.1287/moor.28.4.871.20508.

17 James R. Lee, Prasad Raghavendra, and David Steurer. Lower bounds on the size of semidefinite programming relaxations. In *STOC*, pages 567–576, 2015. doi:10.1145/2746539.2746599.

18 Troy Lee, Anupam Prakash, Ronald de Wolf, and Henry Yuen. On the sum-of-squares degree of symmetric quadratic functions. In *To appear in CCC*, 2016. URL: http://arxiv.org/abs/1601.02311.

19 Claire Mathieu and Alistair Sinclair. Sherali-adams relaxations of the matching polytope. In *STOC*, pages 293–302, 2009. doi:10.1145/1536414.1536456.

20 Raghu Meka, Aaron Potechin, and Avi Wigderson. Sum-of-squares lower bounds for planted clique. In *STOC*, pages 87–96, 2015. doi:10.1145/2746539.2746600.

21 Pablo Parrilo. *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*. PhD thesis, California Institute of Technology, 2000.

22 Shinsaku Sakaue, Akiko Takeda, Sunyoung Kim, and Naoki Ito. Exact sdp relaxations with truncated moment matrix for binary polynomial optimization problems. Technical report, University of Tokyo, 2016. URL: http://www.keisu.t.u-tokyo.ac.jp/research/techrep/data/2016/METR16-01.pdf.

23 Tamon Stephen and Levent Tunçel. On a representation of the matching polytope via semidefinite liftings. *Mathematics of Operations Research*, 24(1):1–7, 1999.

Correlation Decay and Tractability of CSPs*

Jonah Brown-Cohen[1] and Prasad Raghavendra[2]

1 **University of California, Berkeley, CA, USA**
`jonahbc@eecs.berkeley.edu`
2 **University of California, Berkeley, CA, USA**
`prasad@eecs.berkeley.edu`

Abstract

The algebraic dichotomy conjecture of Bulatov, Krokhin and Jeavons yields an elegant characterization of the complexity of constraint satisfaction problems. Roughly speaking, the characterization asserts that a CSP L is tractable if and only if there exist certain non-trivial operations known as polymorphisms to combine solutions to L to create new ones.

In this work, we study the dynamical system associated with repeated applications of a polymorphism to a distribution over assignments. Specifically, we exhibit a correlation decay phenomenon that makes two variables or groups of variables that are not perfectly correlated become independent after repeated applications of a polymorphism.

We show that this correlation decay phenomenon can be utilized in designing algorithms for CSPs by exhibiting two applications:

1. A simple randomized algorithm to solve linear equations over a prime field, whose analysis crucially relies on correlation decay.
2. A sufficient condition for the simple linear programming relaxation for a 2-CSP to be sound (have no integrality gap) on a given instance.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Constraint Satisfaction, Polymorphisms, Linear Equations, Correlation Decay

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.79

1 Introduction

A vast majority of natural computational problems have been classified to be either polynomial-time solvable or NP-complete. While there is little progress in determining the exact time complexity for fundamental problems like matrix multiplication, it can be argued that a much coarser classification of P vs.~NP-complete has been achieved for a large variety of problems. Notable problems that elude such a classification include factorization or graph isomorphism.

A compelling research direction at this juncture is to understand what causes problems to be easy (in P) or hard (NP-complete). More precisely, for specific classes of problems, does there exist a unifying theory that explains and characterizes why some problems in the class are in P while others are NP-complete? For the sake of concreteness, we will present a few examples.

It is well-known that 2-Sat is polynomial-time solvable, while 3-Sat is NP-complete. However, the traditional proofs of these statements are unrelated to each other and therefore shed little light on what makes 2-Sat easy while 3-Sat NP-complete.

* This work was supported by NSF Career Award & Alfred.P. Sloan Fellowship.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 79; pp. 79:1–79:13

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

Over the last decade, a theory of tractability has emerged for the class of constraint satisfaction problems (CSP). While this candidate theory remains conjectural for now, it successfully explains all the existing algorithms and hardness results for CSPs. To set the stage for the results of this paper, we begin with a brief survey of the theory for CSPs.

A constraint satisfaction problem (CSP) Λ is specified by a family of predicates over a finite domain $[q] = \{1, 2, \ldots, q\}$. Every instance of the CSP Λ consists of a set of variables $\mathcal{V}$, along with a set of constraints $\mathcal{C}$ on them. Each constraint in $\mathcal{C}$ consists of a predicate from the family Λ applied to a subset of variables. For a CSP Λ, the associated satisfiability problem Λ-Sat is defined as follows.

▶ **Problem 1** (Λ-Sat)**.** *Given an instance $\Im$ of the CSP Λ, determine whether there is an assignment satisfying all the constraints in $\Im$.*

A classic theorem of Schaefer [11] asserts that among all satisfiability problems over the boolean domain ($\{0, 1\}$), only Linear-Equations-Mod-2, 2-Sat, Horn-Sat, Dual-Horn Sat and certain trivial CSPs are solvable in polynomial time. The rest of the boolean CSPs are NP-complete. The dichotomy conjecture of Feder and Vardi [7] asserts that every Λ-Sat is in P or NP-complete. The conjecture has been shown to hold for CSPs over domains of size up to 3 [4].

In this context, it is natural to question as to what makes certain Λ-Sat problems tractable while the others are NP-complete. Bulatov, Jeavons and Krokhin [6] conjectured a beautiful characterization for tractable satisfiability problems. We will present an informal description of this characterization known as the *algebraic dichotomy conjecture*. We refer the reader to the work of Kun & Szegedy [8] for a more formal description.

To motivate this characterization, let us consider a CSP known as the XOR problem. An instance of the XOR problem consists of a system of linear equations over $\mathbb{Z}_2 = \{0, 1\}$. Fix an instance $\Im$ of XOR over n variables. Given three solutions $X^{(1)}, X^{(2)}, X^{(3)} \in \{0, 1\}^n$ to $\Im$, one can create a new solution $Y \in \{0, 1\}^n$:

$$Y_i = X_i^{(1)} \oplus X_i^{(2)} \oplus X_i^{(3)} \qquad \forall i \in [n]\,.$$

It is easy to check that Y is also a feasible solution to the instance $\Im$. Thus the $XOR : \{0, 1\}^3 \to \{0, 1\}$ yields a way to combine three solutions in to a new solution to the same instance. Note that the function XOR was applied to each bit of the solution individually. An operation of this form that preserves the satisfiability of the CSP is known as a *polymorphism*. Formally, a polymorphism of a CSP Λ-Sat is defined as follows:

▶ **Definition 2** (Polymorphisms)**.** A function $p : [q]^R \to [q]$ is said to be *a polymorphism* for the CSP Λ-Sat, if for every instance $\Im$ of Λ, and R assignments $X^{(1)}, X^{(2)}, \ldots, X^{(R)} \in [q]^n$ that satisfy all constraints in $\Im$, the vector $Y \in [q]^n$ defined below is also a feasible solution.

$$Y_i = p(X_i^{(1)}, X_i^{(2)}, X_i^{(3)}, \ldots, X_i^{(R)}) \qquad \forall i \in [n]\,.$$

Note that the dictator functions $p(x^{(1)}, \ldots, x^{(R)}) = x^{(i)}$ are polymorphisms for every CSP Λ-Sat. These will be referred to as *projections* or trivial polymorphisms. All the tractable cases of boolean CSPs in Schaefer's theorem are characterized by existence of non-trivial polymorphisms. Specifically, 2-SAT has the Majority function, Horn-SAT has the OR function, and Dual Horn-SAT has the AND function as a polymorphism. Roughly speaking, Bulatov et al. [6] conjectured that the existence of non-dictator polymorphisms characterizes CSPs that are tractable. Their work showed that the set of polymorphisms Poly(Λ) of a CSP Λ characterizes the complexity of Λ-Sat. There are many equivalent

ways of formalizing what it means for an operation to be *non-trivial* or *non-dictator*. A particularly simple way to formulate the algebraic dichotomy conjecture arises out of the recent work of Barto and Kozik [2]. A polymorphism $p : [q]^k \to [q]$ is called a *cyclic term* if

$$p(x_1, \ldots, x_k) = p(x_2, \ldots, x_k, x_1) = \ldots = p(x_k, x_1, \ldots, x_{k-1}) \quad \forall x_1, \ldots, x_k \in [q] \,.$$

Note that the above condition strictly precludes the operation p from being a dictator.

▶ **Conjecture 3** ([6, 9, 2]). Λ-SAT *is in* P *if* Λ *admits a cyclic term, otherwise* Λ-SAT *is* NP*-complete.*

Surprisingly, one of the implications of the conjecture has already been confirmed.

▶ **Theorem 4** ([6, 9, 2]). Λ-SAT *is* NP*-complete if* Λ *does not admit a cyclic term.*

The algebraic approach to understanding the complexity of CSPs has received much attention, and the algebraic dichotomy conjecture has been verified for several subclasses of CSPs such as conservative CSPs [3], CSPs with no ability to count [1] and CSPs with Maltsev operations [5]. Recently, Kun and Szegedy reformulated the algebraic dichotomy conjecture using analytic notions similar to influences [8].

Despite considerable progress in recent years [2], the algebraic dichotomy conjecture still remains open. Kun & Szegedy suggested the use of analytic techniques towards resolving the conjecture [8], which forms the inspiration for this work. This work demonstrates a phenomenon of correlation decay associated with iterated applications of polymorphisms and then exploits this phenomenon towards designing algorithms for CSPs.

1.1 Correlation Decay

We associate a natural dynamical system with a polymorphism $p : [q]^k \to [q]$ that corresponds to iterated applications of the polymorphism. Towards a formal definition of the dynamical system, let us fix a probability distribution μ over $[q]^n$. It is useful to think of μ as a distribution over assignments to a CSP instance on n variables.

For an operation $p : [q]^k \to [q]$, the distribution $p(\mu)$ over $[q]^n$ is one that is sampled by taking k independent samples from μ and applying the operation p to them. Define the dynamical system $\{\mu_t\}_{t \in \mathbb{N}}$ with $\mu_0 = \mu$,

$$\mu_t \stackrel{\text{def}}{=} p(\mu_{t-1}) \,, \forall t \in \mathbb{N}.$$

Roughly speaking, the key technical insight of this work is that the correlations among the coordinates decay as $t \to \infty$ for a *non-dictator* operation p. For the sake of simplicity, let us restrict our attention to the case of a distribution μ_{XY} on $[q] \times [q]$ (see Section 4 for the general theorem on distributions over $[q]^n$). Let $\mu_{|X}$ and $\mu_{|Y}$ denote the marginals of μ_{XY}. For any distribution Θ, let $\mathrm{supp}(\Theta)$ denote its support. We are ready to state a version of our correlation decay theorem.

▶ **Theorem 5.** *Let* μ_{XY} *be a distribution over* $[q] \times [q]$*. Let* $G_{\mu_{XY}}$ *denote the bipartite graph on vertices* $\mathrm{supp}(\mu_{|X}) \cup \mathrm{supp}(\mu_{|Y})$ *whose edges are given by the support of* μ_{XY}*. For a cyclic term* $p : [q]^k \to [q]$*, consider the dynamical system* $\{\mu_t\}_{t \in \mathbb{N}}$ *defined as* $\mu_0 := \mu_{XY}$*,* $\mu_t := p(\mu_{t-1}) \forall t \in \mathbb{N}$*. If* $G_{\mu_{XY}}$ *is a connected graph then*

$$\lim_{t \to \infty} \|\mu_t - \mu_{t|X} \times \mu_{t|Y}\|_1 = 0 \,.$$

i.e., μ_t *gets closer and closer to a product distribution as* $t \to \infty$*.*

Suppose p is a polymorphism of a CSP Λ and μ_{XY} is a distribution supported over satisfying assignments to an instance of Λ, then for every t, μ_t is also supported over satisfying assignments to Λ. Intuitively, this seems to be at odds with the correlation decay phenomenon: cyclic polymorphisms make variables uncorrelated yet still preserve satisfying assignments.

To resolve this paradox, notice that Theorem 5 requires that the graph $G_{\mu_{XY}}$ be *connected*. Connectivity of $G_{\mu_{XY}}$ corresponds to asserting that there are *no perfect correlations* between X and Y. This lack of perfect correlation can be quantified using the spectrum of the bipartite graph $G_{\mu_{XY}}$. Specifically, one can associate a correlation parameter $\rho(X,Y)$ (see Definition 18) such that $\rho(X,Y) < 1$ if and only if $G_{\mu_{XY}}$ is connected. $\rho(X,Y)$ is closely related to the second-eigenvalue of the adjacency matrix of the bipartite graph $G_{\mu_{XY}}$.

If $G_{\mu_{XY}}$ is disconnected, every connected component of $G_{\mu_{XY}}$ corresponds to a *perfect correlation* between X and Y. If the support of μ consists of all satisfying assignments to a constraint, then the polymorphism p necessarily preserves these *perfect correlations*, i.e., for all $t \in \mathbb{N}$, (X,Y) sampled from μ_t will be such that X and Y belong to the same connected component of $G_{\mu_{XY}}$. Summarizing, our result suggests that a cyclic polymorphism preserves *perfect correlations*, while *imperfect correlations* decay.

Discussion. A brief overview of the correlation decay argument is presented in Section 4. The details of the argument are fairly technical and draw upon various analytic tools such as hypercontractivity, the Berry-Esseen theorem and Fourier analysis (see Section 4). A key bottleneck in the analysis is that the individual marginals change with each iteration thereby changing the fourier spectrum of the operations involved.

Theorem 5 can be thought of as an analytic analogue of a theorem on *absorbing subalgebras* (Theorem 4.11 in [1]), which formed a key ingredient in the breakthrough work of Barto and Kozik [1]. This work of Barto and Kozik showed that a major subclass of CSPs namely *CSPs with no ability to count* can be solved using local consistency. Roughly speaking, *CSPs with no ability to count* are precisely those that don't contain linear equations within them, i.e., these CSPs don't admit gadget reductions from linear equations over a finite field.

Interestingly, we will show that the same correlation decay phenomenon is useful in solving linear equations over prime fields! We will also present an application of correlation decay towards rounding linear programming relaxations for CSPs. We will outline these two applications in the upcoming subsections.

1.2 Solving Linear Systems

The input to the algorithm consists of a linear system $Ax = b$ where $A \in \mathbb{F}_q^{m \times n}$ and $b \in \mathbb{F}_q^n$. Consider the following naive algorithm for solving the linear systems for some $N \in \mathbb{N}$.

1. $\mathcal{S}_0 \leftarrow N$ uniformly random assignments from $\mathbb{F}_q^n$.
2. for $i = 1$ to m do
   - **Selection:** $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cap \{x | \langle A_i, x\rangle = b_i\}$
3. Output an assignment $z \in \mathcal{S}_m$ if $\mathcal{S}_m \neq \emptyset$, else output INFEASIBLE

Clearly, if the algorithm outputs an assignment z then $Az = b$. By definition, the i^{th} *generation* $\mathcal{S}_i$ consists of uniformly random assignments that satisfy the first i equations $\{\langle A_j, x\rangle = b_j | j \leqslant i\}$. Therefore, the $i+1^{st}$ linear function $\langle A_{i+1}, x\rangle$ is either constant over $\mathcal{S}_i$ or takes every value in $\mathbb{F}_q$ with probability roughly $1/q$.

If the linear system $Ax = b$ is satisfiable and is linearly independent, then the expected size of i^{th} generation $\mathcal{S}_i$ is given by $\mathbb{E}\,|\mathcal{S}_i| = \frac{1}{q}\,\mathbb{E}\,|\mathcal{S}_{i-1}|$. Therefore the initial sample size

Input: $A \in \mathbb{F}_q^{m \times n}$, $b \in \mathbb{F}_q^m$

1. $\mathcal{S}_0 \leftarrow N$ uniformly random assignments from $\mathbb{F}_q^n$.
2. for $i = 1$ to m do
   - **Selection:** $\mathcal{T}_i \leftarrow \mathcal{S}_{i-1} \cap \{x | \langle A_i, x \rangle = b_i\}$
   - **Recombination:** Generate $\mathcal{S}_i$ by using the following sampling procedure N times independently.
     - Sample $y^{(1)}, y^{(2)}, y^{(3)}$ i.i.d from $\mathcal{T}_i$ and compute $p(y^{(1)}, y^{(2)}, y^{(3)}) = y^{(1)} - y^{(2)} + y^{(3)}$.
3. Output an assignment $z \in \mathcal{S}_m$ if $\mathcal{S}_m \neq \emptyset$, else output INFEASIBLE

Figure 1 Randomized algorithm for linear equations over $\mathbb{F}_q$.

$|\mathcal{S}_0|$ has to be at least q^m to ensure the correctness of the algorithm, making the runtime exponential.

A natural approach to fix the algorithm is as follows. After each selection step, use the polymorphism associated with linear systems in order to *create* new assignments from the existing sample. For the sake of concreteness, we fix the following polymorphism $p : [q]^3 \to [q]$ for linear systems.

$$p(y^{(1)}, y^{(2)}, y^{(3)}) = y^{(1)} - y^{(2)} + y^{(3)} \, .$$

The details of the algorithm are as shown in Figure 1.

Since p is a polymorphism, the recombination steps don't affect the progress made in the selection steps, i.e., $\mathcal{S}_i$ satisfy the first i equations for each i. While the polymorphism p is useful to maintain the sample size after each selection, the sample size alone is insufficient to ensure the success of the algorithm. We require the sample $\mathcal{S}_i$ to be somewhat similar a uniformly random sample from the set of all solutions to the first i equations.

Here is an alternate take on the issue. A finite sized sample $\mathcal{S}$ of a distribution μ has *spurious correlations* that are absent in μ. For example, in the initial sample $\mathcal{S}_0$, the first two variables $x_1, x_2 \in [q]$ will be close to independent, but there is bound to be some assignment $\alpha, \beta \in [q]^2$ such that $\mathbb{P}_{x \in \mathcal{S}_0}[x_1 = \alpha \wedge x_2 = \beta] \geqslant 1/q^2 + \Omega(1/\sqrt{N})$ due to random deviation. At the i^{th} stage, the sample $\mathcal{S}_i$ has a set of *perfect correlations* induced by the first i equations, but there are additional *spurious correlations* between the variables owing to sample size being bounded.

The magnitudes of spurious correlations in the sample need to be controlled. Otherwise, a spurious correlation could result in $\langle A_i, x \rangle \neq b_i$ for all $x \in \mathcal{S}_{i-1}$ for some i, making $\mathcal{T}_i = \emptyset$ even though the linear system is satisfiable. Each selection step reduces the sample size thereby potentially amplifying the spurious correlations. However, the recombination step exploits the correlation decay phenomena to decrease the spurious correlations. In Section 3, we will show the following.

▶ **Theorem 6.** *For all primes q, the randomized algorithm in Figure 1 with the choice $N = \lceil (150q^4 \ln q) \cdot n \rceil$ satisfies these properties.*

Completeness: *If algorithm returns $z \in \mathbb{F}_q^n$, then z satisfies the linear system $Az = b$.*

Soundness: *If the system $Ax = b$ is feasible, then with probability at least $1 - e^{-n}$ the algorithm will return a solution to the system.*

The algorithm described above is somewhat similar to a deterministic algorithm of Bulatov and Dalmau [5] for CSPs admitting Maltsev polymorphisms in that it maintains a basis for the solution space and updates the basis by including one equation in to the system at each step. We find the randomized algorithm interesting in that it admits a very generic

description that uses little about the structure of the underlying CSP. Moreover, the analysis of the algorithm crucially relies on correlation decay – a phenomenon that seems inherent to all tractable CSPs.

1.3 Rounding Linear Programs via Correlation Decay

Correlation decay can also be used towards rounding linear programming relaxations. For the sake of clarity, let us restrict our attention to 2-CSPs where every constraint has arity two. These results can be generalized to k-CSPs. Further, every CSP can be reduced to a 2-CSP while preserving the existence of cyclic terms.

First, we introduce the BasicLP linear programming relaxation for CSPs of arity two. Let Λ be a CSP of arity two over the alphabet $[q]$, and $\mathcal{I}$ be an instance of Λ. For every variable X in $\mathcal{I}$, the LP associates a probability distribution μ_X over $[q]$. For every constraint $C_i(X,Y)$ in $\mathcal{I}$, the LP associates a probability distribution μ_{XY} over $[q] \times [q]$, supported on satisfying assignments to the constraint C_i. The pairwise distributions μ_{XY} are constrained to be consistent with the marginal distributions μ_X and μ_Y. The BasicLP program for 2-CSPs is described in detail in Section 5.

▶ **Definition 7.** An LP relaxation $\mathcal{L}$ for a CSP is *sound* on an instance $\mathcal{I}$ if the feasibility of the LP relaxation $\mathcal{L}$ on $\mathcal{I}$ implies satisfiability of the instance $\mathcal{I}$.

Typically, one shows the soundness of an LP relaxation by a rounding scheme that extracts an assignment to the CSP from the LP solution. We exhibit a sufficient condition for an LP relaxation to be *sound* on an instance $\mathcal{I}$. For a constraint $C_i(X,Y)$, let $G_{\mu_{XY}}$ denote the bipartite graph whose edges are given by the support of the distribution μ_{XY}.

▶ **Theorem 8.** *Let Λ be a 2-CSP that admits a cyclic polymorphism and let $\mathcal{I}$ be an instance Λ. Suppose there exists a solution to the* BasicLP *relaxation for $\mathcal{I}$ such that all the associated graphs $G_{\mu_{XY}}$ are connected, then the* BasicLP *relaxation is* sound *on the instance $\mathcal{I}$, i.e., $\mathcal{I}$ is satisfiable.*

2 Background

We first introduce some basic notation. Let $[q]$ denote the alphabet $[q] = \{1, \ldots, q\}$. For a probability distribution μ on the finite set $[q]$ we will write μ^k to denote the product distribution on $[q]^k$ given by drawing k independent samples from μ.

If μ is a joint probability distribution on $[q]^n$ we will write $\mu_1, \mu_2, \ldots \mu_n$ for the n marginal distributions of μ. Further we will use $\mu^\times$ to denote the product distribution with the same marginals as μ. That is we define $\mu^\times \stackrel{\text{def}}{=} \mu_1 \times \mu_2 \times \cdots \times \mu_n$.

An operation p of arity k is a map $p : [q]^k \to [q]$. For a set of k assignments $x^{(1)}, \ldots, x^{(k)} \in [q]^n$, we will use $p(x^{(1)}, \ldots, x^{(k)}) \in [q]^n$ to be the assignment obtained by applying the operation p on each coordinate of $x^{(1)}, \ldots, x^{(k)}$ separately. More formally, let $x_j^{(i)}$ be the jth coordinate of x_i. We define

$$p(x^{(1)} \ldots x^{(k)}) = \left(p(x_1^{(1)} \ldots x_1^{(k)}), p(x_2^{(1)} \ldots x_2^{(k)}), \ldots, p(x_n^{(1)} \ldots x_n^{(k)})\right).$$

▶ **Definition 9.** For two operations $p_1 : [q]^{k_1} \to [q]$ and $p_2 : [q]^{k_2} \to [q]$, define an operation $p_1 \otimes p_2 : [q]^{k_1 \times k_2} \to [q]$ as follows:

$$p_1 \otimes p_2(\{x_{ij}\}_{i \in [k_1], j \in [k_2]}) = p_1\left(p_2(x_{11}, x_{12}, \ldots, x_{1k_2}), \ldots, p_2(x_{k_1 1}, x_{k_1 2}, \ldots, x_{k_1 k_2})\right)$$

▶ **Lemma 10** (Hoeffding bound)**.** *Suppose $Z_1, \ldots, Z_N$ are complex-valued random variables such that $|Z_i|$ is always bounded by 1. If $Z = \frac{1}{N}\sum_i Z_i$ then, $\mathbb{P}[|Z - \mathbb{E}[Z]| \geqslant \delta] \leqslant 2e^{-\delta^2 N/4}$.*

3 Solving Linear Systems via Correlation Decay

In this section, we will analyze the randomized algorithm for linear equations in Figure 1 and present a proof of Theorem 6. We begin with setting up some notation dealing with affine subspaces of $\mathbb{F}_q^n$.

▶ **Definition 11.** For an affine subspace $V \subseteq \mathbb{F}_q^n$ and $b \in \mathbb{F}_q$, set $V_b^\perp \stackrel{\text{def}}{=} \{w | \forall x \in V, \langle x, w\rangle = b\}$, and let $V^\perp \stackrel{\text{def}}{=} \cup_{b\in\mathbb{F}_q} V_b$.

In the setting of linear equations, correlations can be measured using Fourier analysis. Therefore, we recall the definition of characters over $\mathbb{F}_q^n$.

▶ **Definition 12.** For every $w \in \mathbb{F}_q^n$, the corresponding character χ_w is a function $\chi_w : \mathbb{F}_q^n \to \mathbb{C}$ given by $\chi_w(x) = \omega^{\langle w,x\rangle}$, where ω is a primitive q^{th} root of unity and $\langle w, x\rangle$ denotes the inner product of w and x over $\mathbb{F}_q$.

We will quantify the *spurious correlations* in our sample using the notion of bias as defined below.

▶ **Definition 13** (Bias)**.** For a vector $w \in \mathbb{F}_q^n$ and a multiset $\mathcal{S} \subseteq \mathbb{F}_q^n$, define the bias of w over $\mathcal{S}$ as,

$$\text{bias}_w(\mathcal{S}) = |\mathop{\mathbb{E}}_{x\in\mathcal{S}}[\chi_w(x)]| = \frac{1}{|\mathcal{S}|}|\sum_{x\in\mathcal{S}} \chi_w(x)|\,.$$

An ε-biased sample from an affine subspace V is one in which all the *spurious correlations* are bounded by ε. Formally,

▶ **Definition 14** (ε-biased sample)**.** For $\varepsilon \in [0, 1]$, a multiset of vectors $\mathcal{S} \in \mathbb{F}_q^n$ is a ε-biased sample of an affine subspace $V \subseteq \mathbb{F}_q^n$ if $\mathcal{S} \subseteq V$ and for all $w \notin V^\perp$, $|\text{bias}_w(\mathcal{S})| \leqslant \varepsilon$.

Effect of Selection on Bias

▶ **Lemma 15.** *Let $\mathcal{S}$ be a ε-biased sample from an affine subspace V. For all $w \notin V^\perp$ and $b \in \mathbb{F}_q$, the following holds:*

1. $\mathbb{P}_{z\in\mathcal{S}}[\langle w, z\rangle = b] \in \left[\frac{1}{q} - \varepsilon, \frac{1}{q} + \varepsilon\right]$.
2. *If $\mathcal{T} = \mathcal{S} \cap \{z | \langle w, z\rangle = b\}$ then $\mathcal{T}$ is a $q\varepsilon/(1-q\varepsilon)$-biased sample from the affine subspace $V \cap \{z \in \mathbb{F}_q^n | \langle w, z\rangle = b\}$.*

Proof. Let $\mathbb{I}_{\langle w,z\rangle=b}$ be the indicator of the event that $\langle w, z\rangle = b$. Using the identity $\mathbb{I}_{\langle w,z\rangle=b} = \frac{1}{q}\sum_{\alpha\in\mathbb{F}_q}(\chi_w(z)\omega^{-b})^\alpha$, we can write

$$\mathop{\mathbb{P}}_{z\in\mathcal{S}}[\langle w, z\rangle = b] = \mathop{\mathbb{E}}_{z\in\mathcal{S}}\left[\frac{1}{q}\sum_{\alpha\in\mathbb{F}_q}(\chi_w(z)\omega^{-b})^\alpha\right]\,.$$

Simplifying the above expression using the identity $\chi_w(z)^\alpha = \chi_{\alpha w}(z)$ we get

$$\mathop{\mathbb{P}}_{z\in\mathcal{S}}[\langle w, z\rangle = b] = \frac{1}{q} + \frac{1}{q}\sum_{\alpha\in\mathbb{F}_q/\{0\}} \omega^{-\alpha b} \mathop{\mathbb{E}}_{z\in\mathcal{S}}[\chi_{\alpha w}(z)]\,.$$

Hence,

$$\left|\mathop{\mathbb{P}}_{z\in\mathcal{S}}[\langle w, z\rangle = b] - \frac{1}{q}\right| \leqslant \frac{1}{q}\sum_{\alpha\in\mathbb{F}_q/\{0\}}\left|\mathop{\mathbb{E}}_{z\in S}[\chi_{\alpha w}(z)]\right| \leqslant \frac{1}{q}\sum_{\alpha\in\mathbb{F}_q/\{0\}}|\text{bias}_{\alpha w}(\mathcal{S})| \leqslant \varepsilon$$

Let V' denote the affine subspace $V' = V \cap \{z \in \mathbb{F}_q^n | \langle w, z\rangle = b\}$. Clearly, $\mathcal{T}$ is a subset of V' if $\mathcal{S}$ is a subset of V. By definition, for every $v \in \mathbb{F}_q^n$ the bias over $\mathcal{T}$ is given by

$$\text{bias}_v(\mathcal{T}) = \mathop{\mathbb{E}}_{z\in\mathcal{T}}[\chi_v(z)] = \mathop{\mathbb{E}}_{z\in\mathcal{S}}[\chi_v(z)|\langle w, z\rangle = b] = \frac{1}{\mathbb{P}[\langle w, z\rangle = b]} \mathop{\mathbb{E}}_{z\in\mathcal{S}}\left[\chi_v(z)\mathbb{I}_{\langle w,z\rangle=b}\right] \quad (3.1)$$

Evaluating the expectation inside,

$$\mathop{\mathbb{E}}_{z\in\mathcal{S}}\left[\chi_v(z)\mathbb{I}_{\langle w,z\rangle=b}\right] = \frac{1}{q}\sum_{\alpha\in\mathbb{F}_q}\mathop{\mathbb{E}}_{z\in\mathcal{S}}\left[\chi_v(z)(\chi_w(z)\omega^{-b})^\alpha\right] = \frac{1}{q}\sum_{\alpha\in\mathbb{F}_q}\omega^{-b\alpha}\text{bias}_{v+\alpha w}(\mathcal{S}) \quad (3.2)$$

For a vector $v \notin (V')^\perp$, we claim that $v + \alpha w \notin V^\perp$ for any $\alpha \in \mathbb{F}_q$. Suppose not, then $v + \alpha w \in V^\perp$ which implies that for some $b' \in \mathbb{F}_q$, we have $\langle v + \alpha w, z\rangle = b'$ for all $z \in V$. This implies that for all $z \in V'$, $\langle v, z\rangle = \langle v + \alpha w, z\rangle - \alpha\langle w, z\rangle = b' - \alpha b$ – a constant, a contradiction to the fact that $v \notin (V')^\perp$.

Since $\mathcal{S}$ is an ε-biased sample from V and $v + \alpha w \notin V^\perp$, we have $\text{bias}_{v+\alpha w}(\mathcal{S}) \leqslant \varepsilon$. Using this bound in (3.2) we get $\mathbb{E}_{z\in S}\left[\chi_v(z)\mathbb{I}_{\langle w,z\rangle=b}\right] \leqslant \varepsilon$ .Substituting in (3.1) and using the fact that $\mathbb{P}[\langle w, z\rangle = b] \geqslant 1/q - \varepsilon$, we conclude that $\text{bias}_v(\mathcal{T}) \leqslant \frac{\varepsilon}{(1/q-\varepsilon)}$ for any $v \in V^\perp$. ◀

Recombination Reduces Bias

▶ **Lemma 16.** *For all $i \in [m]$, if the sample $\mathcal{T}_i \in \mathbb{F}_q^n$ is a ε-biased sample of an affine subspace $V \subseteq \mathbb{F}_q^n$, then for all $\delta > 0$, then the sample $\mathcal{S}_i$ generated by recombination is a $\varepsilon^3 + \delta$-biased sample from V with probability at least $1 - 2q^n e^{-\delta^2 N/4}$.*

Proof. The multiset $\mathcal{S}_i$ consists of N-i.i.d samples from a probability distribution. Fix any $w \in V^\perp$. The expected value of the bias w over U is given by,

$$\mathbb{E}\left[\text{bias}_w(U)\right] = \mathbb{E}\left[\frac{1}{N}\sum_{z\in U}\chi_w(z)\right] = \frac{1}{N}\sum_{z\in U}\mathbb{E}\left[\chi_w(z)\right]$$

For every sample $z = p(y^{(1)}, y^{(2)}, y^{(3)})$ in $\mathcal{S}_i$, the expectation of the bias is given by

$$|\mathop{\mathbb{E}}_{y^{(j)}\in\mathcal{T}_i}[\chi_w(y_1 - y_2 + y_3)]| = |\mathop{\mathbb{E}}_{y^{(j)}\in\mathcal{T}_i}[\chi_w(y^{(1)})\overline{\chi_w(y^{(2)})}\chi_w(y^{(3)})]| = |\prod_{j=1}^{3}\mathop{\mathbb{E}}_{y^{(j)}\in\mathcal{T}_i}[\chi_w(y^{(j)}]|$$

Therefore, the expected value of the bias of w over $\mathcal{S}_i$ is, $\mathbb{E}\left[\text{bias}_w(\mathcal{S}_i)\right] \leqslant \text{bias}_w^3(\mathcal{T}_i)$. Since $\mathcal{T}_i$ is a ε-biased sample from V, for each $w \in V^\perp$ $|\text{bias}_w(\mathcal{T}_i)| \leqslant \varepsilon$. Therefore, we get that

$$\begin{aligned}\mathbb{P}\left[|\text{bias}_w(\mathcal{S}_i)| \geqslant \varepsilon^3 + \delta\right] &\leqslant \mathbb{P}\left[|\text{bias}_w(\mathcal{S}_i) - \mathbb{E}[\text{bias}_w(\mathcal{S}_i)]| \geqslant \delta\right] \\ &\leqslant 2e^{-\delta^2 N/4} \qquad \text{(Lemma 10)}\end{aligned}$$

By a union bound over all q^n characters $w \in \mathbb{F}_q^n$, we get the desired result. ◀

Analysis of the Algorithm

Proof of Theorem 6. Fix $\varepsilon = \frac{1}{2q^2}$ and $\delta = \frac{1}{6q}$. Let V_k denote the affine subspace consisting of solutions to the first k equations $\{A_i x = b_i | 1 \leqslant i \leqslant k\}$. We will show the following claim from which Theorem 6 follows immediately.

▶ **Claim 17.** *If $N = \lceil(150q^4 \ln q) \cdot n\rceil$, then with probability at least $1 - e^{-n}$ the following holds: for all $0 \leqslant k \leqslant m$, $\mathcal{S}_k$ is an ε-biased sample from V_k for $\varepsilon = \frac{1}{4q}$.*

The argument is by induction on k. For $k = 0$, the set S_0 consists of N random vectors from $\mathbb{F}_q^n$ and $V_0 = \mathbb{F}_q^n$. By definition, $V_0^\perp = \mathbb{F}_q^n - \{0\}$. For every $w \in \mathbb{F}_q^n - \{0\}$, the bias of w is given by, $\text{bias}_w(\mathcal{S}_0) = \frac{1}{N}\sum_{i=1}^N \chi_w(z_i)$. In particular, it is easy to see that $\mathbb{E}[\text{bias}_w(\mathcal{S}_0)] = 0$. By applying Lemma 10, we get that

$$\mathbb{P}[|\text{bias}_w(\mathcal{S}_0)| \geqslant \varepsilon] \leqslant 2e^{-\varepsilon^2 N/4}\,.$$

By a simple union bound, $\mathcal{S}_0$ is ε-biased sample with probability at least $1 - q^n 2e^{-\varepsilon^2 N/4} = 1 - q^n 2e^{-N/16q^4}$.

Let us suppose $\mathcal{S}_\ell$ is a ε-biased sample from V_ℓ. By Lemma 15, we get that $\mathcal{T}_{\ell+1}$ is a $q\varepsilon/(1-q\varepsilon)$-biased sample from $V_{\ell+1}$. By Lemma 16, with probability at least $1 - 2q^n e^{-\delta^2 N/4}$, the bias of $\mathcal{S}_{\ell+1}$ obtained by recombination is at most,

$$\text{bias}(\mathcal{S}_{\ell+1}) \leqslant \text{bias}(\mathcal{T}_{\ell+1})^3 + \delta \leqslant (q\varepsilon/(1-q\varepsilon))^3 + \delta = \frac{1}{q^3} + \delta < \varepsilon\,.$$

Applying a union bound over all $\ell \in \{0,\ldots,m\}$, with probability at least $1 - 2mq^n e^{-N/144q^2}$, $\mathcal{S}_\ell$ is an ε-biased sample from V_ℓ for all $\ell \in \{1,\ldots,m\}$. Setting $N = \lceil (150q^4 \ln q)\cdot n\rceil$, the claim follows. ◀

4 Correlation Decay

In this section we state our main theorem regarding the decay of correlation between random variables under repeated applications of cyclic operations. Recall that Theorem 5 states the theorem for two variables. Throughout this section we will use this two variable case as a running example. We begin by defining a quantitative measure of correlation and using it to bound the statistical distance to a product distribution.

4.1 Correlation and Statistical Distance

To gain intuition for our measure of correlation consider the example of two boolean random variables X and Y with joint distribution μ. In this case we will measure correlation by taking the supremum over appropriately normalized test functions $f, g : \{0,1\} \to \mathbb{R}$ and computing $\mathbb{E}[f(X)g(Y)]$.

▶ **Definition 18.** Let X, Y be discrete-valued random variables with joint distribution μ. Let $\Omega_1 = ([q_1], \mu_1)$ and $\Omega_2 = ([q_2], \mu_2)$ denote the probability spaces corresponding to X, Y respectively. The correlation $\rho(X, Y)$ is given by

$$\rho(X,Y) \stackrel{\text{def}}{=} \sup_{f,g} \mathbb{E}[f(X)g(Y)]$$

where the supremum runs over all f, g where $\mathbb{E}[f] = \mathbb{E}[g] = 0$ and $\mathbf{Var}[f] = \mathbf{Var}[g] = 1$. We will interchangeably use the notation $\rho(\mu)$ or $\rho(\Omega_1, \Omega_2)$ to denote the correlation.

To see that this notion of correlation makes intuitive sense, suppose X and Y are independent. In this case correlation is zero because $\mathbb{E}[f(X)g(Y)] = \mathbb{E}[f(X)]\,\mathbb{E}[g(Y)] = 0$. Next suppose that $X = Y = 1$ with probability $\frac{1}{2}$ and $X = Y = 0$ with probability $\frac{1}{2}$. In this case we can set $f(1) = g(1) = 1$ and $f(0) = g(0) = -1$ to obtain $\mathbb{E}[f(X)g(Y)] = 1$. This matches up with the intuition that such an X and Y are perfectly correlated. We now give the general definition for our measure of correlation. Next we show that, as the correlation for a pair of random variables X and Y becomes small, the variables become nearly independent.

▶ **Lemma 19.** *Let X, Y be discrete-valued random variables with joint distribution μ_{XY} and respective marginal distributions μ_X and μ_Y. If X takes values in $[q_1]$ and Y takes values in $[q_2]$, then $\|\mu_{XY} - \mu_X \times \mu_Y\|_1 \leqslant \min(q_1, q_2)\rho(X, Y)$*

It turns out that there is also a simple combinatorial condition that is essentially equivalent to a bound on the correlation. First we define a natural bipartite graph associated to a joint distribution.

▶ **Definition 20.** Let X, Y be jointly distributed according to μ as in Definition 18. Define a bipartite graph G_μ on vertex set $([q_1], [q_2])$ by adding an edge (a, b) whenever $\mathbb{P}_\mu[X = a, Y = b] > 0$.

Now the following lemma from [10] states that $\rho(\mu) < 1$ whenever the graph G_μ is connected.

▶ **Lemma 21** (Lemma 2.9 in [10])**.** *Let μ be a joint distribution where the minimum non-zero probability that μ assigns to any element is α. If G_μ is connected then $\rho(\mu) < 1 - \frac{\alpha^2}{2}$.*

In addition, if G_μ is disconnected, then $\rho(\mu) = 1$. Therefore checking if $\rho(\mu) < 1$ amounts to checking connectivity of G_μ.

4.2 Proof Overview

To begin with, we explain why one should expect correlations to decay under repeated applications of cyclic operations. Consider the simple example of two boolean random variables X and Y with a joint distribution μ. Let the marginal distributions of X and Y be uniform and let us suppose $X = Y$ with probability $\frac{1}{2} + \gamma$ and $X \neq Y$ with the remaining probability. Let $p : \{0,1\}^k \to \{0,1\}$ be the majority operation on k bits.

Next suppose we draw k samples (X_i, Y_i) from μ and evaluate $p(X_1 \dots X_k)$ and $p(Y_1 \dots Y_k)$. Since the marginal distributions of both X and Y are uniform, the same is true for $p(X_1 \dots X_k)$ and $p(Y_1 \dots Y_k)$. However, the probability that $p(X_1 \dots X_k) = p(Y_1 \dots Y_k)$ is strictly less than $\frac{1}{2} + \gamma$. To see why first let $F : \{-1, 1\} \to \{-1, 1\}$ be the majority function where 1 encodes boolean 0 and -1 encodes boolean 1. Note that the probability that $F(X_1 \dots X_k) = F(Y_1 \dots Y_k)$ is given by $\frac{1}{2} + \frac{1}{2}\mathbb{E}[F(X_1 \dots X_k)F(Y_1 \dots Y_k)]$.

Now if we write the Fourier expansion of F the above expectation is

$$\sum_{S,T} \hat{F}_S \hat{F}_T \, \mathbb{E}\left[\prod_{i \in S} X_i \prod_{j \in T} Y_j\right] = \sum_S \hat{F}_S^2 \prod_{i \in S} \mathbb{E}[X_i Y_i] = \sum_S \hat{F}_S^2 (2\gamma)^{|S|}$$

Suppose first that all the non-zero Fourier coefficients $\hat{F}_S$ have $|S| = 1$. In this case the probability that $F(X_1 \dots X_k) = F(Y_1 \dots Y_k)$ stays the same since $\frac{1}{2} + \frac{1}{2}(2\gamma) = \frac{1}{2} + \gamma$. However, in the case of majority, it is well known that $\sum_{|S|=1} \hat{F}_S^2 < 1 - c$ for a constant $c > 0$. Thus, the expectation is in fact given by

$$\mathbb{E}[F(X_1 \dots X_k)F(Y_1 \dots Y_k)] \leqslant (1 - c)(2\gamma) + c(2\gamma)^2 < 2\gamma$$

Thus the probability that $F(X_1 \dots X_k) = F(Y_1 \dots Y_k)$ is strictly less than $\frac{1}{2} + \gamma$. Therefore, if we repeatedly apply the majority operation, we should eventually have that X and Y become very close to independent.

There are two major obstacles to generalizing the above observation to arbitrary cyclic operations. First, for a general operation p, we will not be able to explicitly compute the entire Fourier expansion. Instead, we will have to use the fact that p is cyclic to get a bound

on the total Fourier mass on degree-one terms. Second, unlike in our example, the marginal distributions of X and Y may change after every application of p. This means that the correct Fourier basis to use also changes.

The fact that the marginal distributions change under p causes difficulties even for the simple example of the boolean OR operation on two bits. Consider a highly biased distribution over $0, 1$ given by $X = 1$ with probability ε and $X = 0$ with probability $1 - \varepsilon$. Now consider the function $f(X) = \frac{1}{2}(X_1 + X_2)$. Note that this function agrees with OR except when $X_1 \neq X_2$. Thus, $f(X) = OR(X)$ with probability $1 - 2\varepsilon(1 - \varepsilon) > 1 - 2\varepsilon$. This means that as ε approaches zero, OR approaches a function f with $\sum_{|S|=1} \hat{f}_S^2 = 1$.

Thus, there are distributions for which the correlation decay under the OR operation approaches zero. This means that we cannot hope to prove a universal bound on correlation decay for every marginal distribution, even in this very simple case. The problem for the general case is that as we repeatedly apply some operation p it could be that the marginals converge to some point where p does not result in correlation decay.

It is useful to note that for the OR operation, the probability that $X = 1$ increases under every application. Thus, as long as the initial distribution has a non-negiligible probability that $X = 1$, we will have that correlation does indeed decay in each step. Though this particular observation applies only to the OR operation, our proof in the general case does rely on the fact that, using only properties of the initial distribution of X we can get bounds on correlation decay in every step. In summary, we are able to achieve correlation decay for arbitrary cyclic operations. We now state our main theorem to this effect.

▶ **Theorem 22** (Correlation Decay). *Let μ be a distribution on $[q]^n$. Let $X_1, \ldots, X_n$ be the jointly distributed $[q]$-valued random variables drawn from μ. Further, let $\rho = \max_i \rho((X_1; X_2; \ldots; X_{i-1}), X_i) < 1$ and λ be the minimum probability of an atom in the marginal distributions $\{\mu_i\}_{i\in[n]}$. For any $\eta > 0$ and $r \geqslant \Omega_q\left(\frac{\log \lambda}{\log \rho} \log^2\left(\frac{qn}{\eta}\right)\right)$, if $p_1, \ldots, p_r$ is a sequence of operations each of which are cyclic terms then,*

$$\|p_1 \otimes p_2 \otimes \ldots \otimes p_r(\mu) - p_1 \otimes p_2 \otimes \ldots \otimes p_r(\mu^\times)\|_1 \leqslant \eta\,.$$

We now give a brief outline of the main ideas of the proof. For a cyclic operation p, the degree-one Fourier coefficients with respect to any distribution are all equal. Suppose that for some probability distribution μ, the operation p has nearly all of its Fourier mass on degree one coefficients. Then $p(x)$ is close to a sum of independent random variables. Therefore, a quantitative version of the Central Limit Theorem (in particular a variant of the Berry-Esseen Theorem), implies that $p(x)$ is close to a Gaussian random variable.

Next, since $p(x)$ is an operation on $[q]$ it only takes q different values. This should then give us a contradiction: a random variable taking only q different values cannot be close to a continuous random variable like a Gaussian. Unfortunately there is a problem with this argument. The error term in the Berry-Esseen theorem depends on the L^3-norm of the independent random variables. Thus, we must control the L^3-norms of the Fourier basis for p under the distribution μ in order for the previous argument to work.

Now the problem is that, even in the case of the OR operation, the L^3-norms of vectors in the Fourier basis can become arbitrarily large as μ changes under repeated applications of the operation. So, we are forced to prove that the elements of the Fourier basis that have high L^3-norm somehow have very small contribution to the correlation. The main idea here is that the correlation of a joint distribution μ is determined by the singular values of a certain linear operator T_μ known as the conditional expectation operator.

We establish a trade-off between the L^3-norm of the singular vectors of T_μ and the correlation contributed by their corresponding singular values. In particular we show that,

for any singular vector v of T_μ with large L^3-norm, the corresponding singular value must be small. This in turn implies that we need only look at the elements of the Fourier basis with small L^3-norm, as all the other elements do not contribute to the correlation of μ.

Our proof relating L^3-norms to singular values relies heavily on the fact that the operator T_μ is *hypercontractive*. Briefly, hypercontractivity is a property that allows us to bound $\|T_\mu f\|_3 \leqslant \|f\|_2$ under certain conditions on μ. If f is a singular vector of T_μ with singular value σ and unit L^2-norm, we then have $\sigma\|f\|_3 \leqslant 1$. This is precisely the sort of trade-off between the L^3-norm of f and the corresponding singular value that we use in our proof. We defer the details of the proof of the theorem to the full version.

5 Soundness of a LP relaxation

In this section, we use correlation decay to give a sufficient condition for when linear programming can be used to solve a CSP with a cyclic polymorphism. For clarity we state and prove everything in this section for CSPs where every constraint has arity two. First we introduce the basic LP relaxation for CSPs of arity two.

Let Λ be a CSP of arity two over the alphabet $[q]$, and $\mathcal{I}$ be an instance of Λ. For every variable X in $\mathcal{I}$ and element $a \in [q]$ we introduce an LP variable $\mu_X(a)$, which can be thought of as the probability that X is assigned a. For every constraint $C_i(X,Y)$ in $\mathcal{I}$ and every pair of elements $a, b \in [q]$ we introduce an LP variable $\mu_{XY}(a,b)$, which can be thought of as the probability that the pair of variables (X,Y) are assigned the values (a,b). The basic LP relaxation for instance $\mathcal{I}$ is then given by the following LP feasibility problem.

BasicLP Relaxation

$\sum_{a\in[q]} \mu_X(a) = 1$	$\forall X$ (μ_X is a probability distribution)
$\sum_{a,b\in[q]} \mu_{XY}(a,b) = 1$	$\forall X,Y$ (μ_{XY} is a probability distribution)
$\sum_{b\in[q]} \mu_{XY}(a,b) = \mu_X(a)$	$\forall b, C_i(X,Y)$ (local consistency for X)
$\sum_{a\in[q]} \mu_{XY}(a,b) = \mu_Y(b)$	$\forall b, C_i(X,Y)$ (local consistency for Y)
$\mu_{XY}(a,b) = 0$	$\forall C_i(X,Y), a, b$ s.t. $C_i(a,b) = 0$ (μ_{XY} satisfies $C_i(X,Y)$)

Proof of Theorem 8. Let p be a cyclic polymorphism of Λ. For each constraint $C_i(X,Y)$, since $G_{\mu_{XY}}$ is connected, Theorem 5 implies that as $k \to \infty$,

$$\|p^{\otimes k}(\mu_{XY}) - p^{\otimes k}(\mu_X) \times p^{\otimes k}(\mu_Y)\|_1 \to 0$$

Now independently sample the value of every variable V from the distribution $p^{\otimes k}(\mu_V)$. The joint distribution of values for every pair (X,Y) is precisely the product distribution $p^{\otimes k}(\mu_X) \times p^{\otimes k}(\mu_Y)$. Thus, for every constraint $C_i(X,Y)$, the distribution of the values for (X,Y) can be made arbitrarily close to the distribution $p^{\otimes k}(\mu_{XY})$ by taking k large enough. Since p is a polymorphism of Λ and μ_{XY} is a distribution on satisfying assignments to $C_i(X,Y)$, we have that $p^{\otimes k}(\mu_{XY})$ is a distribution on satisfying assignments.

Therefore, for large enough k, there will be a non-zero probability that every constraint $C_i(X,Y)$ is satisfied. In particular, this implies that the instance $\mathcal{I}$ is satisfiable. ◀

References

1. Libor Barto and Marcin Kozik. Constraint satisfaction problems of bounded width. In *FOCS*, pages 595–603, 2009. doi:10.1109/FOCS.2009.32.
2. Libor Barto and Marcin Kozik. Absorbing subalgebras, cyclic terms, and the constraint satisfaction problem. *Logical Methods in Computer Science*, 8(1), 2012. doi:10.2168/LMCS-8(1:7)2012.
3. Andrei A. Bulatov. Tractable conservative constraint satisfaction problems. In *LICS*, pages 321–330, 2003. doi:10.1109/LICS.2003.1210072.
4. Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *J. ACM*, 53(1):66–120, 2006. doi:10.1145/1120582.1120584.
5. Andrei A. Bulatov and Víctor Dalmau. A simple algorithm for mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006. doi:10.1137/050628957.
6. Andrei A. Bulatov, Andrei A. Krokhin, and Peter Jeavons. Constraint satisfaction problems and finite algebras. In *ICALP*, pages 272–282, 2000. doi:10.1007/3-540-45022-X_24.
7. Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1998. doi:10.1137/S0097539794266766.
8. Gábor Kun and Mario Szegedy. A new line of attack on the dichotomy conjecture. In *STOC*, pages 725–734, 2009. doi:10.1145/1536414.1536512.
9. Miklós Maróti and Ralph McKenzie. Existence theorems for weakly symmetric operations. *Algebra Universalis*, 59:463–489, 2008. doi:10.1007/s00012-008-2122-9.
10. E. Mossel. Gaussian bounds for noise correlation of functions. *In FOCS'08: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008.
11. Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, 1978. doi:10.1145/800133.804350.

On Percolation and $\mathcal{NP}$-Hardness*

Huck Bennett[†1], **Daniel Reichman**[2], **and Igor Shinkar**[‡3]

1 **Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, USA**
hbennett@cs.nyu.edu

2 **Department of Cognitive and Brain Sciences, University of California, Berkeley, CA, USA**
daniel.reichman@gmail.com

3 **Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York, USA**
ishinkar@cims.nyu.edu

Abstract

The edge-percolation and vertex-percolation random graph models start with an arbitrary graph G, and randomly delete edges or vertices of G with some fixed probability. We study the computational hardness of problems whose inputs are obtained by applying percolation to worst-case instances. Specifically, we show that a number of classical $\mathcal{NP}$-hard graph problems remain essentially as hard on percolated instances as they are in the worst-case (assuming $\mathcal{NP} \not\subseteq \mathcal{BPP}$). We also prove hardness results for other $\mathcal{NP}$-hard problems such as Constraint Satisfaction Problems, where random deletions are applied to clauses or variables.

We focus on proving the hardness of the Maximum Independent Set problem and the Graph Coloring problem on percolated instances. To show this we establish the robustness of the corresponding parameters $\alpha(\cdot)$ and $\chi(\cdot)$ to percolation, which may be of independent interest. Given a graph G, let G' be the graph obtained by randomly deleting edges of G. We show that if $\alpha(G)$ is small, then $\alpha(G')$ remains small with probability at least 0.99. Similarly, we show that if $\chi(G)$ is large, then $\chi(G')$ remains large with probability at least 0.99.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases percolation, NP-hardness, random subgraphs, chromatic number

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.80

1 Introduction

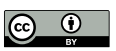
The theory of $\mathcal{NP}$-hardness suggests that we are unlikely to find optimal solutions to $\mathcal{NP}$-hard problems in polynomial time. This theory applies to the worst-case setting where one considers the worst running-time over all inputs of a given length. It is less clear whether these hardness results apply to "real-life" instances. One way to address this question is to examine to what extent known $\mathcal{NP}$-hardness results are stable under random perturbations, as it seems reasonable to assume that a given instance of a problem may be subjected to noise originating from multiple sources.

Recent work has studied the effect of random perturbations of the input on the runtime of algorithms. In their seminal paper Spielman and Teng [28] introduced the idea of *smoothed*

* The full version of this paper is available at http://eccc.hpi-web.de/report/2015/132.

† Huck Bennett was supported by NSF grant CCF 1423228.

‡ Igor Shinkar was supported by NSF grants CCF 1422159, 1061938, 0832795, and a Simons Collaboration on Algorithms and Geometry grant.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 80; pp. 80:1–80:14

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

analysis to explain the superior performance of algorithms in practice compared with formal worst-case bounds. Roughly speaking, smoothed analysis studies the running time of an algorithm on a perturbed worst-case instance. In particular, they showed that subjecting the weights of an arbitrary linear program to Gaussian noise yields instances on which the simplex algorithm runs in expected polynomial time, despite the fact that there are pathological linear programs for which the simplex algorithm requires exponential time. Since then smoothed analysis has been applied to a number of other problems [10, 29].

In contrast to smoothed analysis, we study when worst-case instances of problems remain hard under random perturbations. Specifically, we study to what extent $\mathcal{NP}$-hardness results are robust when instances are subjected to random deletions. Previous work is mainly concerned with *Gaussian* perturbations of *weighted* instances. Less work has examined the robustness of hardness results of unweighted instances with respect to discrete noise.

We focus on two forms of percolation on graphs. Given a graph $G = (V, E)$ and a parameter $p \in (0, 1)$, we define $G_{p,e} = (V, E')$ as the probability space of graphs on the same set of vertices, where each edge $e \in E$ is contained in E' independently with probability p. We say that $G_{p,e}$ is obtained from G by *edge percolation*. We define $G_{p,v} = (V', E')$ as the probability space of graphs, in which every vertex $v \in V$ is contained in V' independently with probability p, and $G_{p,v}$ is the subgraph of G induced by the vertices V'. We say that $G_{p,v}$ is obtained from G by *vertex percolation*. We also study appropriately defined random deletions applied to instances of other $\mathcal{NP}$-hard problems, such as 3-SAT and Subset-Sum.

Throughout we refer to instances that are subjected to random deletions as *percolated instances*. Our main question is whether such percolated instances remain hard to solve by polynomial-time algorithms assuming $\mathcal{NP} \nsubseteq BPP$.

1.1 A first example – 3-Coloring

Consider the 3-Coloring problem, where given a graph $G = (V, E)$ we need to decide whether G is 3-colorable. Suppose that given a graph G we sample a random subgraph G' of G, by deleting each edge of G independently with probability $p = \frac{1}{2}$, and ask whether the resulting graph is 3-colorable. Is there a polynomial time algorithm that can decide with high probability whether G' is 3-colorable?

We demonstrate that a polynomial-time algorithm for deciding whether G' is 3-colorable is impossible assuming $\mathcal{NP} \nsubseteq \mathcal{BPP}$. We show this by considering the following polynomial time reduction from the 3-Coloring problem to itself.

Given an n-vertex graph H the reduction outputs a graph G that is an R-blow-up of H for $R = C\sqrt{\log(n)}$, where $C > 0$ is large enough. That is, replace each vertex of H by a cloud of R vertices that form an independent set in G, and for each edge in H place a complete $R \times R$ bipartite graph in G between the corresponding clouds in G. It is easy to see that H is 3-colorable if and only if G is 3-colorable.

In fact, the foregoing reduction satisfies a stronger *robustness* property for random subgraphs G' of G. Namely, if H is 3-colorable, then G is 3-colorable, and hence G' is also 3-colorable with probability 1. On the other hand, if H is not 3-colorable, then G is not 3-colorable, and with high probability G' is not 3-colorable either.

Indeed, for any edge (v_1, v_2) in H let U_1, U_2 be two clouds in G corresponding to v_1 and v_2. Fixing two arbitrary sets $U_1' \subseteq U_1$ and $U_2' \subseteq U_2$ each of size $R/3$, the probability that there is no edge connecting a vertex from U_1 to a vertex in U_2 is $2^{-R^2/9}$. By union bounding over the $|E| \cdot \binom{R}{R/3}^2 \ll 2^{R^2/9}$ choices of U_1', U_2' we get that there is at least one edge between U_1' and U_2' with high probability. When this holds we can decode any 3-coloring of G' to a 3-coloring of H by coloring each vertex v of H with the color that appears the largest number

of times in the coloring of the corresponding cloud in G', breaking ties arbitrarily. Therefore a polynomial time algorithm for deciding the 3-colorability of G implies a polynomial time algorithm for determining the 3-colorability of H with high probability. It follows that unless $\mathcal{NP} \subseteq \text{co}\mathcal{RP}$ there is no polynomial time algorithm that given a 3-colorable graph G finds a 3-coloring of a random subgraph of G.[1]

Toward a stronger notion of robustness

The example above raises the question of whether the blow-up described above is really necessary. Naïvely, one could hope for stronger hardness of the 3-Coloring problem, namely, that for any graph H if H is not 3-colorable, then with high probability a random subgraph H' of H is not 3-colorable either. However, this is not true in general, as H can be a 3-critical graph, i.e., a 3-colorable graph such that deletion of *any* edge of H decreases its chromatic number (consider for example the case of an odd cycle).

Nonetheless, if random deletions do not decrease the chromatic number of a graph by much, then one could use hardness of approximation results for chromatic number to deduce hardness results for coloring percolated graphs. In this paper we show that the chromatic number of a graph is indeed robust to random deletions. We show that if we delete each edge of a graph with probability $\frac{1}{2}$, then (with probability 0.99) the chromatic number does not drop by much.

We also consider the question of robustness for other graph parameters. For independent sets we demonstrate that if the independence number of G is small, then with high probability the independence number of a random subgraph of G is small as well. Similarly, we show that for a k-SAT formula that is sufficiently dense, randomly deleting its clauses does not change the maximum possible fraction of clauses that can be satisfied simultaneously. In particular, this implies that these problems remain essentially as hard on percolated instances as they are on worst-case instances.

▶ Remark. It is worth noting that there are graph parameters for which percolated instances differ significantly from the original instance. For example, standard results in random graph theory imply that for every n-vertex graph G, with high probability the size of the largest clique in the graph G' obtained by edge percolation with $p = \frac{1}{2}$ is $O(\log n)$. In particular, a maximum clique in G' can be found in time $n^{O(\log n)}$, which is significantly faster than the fastest known algorithm for finding a maximum clique in the worst-case.

1.2 Robustness of $\mathcal{NP}$-hard problems under percolation

In proving hardness results for percolated instances we use the concept of *robust reductions* which we explain below. It will be convenient to consider promise problems[2]. We start by introducing the following definition.

▶ **Definition 1.** Let $A = (A_{YES}, A_{NO})$ and $B = (B_{YES}, B_{NO})$ be two promise problems. For each $y \in \{0,1\}^*$ (an instance of the problem B) let $\mathsf{noise}(y)$ be a distribution on $\{0,1\}^*$, that is samplable in time $\text{poly}(|y|)$.

[1] Note that in the foregoing example, if we start with a bounded degree graph H, we can reduce it to a bounded degree graph G by using an $R \times R$ bipartite expander instead of the complete bipartite graph.

[2] Recall, that a promise problem is a generalization of a decision problem, where for the problem L there are two disjoint subsets L_{YES} and L_{NO}, such that an algorithm that solves L must accept all the inputs in L_{YES} and reject all inputs in L_{NO}. If the input does not belong to $L_{YES} \cup L_{NO}$, there is no requirement on the output of the algorithm.

- A polynomial time reduction R from A to B is said to be *noise-robust* if
  1. For all $x \in A_{YES}$ it holds that $\mathsf{R}(x) \in B_{YES}$, and $\Pr[\mathsf{noise}(\mathsf{R}(x)) \in B_{YES}] > 0.99$.
  2. For all $x \in A_{NO}$ it holds that $\mathsf{R}(x) \in B_{NO}$, and $\Pr[\mathsf{noise}(\mathsf{R}(x)) \in B_{NO}] > 0.99$.
- If in the first item we have $\Pr[\mathsf{noise}(\mathsf{R}(x)) \in B_{YES}] = 1$, then we say that R is a noise-robust co$\mathcal{RP}$-reduction. Similarly, if in the second item we have $\Pr[\mathsf{noise}(\mathsf{R}(x)) \in B_{NO}] = 1$, then we say that R is a noise-robust $\mathcal{RP}$-reduction.
- The problem $B = (B_{YES}, B_{NO})$ is said to be *$\mathcal{NP}$-hard under a noise-robust reduction* if there exists a *noise-robust* reduction from an $\mathcal{NP}$-hard problem to B.
- We say that the problem A is *strongly-noise-robust* to B if
  1. For all $x \in A_{YES}$ it holds that $x \in B_{YES}$, and $\Pr[\mathsf{noise}(x) \in B_{YES}] > 0.99$.
  2. For all $x \in A_{NO}$ it holds that $x \in B_{NO}$, and $\Pr[\mathsf{noise}(x) \in B_{NO}] > 0.99$.

Note that in the last item of Definition 1 there is no reduction involved. Instead, we think of the problem A as a relaxation of B with $A_{YES} \subseteq B_{YES}$ and $A_{NO} \subseteq B_{NO}$, and hence any algorithm that solves B in particular solves A. However, it is a relaxed problem in a stronger sense, namely, after applying noise to a YES-instance (resp. NO-instance) of A, it stays a YES-instance (resp. NO-instance) of B with high probability.

We use the term noise-robust to avoid confusion with other notions of robust reductions that have appeared in the literature. In order to ease readability, we will often write robust reductions instead, always referring to noise-robust reductions as defined above.

▶ **Proposition 2.** *Let $L = (L_{YES}, L_{NO})$ be a promise problem, and for each y instance of L, let $\mathsf{noise}(y)$ be a distribution on instances of L that is samplable in time* $\mathrm{poly}(|y|)$.

If L is $\mathcal{NP}$-hard under a noise-robust reduction, then there is no polynomial time algorithm that when given an input y decides with high probability whether $\mathsf{noise}(y) \in L_{YES}$ or $noise(y) \in L_{NO}$, unless $\mathcal{NP} \subseteq \mathcal{BPP}$.

Indeed, the example given in Section 1.1 gives a noise-robust reduction from the 3-Coloring problem to itself, where noise refers to random deletions of the edges in a given graph. Therefore, the 3-Coloring problem is $\mathcal{NP}$-hard under a noise-robust reduction.

1.3 Our results

In this paper we show that a number of $\mathcal{NP}$-hard problems remain hard to solve even after random deletions, i.e., they are $\mathcal{NP}$-hard under noise-robust reductions. Furthermore, we show that some gap $\mathcal{NP}$-hard problems are, in fact, strongly-noise-robust to the same problems with a smaller gap. Specifically, we focus on showing these results for the gap versions of the maximum independent set and chromatic number problems. As technical tools, we prove a number of combinatorial results about the independence number and the chromatic number of percolated graphs that might be of independent interest.

Maximum Independent Set and Percolation

▶ **Theorem 3.** *Let $G = (V, E)$ be an n-vertex graph. Then, with high probability $\alpha(G_{p,e}) \leq O\left(\frac{\alpha(G)}{p}\log(np)\right)$.*

We observe that in general, the upper bound above cannot be improved, as it is well known that the independence number of $G(n,p)$ is $\Omega\left(\frac{\log(np)}{p}\right)$ with high probability (see, e.g., [4]).

In the Coloring-vs-MIS(q,a) problem, given an n-vertex graph G such that $q \cdot a \geq n$, the goal is to distinguish between the YES-case where $\chi(G) \leq q$ and the NO-case where

$\alpha(G) \leq a$. By using Theorem 3 together with the inapproximability results of Feige and Kilian [11] saying that for every $\varepsilon > 0$ it is $\mathcal{NP}$-hard to decide whether a given n-vertex graph G satisfies $\chi(G) \leq n^{\varepsilon}$ or $\alpha(G) \leq n^{\varepsilon}$ we obtain the following hardness result.

▶ **Theorem 4.** *For any q, a the Coloring-vs-MIS(q, a) problem is strongly-noise-robust to Coloring-vs-MIS$\left(q, O\left(\frac{a}{p}\log(np)\right)\right)$, where n denotes the number of vertices in the given graph, and noise is the p-edge-percolation of this graph.*

In particular, for any constant $\varepsilon > 0$, unless $\mathcal{NP} \subseteq \mathcal{BPP}$ there is no polynomial time algorithm that given an n-vertex graph G approximates either $\alpha(G_{p,e})$ or $\chi(G_{p,e})$ within a $\frac{1}{pn^{1-2\varepsilon}}$ (resp. $pn^{1-2\varepsilon}$) factor for any $p > \frac{1}{n^{1-2\varepsilon}}$.

We also prove analogous theorems for vertex percolation.

Graph Coloring and Percolation

Theorem 3 says that it is hard to approximate the chromatic number of a percolated graph within a $n^{1-\varepsilon}$ factor, but says nothing about hardness of coloring percolated graphs with small (constant) chromatic number. We address this question below by proving lower bounds[3] on the chromatic number of percolated graphs. To do this we use results from additive combinatorics and discrete Fourier analysis.

▶ **Theorem 5.** *Let $G = (V, E)$ be an n-vertex graph. Then, for every $\alpha \in (0, 1)$ it holds that $\Pr[\chi(G_{\frac{1}{2},v}) \geq \max\{\chi(G)/3 - O_\alpha(1), \chi(G)/2 - O_\alpha(\sqrt{n})\}] > 1 - \alpha$.*

▶ **Theorem 6.** *Let $G = (V, E)$ be an n-vertex graph with m edges. Then, for every $\alpha \in (0, 1)$ it holds that $\Pr[\chi(G_{\frac{1}{2},e}) \geq \max\{\Omega_\alpha(\chi(G)^{1/3}), \Omega_\alpha(\chi(G)/m^{1/4})\}] > 1 - \alpha$.*

For $G_{\frac{1}{2},v}$ the $\chi(G)/2 - O_\alpha(\sqrt{n})$ lower bound is better when $\chi(G) = \omega(\sqrt{n})$, and the $\chi(G)/3 - O_\alpha(1)$ lower bound is better when $\chi(G) = o(\sqrt{n})$. For $G_{\frac{1}{2},e}$ the $\Omega_\alpha(\chi(G)/m^{1/4})$ lower bound is better when $\chi(G) = \omega(m^{3/8})$, and the $\Omega_\alpha(\chi(G)^{1/3})$ lower bound is better when $\chi(G) = o(m^{3/8})$.

Note that this result also gives lower bounds on the chromatic number of $G_{p,v}, G_{p,e}$ where $p \neq \frac{1}{2}$ by composing the bounds in Theorems 5 and 6 $\lceil \log_2(1/p) \rceil$ times.

▶ Remark. Bukh [7] has considered coloring edge-percolated graphs, and states the question of whether $\mathbf{E}[\chi(G_{\frac{1}{2},e})] = \Omega(\chi(G)/\log(\chi(G)))$ as an "interesting problem." Bukh observed that the chromatic number of $G_{\frac{1}{2},e}$ has the same distribution as the chromatic number of the complement of $G_{\frac{1}{2},e}$, and therefore $\mathbf{E}[\chi(G_{\frac{1}{2},e})] \geq \sqrt{\chi(G)}$. However, it is not clear how to leverage the lower bound on the expectation to obtain a lower bound on $\chi(G_{\frac{1}{2},e})$ *with high probability*, which is required for our noise robust reductions. Moreover, for $k \ll \sqrt{n}$ standard martingale methods do not seem to work for showing high probability estimates.

In the Gap-Coloring(q, Q) problem we are given an n-vertex graph G and the goal is to distinguish between the YES-case where G is q-colorable, and the NO-case where the chromatic number of G is at least Q. There is a large body of work proving hardness results for this problem [14, 20, 18] including stronger results assuming variants of the Unique Games Conjecture [8, 9]. Using the $\mathcal{NP}$-hardness of the Gap-Coloring$(q, \exp(\Omega(q^{1/3})))$ problem of Huang [18] we obtain an analogous hardness result under noise-robust reductions for this problem.

[3] The notation $O_\alpha(f(n))$ means that $O(f(n))$ holds for fixed α.

▶ **Theorem 7.** *For all $q < Q$ the Gap-Coloring(q, Q) problem is strongly-noise-robust to the Gap-Coloring$(q, \Omega(Q^{1/3}))$ problem, where noise is $\frac{1}{2}$-edge-percolation applied to the graph.*

In particular, for any sufficiently large constant q given a q-colorable graph G it is $\mathcal{NP}$-hard to find a $2^{\Omega(q^{1/3})}$-coloring of $G_{\frac{1}{2},e}$.

Satisfiability and Other Problems

We also state a hardness result for approximating the value of a clause-percolated instance of k-SAT. A k-SAT formula Φ is a collection of m clauses on n Boolean variables, where each clause is an OR of k-literals. Given a formula Φ, and an assignment σ to its variables, denote by $val_\sigma(\Phi)$ the fraction of constraints of Φ satisfied by σ. The value of Φ is defined as $val(\Phi) = \max_\sigma val_\sigma(\Phi)$. If $val(\Phi) = 1$ we say that Φ is satisfiable.

Given an instance Φ of k-SAT its *clause percolation* is a random formula Φ^c_p over the same set of variables, obtained from Φ by keeping each clause of Φ independently with probability p.

▶ **Theorem 8.** *Let $\varepsilon, \delta \in (0, 1)$ be fixed constants. Then, unless $\mathcal{NP} \subseteq \text{co}\mathcal{RP}$, there is no polynomial time algorithm that when given a satisfiable instance Φ over n-variables of 3-SAT, finds an assignment σ to Φ^c_p such that $val_\sigma(\Phi^c_p) > 7/8 + \varepsilon$ for all $p > \frac{1}{n^{2-\delta}}$.*

One ingredient of the proof of Theorem 8, that may be of independent interest, is establishing that k-SAT does not admit a non-trivial approximation on dense formulas that contain $n^{k-\eta}$ clauses, where $\eta > 0$ is an arbitrary small positive constant.

We prove analogous theorems also for other CSP's as well as other graph theoretic problems such as Vertex Cover and Directed Hamiltonian Cycle. We also prove hardness results for the percolated Subset Sum problem. The exact statements and complete proofs, including of Theorem 8, appear in the full version of the paper.

1.4 Preliminaries

An *independent set* in a graph $G = (V, E)$ is a set of vertices that spans no edges. The *independence number* $\alpha(G)$ denotes the maximum size of an independent set in G. A *legal coloring* of a graph G is an assignment of colors to vertices such that no two adjacent vertices share the same color. The *chromatic number* $\chi(G)$ denotes the minimum number of colors needed for a legal coloring of G. Note that in a legal coloring of G each color class forms an independent set, and hence $\chi(G) \cdot \alpha(G) \geq n$.

We will always measure the running time of algorithms in terms of the size of the percolated instance. Since G and $G_{p,e}$ have the same number of vertices, this generally does not affect the size of the instance by more than a polynomial factor. On the other hand, $G_{p,v}$ may be much smaller than G for very small values of p. However, in this work we will be only dealing with the case where $p = \frac{1}{n^{1-\Omega(1)}}$, hence with high probability the size of the vertex percolated and original graphs are polynomially related as well.

We will use the following version of the Chernoff bound.

▶ **Lemma 9** (Chernoff bound, Theorem 7.3.2 in [17])**.** *Let $x_1, \ldots, x_n$ be independent Bernoulli trials with $\Pr[x_i = 1] = p$, and let $\mu = \mathbf{E}[\sum_{i=1}^n x_i] = pn$. Let $r \geq e^2$. Then $\Pr[\sum_{i=1}^n x_i > (1 + r)\mu] < \exp(-(\mu r/2) \ln r)$.*

1.5 Related Work

There is a wide body of work on random discrete structures that has produced a wide range of mathematical tools [4, 13, 15, 23]. Randomly subsampling subgraphs by including each

edge independently in the sample with probability p has been studied extensively in works concerned with cuts and flows (e.g., [19]). More recently, sampling subgraphs has been used to find independent sets [12]. The effect of subsampling variables in mathematical relaxations of constraint satisfaction problems on the value of these relaxations was studied in [2].

Edge-percolated graphs have been also used to construct hard instances for specific algorithms. For example, Kučera [21] proved that the well known greedy coloring algorithm performs poorly on the complete r-partite graph in which every edge is removed independently with probability $1/2$ and $r = n^{\varepsilon}$ for $\varepsilon > 0$. Namely, for this graph G, even if vertices are considered in a random order by the greedy algorithm, with high probability $\Omega(\frac{n}{\log n})$ colors are used to color the percolated graph whereas $\chi(G) \leq n^{\varepsilon}$.

Misra [24] studies edge percolated instances of the Max-Cut problem. He proves that in graphs of fixed maximal degree d it is impossible (assuming $\mathcal{NP} \neq \mathcal{BPP}$) to compute the size of the maximum cut in $G_{p,e}$ in polynomial time whenever $p = \frac{1+\varepsilon}{d-1}$. The techniques used in [24] differ from ours and rely on the recent hardness result for counting independent sets in sparse graphs [27].

The chromatic number of Erdős-Rényi random graphs $G(n,p)$ has been studied extensively. Grimmett and McDiarmid [16] showed that for a fixed p with high probability it holds that $\chi(G(n,p)) = \Theta(\log(1/1-p)\frac{n}{\log n})$. Bollobás [3] later determined the right constant, proving that $\chi(G(n,p)) \sim \log(1/(1-p))\frac{n}{2\log(n)}$ for every $p \in (0,1)$. Łuczak [22] further improved the previous result by showing that it holds for subconstant values of p. In this paper we study the independence number and chromatic number of general percolated graphs. A recent paper by Bollobás et al. [5] studied a special case of this, namely the independence number of edge percolated Kneser graphs.

2 Maximum Independent Set and Percolation

In this section we demonstrate the hardness of approximating $\alpha(G)$ and $\chi(G)$ in both edge percolated and vertex percolated graphs. We base our results on a theorem of Feige and Kilian, saying that for every fixed $\varepsilon > 0$ the problem Coloring-vs-MIS$(n^{\varepsilon}, n^{\varepsilon})$ is $\mathcal{NP}$-hard.

▶ **Theorem 10** ([11]). *For every $\varepsilon > 0$ it is $\mathcal{NP}$-hard to decide whether a given n-vertex graph G satisfies $\chi(G) \leq n^{\varepsilon}$ or $\alpha(G) \leq n^{\varepsilon}$.*

Edge percolation

Below we prove Theorem 3. We will use the following lemma, due to Turan (see, e.g. [1]).

▶ **Lemma 11.** *Every graph H with l vertices and e edges contains an independent set of size at least $\frac{l^2}{2e+l}$.*

As a corollary we observe that if a graph contains no large independent sets, then it can also cannot contain large subsets of the vertices that span a small number of edges.

▶ **Corollary 12.** *Let $G = (V,E)$ be an n-vertex graph satisfying $\alpha(G) < k$. Then every set of vertices of size $l \geq k$ spans at least $l(l-k)/2k$ edges.*

Proof. Let H be a subgraph of G induced by l vertices, and suppose that H spans e edges. Then, by Lemma 11 we have $\alpha(H) \geq \frac{l^2}{2e+l}$. On the other hand, $\alpha(H) \leq \alpha(G) \leq k$, and hence $\frac{l^2}{2e+l} \leq k$, as required. ◀

We are now ready to prove Theorem 3 saying that for any n-vertex graph $G = (V,E)$ it holds that with high probability $\alpha(G_{p,e}) \leq O\left(\frac{\alpha(G)}{p}\log(np)\right)$.

Proof of Theorem 3. For a given graph G, let $k = \alpha(G)+1$ and let $C > 0$ be a large enough constant. By Corollary 12, every subset of size $l = C\frac{\alpha(G)}{p}\log(np)$ spans at least $\frac{l(l-k)}{2k}$ edges in G. Hence, by taking union bound over all subsets of size l, the probability there exists a set of size l in $G_{p,e}$ that spans no edge is at most

$$\binom{n}{l}\cdot(1-p)^{\frac{l(l-k)}{2k}} < \left(\frac{en}{l}\right)^{l}\cdot\exp\left(-p\cdot\frac{l(l-k)}{2k}\right) < (np)^{-\Omega(l)},$$

where the last inequality uses the choices of l and k, implying that $\left(\frac{en}{l}\right)^{l} < (np)^{l}$ and $\exp(-p\frac{l(l-k)}{2k}) < \exp(-\Omega(l\cdot\log(np))) = (np)^{-\Omega(l)}$. Therefore, with high probability $\alpha(G_{p,e}) \leq C\frac{\alpha(G)}{p}\log(np)$. ◀

Theorem 4 follows immediately from Theorem 3.

Proof of Theorem 4. Let G be an instance G of the Coloring-vs-MIS(q,a) problem. Note that for the YES-case if $\chi(G) \leq q$, then clearly $\chi(G_{p,e}) < q$. For the NO-case by Theorem 3 if $\alpha(G) \leq a$, then with high probability $\alpha(G_{p,e}) \leq O\left(\frac{a}{p}\log(np)\right)$ which implies the strongly-noise-robust hardness.

The "in particular" part follows immediately from Theorem 10. ◀

▶ Remark. Note that for constant $p > 0$ (e.g., $p = 1/2$) this theorem establishes inapproximability for the independence number of $G_{p,e}$ that matches the inapproximability for the worst case.

▶ Remark. Note also that for $p > \frac{1}{n^{1-\varepsilon}}$ (in fact, for $p > \frac{\log(n)}{n}$) such random percolated graphs have maximal degree at most $O(pn)$ with high probability. Therefore, such graphs $G_{p,e}$ can be colored efficiently using $O(pn)$ colors. In particular, with high probability $G_{p,e}$ contains an independent set of size $\Omega(1/p)$ and hence, the independence number can be approximated within a factor of $1/pn$ on p-percolated instances.

Vertex percolation

Next we handle vertex percolation. We show that approximating $\alpha(G)$ and $\chi(G)$ on percolated instances is essentially as hard as worst-case instances, where the vertices remain with probability $p > \frac{1}{n^{1-\delta}}$, where n is the number of vertices in the graph for any constant $\delta \in (0,1)$. We do it again by relying on the hardness of the gap problem Coloring-vs-MIS for percolated instances.

Note that in the case of vertex percolation, the (in)approximability guarantee should depend on the number of vertices in the percolated graph $G_{p,v}$, and not on the number in the original graph.

▶ **Theorem 13.** *The Coloring-vs-MIS(q,a) problem is strongly-noise-robust to itself, where noise is the vertex percolation with parameter any $p > 0$.*

In particular, for any $\delta, \varepsilon > 0$ unless $\mathcal{NP} \subseteq \mathcal{BPP}$ there is no polynomial time algorithm that approximates either $\alpha(G_{p,v})$ or $\chi(G_{p,v})$ within a factor $m^{1-\varepsilon}$ for constant any $\varepsilon > 0$, where m denotes the number of vertices in $G_{p,v}$, and any $p > \frac{1}{n^{1-\delta}}$.

Proof. The strong robustness of Coloring-vs-MIS(q,a) is clear, since for any graph G if G' is a vertex induced subgraph of G, then $\chi(G') \leq \chi(G)$, and $\alpha(G') \leq \alpha(G)$, which is, in particular, true for $G' = G_{p,v}$.

For the "in particular" part, for a given $p > \frac{1}{n^{1-\delta}}$ let $c = \frac{\log(pn)}{\log(n)} \in (\delta, 1)$ so that $p = \frac{1}{n^{1-c}}$, and let $\eta = \varepsilon \cdot c$.

Let G be an n-vertex graph, let $G_{p,v}$ be it vertex-percolated subgraph, and let m be the number of vertices in $G_{p,v}$. By the Chernoff bound in Lemma [17], with high probability we have $|m - pn| < 0.1pn$, and so, we assume from now on that $n^\eta < 2m^\varepsilon$.

By Theorem 10 it is $\mathcal{NP}$-hard to decide whether a given n-vertex graph G satisfies $\chi(G) \leq n^\eta$ or $\alpha(G) \leq n^\eta$. By the choice of parameters, if $\chi(G) \leq n^\eta$ then $\chi(G_{p,v}) \leq n^\eta < 2m^\varepsilon$, and similarly, if $\alpha(G) \leq n^\eta$ then $\alpha(G_{p,v}) < n^\eta < 2m^\varepsilon$. This completes the proof of the theorem. ◀

3 Graph Coloring and Percolation

We present our results in terms of the *maximum coverage problem* [30], which is a variant of the set cover problem, and show later how graph coloring is related to maximum coverage.

3.1 Maximum Coverage

In the maximum coverage problem we are given a family of sets $\mathcal{F} = \{S_1, \ldots, S_m\}$ with $S_i \subseteq [n]$ and a number c. The goal is to find c sets in $\mathcal{F}$ such the cardinality of the union of these c sets is as large as possible. We will make use of the representation of a set S in terms of its incidence vector $x(S) \in \{0,1\}^n$. In this way, we can reformulate the maximum coverage problem as follows. Given $A \subseteq \mathbb{F}_2^n$, find elements $y_1, \ldots, y_c \in A$ that maximize $\|\vee_{i=1}^c y_i\|_1$, the Hamming weight of the bitwise-OR of the vectors.

We will prove two existential results saying that if A is of constant density $\alpha > 0$, then there exists a good cover using only 2 or 3 vectors.

▶ **Lemma 14.** *Let $A \subseteq \mathbb{F}_2^n$ with $|A| \geq \alpha 2^n$. Then there exist $y_1, y_2, y_3 \in A$ such that $\|y_1 \vee y_2 \vee y_3\|_1 \geq n - 4/\alpha^3$.*

▶ **Lemma 15.** *Let $A \subseteq \mathbb{F}_2^n$ with $|A| \geq \alpha 2^n$. Then there exist $y_1, y_2 \in A$ such that $\|y_1 \vee y_2\|_1 \geq n - (1+r)\sqrt{n}$, where $r = \max\{e^2, 2\ln 1/\alpha\}$.*

3.2 Proof of Lemma 14 using additive combinatorics

Lemma 14 follows almost immediately from a result about sumsets. Recall that the Minkowski sum of two sets A, B is defined as $A + B = \{x + y : x \in A, y \in B\}$.

▶ **Lemma 16** (Corollary 3.5 in [26]). *Let $A \subseteq \mathbb{F}_2^n$ with $|A| \geq \alpha 2^n$. Then $A + A + A$ contains an affine subspace of dimension at least $n - 4/\alpha^3$.*

Because an affine subspace of dimension at least $n - 4/\alpha^3$ must contain an element of Hamming weight at least $n - 4/\alpha^3$, Lemma 14 follows from Lemma 16 and the observation that $\|\sum_{i=1}^c y_i\|_1 \leq \|\bigvee_{i=1}^c y_i\|_1$.

3.3 Proof of Lemma 15 using Fourier analysis

We use an inequality from Fourier analysis to give a proof of Lemma 15 via the probabilistic method.

▶ **Definition 17.** Given $x \in \mathbb{F}_2^n$, define $y \sim N_\rho(x)$ by letting each y_i be equal to x_i with probability $\frac{1+\rho}{2}$, and be equal to $1 - x_i$ with probability $\frac{1-\rho}{2}$.

Let $\mathrm{Uni}(S)$ denote the uniform distribution on a set S, and let U_n denote $\mathrm{Uni}(\mathbb{F}_2^n)$. The following lemma is a corollary of the reverse Bonami-Beckner inequality.

▶ **Lemma 18** (Corollary 3.5 in [25]). *Let $A, B \subseteq \mathbb{F}_2^n$ with $|A| = |B| = \alpha 2^n$. Then*

$$\Pr_{\substack{x \leftarrow Uni(A) \\ y \leftarrow N_\rho(x)}}[y \in B] \geq \alpha^{(1+\rho)/(1-\rho)}.$$

Proof of Lemma 15. Let $A \subseteq \mathbb{F}_2^n$ with $|A| \geq \alpha 2^n$, and let $B = A + \vec{1} = \{x + \vec{1} : x \in A\}$, where $\vec{1}$ is the n-dimensional all 1s vector. Note that to prove Lemma 15 it suffices to show that there exist $x \in A, y \in B$ such that $\|x + y\|_1 = (1 + r) \cdot \sqrt{n}$, since then $y + \vec{1} \in A$ and $\|x + (y + \vec{1})\|_1 = n - (1 + r) \cdot \sqrt{n}$.

Let $\varepsilon = 1/\sqrt{n}$ and let $\rho = 1 - 2\varepsilon$. By Lemma 18,

$$\Pr_{\substack{x \leftarrow U_n \\ y \leftarrow N_\rho(x)}}[x \in A, y \in B] = \Pr_{\substack{x \leftarrow \mathrm{Uni}(A) \\ y \leftarrow N_\rho(x)}}[y \in B] \cdot \Pr_{x \leftarrow U_n}[x \in A] \geq \alpha^{2/(1-\rho)} = \alpha^{\sqrt{n}}. \tag{1}$$

Set $r = \max\{e^2, 2\ln(1/\alpha)\}$. Note that by definition of $y \sim N_\rho(x)$ we have that $\Pr[x_i \neq y_i] = 1/\sqrt{n}$ for each i independently. Therefore, by the Chernoff bound in Lemma 9,

$$\Pr_{\substack{x \leftarrow U_n \\ y \leftarrow N_\rho(x)}}[\|x + y\|_1 \leq (1 + r)\sqrt{n}] \geq 1 - e^{-(r/2 \ln r)\sqrt{n}} \geq 1 - \alpha^{2\sqrt{n}}. \tag{2}$$

Since the sum of the probabilities in Equations (1) and (2) is strictly greater than 1, the corresponding events cannot be disjoint. Hence there exist $x \in A, y \in B$ such that $\|x + y\|_1 \leq (1 + r)\sqrt{n}$. ◀

3.4 Coloring Using Subgraphs

We now show how to apply the results in the previous subsection to the graph coloring problem. Throughout this section we let $G = (V, E)$ with $n = |V|, m = |E|$. We will identify the elements of $[n]$ with vertices V in the vertex percolation case and the elements of $[m]$ with edges E in the edge percolation case. Let $G_{|U}$ denote the subgraph of G induced by $U \subseteq V$.

▶ **Lemma 19.** *Let $G = (V, E)$ and let $V_1, V_2 \subseteq V$ with $V_1 \cup V_2 = V$. If $\chi(G_{|V_1}) \leq k_1$ and $\chi(G_{|V_2}) \leq k_2$ then $\chi(G) \leq k_1 + k_2$.*

Proof. Assume that $V_1 \cap V_2 = \emptyset$ (if not, replace V_1 with $V_1 \setminus V_2$ in the following argument). Color $G_{|V_1}$ with k_1 colors and color $G_{|V_2}$ with k_2 fresh colors. Because $G_{|V_1}$ and $G_{|V_2}$ are colored with separate colors any edges between V_1 and V_2 have endpoints with distinct colors. ◀

▶ **Lemma 20.** *Let $G = (V, E)$, let $E_1, E_2 \subseteq E$ with $E_1 \cup E_2 = E$, and let $G_1 = (V, E_1), G_2 = (V, E_2)$. If $\chi(G_1) \leq k_1$ and $\chi(G_2) \leq k_2$ then $\chi(G) \leq k_1 k_2$.*

Proof. Let c_1 be a coloring of G_1 with k_1 colors, and let c_2 be the coloring of G_2 with k_2 colors. We claim that the coloring as $c(v) = (c_1(v), c_2(v))$ is a legal coloring of G with $k_1 k_2$ colors. Consider an edge $e = (u, v) \in E$. If $e \in E_1$ then $c(u)$ differs from $c(v)$ in the fist coordinate. Otherwise $e \in E_2$ in which case $c(u)$ differs from $c(v)$ in the second coordinate. ◀

3.5 Lower Bounding the Chromatic Number

We now prove lower bounds on the chromatic number of percolated graphs. We will consider both vertex and edge percolation with $p = \frac{1}{2}$. This choice of p is important because $G_{\frac{1}{2},v}$,

$G_{\frac{1}{2},e}$ become the distributions of graphs induced by uniformly random subsets of V and E, respectively. However, we also obtain bounds for $p < \frac{1}{2}$ by composing the bounds for $p = \frac{1}{2}$. When stating bounds based on Lemma 15 we set $r = \max\{e^2, 2\ln(1/\alpha)\}$.

The idea will be to argue that if many subgraphs of a graph G are k-colorable then G is colorable with $f(k)$ colors for relatively small $f(k)$. To see how this idea works, consider the following easy case. Suppose that $\Pr[\chi(G_{\frac{1}{2},v}) \leq k] > \frac{1}{2}$. Then there exists $V' \subseteq V$ such that $G_{|V'}$ and $G_{|\overline{V'}}$ are both k-colorable. It follows that G is $2k$-colorable by Lemma 19. We now consider the case where the density of k colorable subgraphs α is less than $\frac{1}{2}$.

Proof of Theorem 5

We now prove Theorem 5, saying that if G is an n-vertex graph, then for every $\alpha \in (0,1)$ it holds that $\Pr[\chi(G_{\frac{1}{2},v}) \geq \max\{\chi(G)/3 - O_\alpha(1), \chi(G)/2 - O_\alpha(\sqrt{n})\}] > 1 - \alpha$. The proof relies on the following two lemmas.

▶ **Lemma 21.** $\Pr[\chi(G_{\frac{1}{2},v}) \leq k] \geq \alpha \Rightarrow \chi(G) \leq 3k + 4/\alpha^3$.

Proof. Identify subsets of vertices V with vectors in $\mathbb{F}_2^n$. Because $\Pr[\chi(G_{\frac{1}{2},v}) \leq k] \geq \alpha$ by Lemma 14 there exist $V_1, V_2, V_3 \subseteq V$ such that each $G_{|V_i}$ is k-colorable and $|V_1 \cup V_2 \cup V_3| \geq n - 4/\alpha^3$. Using Lemma 19, we can then color $G_{|V_1 \cup V_2 \cup V_3}$ with $3k$ colors. Coloring the remaining $4/\alpha^3$ nodes each with a different, new color implies the lemma. ◀

▶ **Lemma 22.** $\Pr[\chi(G_{\frac{1}{2},v}) \leq k] \geq \alpha \Rightarrow \chi(G) \leq 2k + (1+r)\sqrt{n}$.

Proof. Identify subsets of vertices V with vectors in $\mathbb{F}_2^n$. Because $\Pr[\chi(G_{\frac{1}{2},v}) \leq k] \geq \alpha$ by Lemma 15 there exist $V_1, V_2 \subseteq V$ such that $G_{|V_1}, G_{|V_2}$ are k-colorable and $|V_1 \cup V_2| \geq n - (1+r)\sqrt{n}$. Using Lemma 19, we can then color $G_{|V_1 \cup V_2}$ with $2k$ colors. Coloring the remaining $(1+r)\sqrt{n}$ nodes each with a different, new color implies the lemma. ◀

Lemmas 21 and 22 imply Theorem 5. Taking the contrapositive of Lemma 21 we get

$$\chi(G) > 3k + 4/\alpha^3 \Rightarrow \Pr[\chi(G_{\frac{1}{2},v}) > k] \geq 1 - \alpha,$$

which is equivalent to

$$\chi(G) > \ell \Rightarrow \Pr[\chi(G_{\frac{1}{2},v}) > (\ell - 4/\alpha^3)/3] \geq 1 - \alpha.$$

Similarly, taking the contrapositive of Lemma 22 we get the bound

$$\chi(G) > \ell \Rightarrow \Pr[\chi(G_{\frac{1}{2},v}) > \ell/2 - O_\alpha(\sqrt{n})] \geq 1 - \alpha.$$

Proof of Theorem 6

Next we prove Theorem 6, saying that if G is an n-vertex graph with m edges, then for every $\alpha \in (0,1)$ it holds that $\Pr[\chi(G_{\frac{1}{2},e}) \geq \max\{\Omega_\alpha(\chi(G)^{1/3}), \Omega_\alpha(\chi(G)/m^{1/4})\}] > 1 - \alpha$. The techniques are similar to those used for proving Theorem 5, but with several additional observations we push the techniques further.

▶ **Lemma 23.** $\Pr[\chi(G_{\frac{1}{2},e}) \leq k] \geq \alpha \Rightarrow \chi(G) \leq k^3 + 8/\alpha^3$.

Proof. Identify subsets of edges E with vectors in $\mathbb{F}_2^m$. Because $\Pr[\chi(G_{\frac{1}{2},e}) \leq k] \geq \alpha$ by Lemma 14 there exist $E_1, E_2, E_3 \subseteq E$ such that each $G_i = (V, E_i)$ is k-colorable and $|E_1 \cup E_2 \cup E_3| \geq m - 4/\alpha^3$. Using Lemma 20, we can then color $G(V, E_1 \cup E_2 \cup E_3)$ with k^3 colors. Color the endpoints of the remaining $E \setminus (E_1 \cup E_2 \cup E_3)$ edges using $8/\alpha^3$ new colors to achieve a $(k^3 + 8/\alpha^3)$-coloring of G. ◀

The next lemma gives an unconditional upper bound on chromatic number.

▶ **Lemma 24.** *Let $G = (V, E)$ be a graph with $|E| = m$. Then $\chi(G) \leq 3\sqrt{m} + 1$.*

Proof. Partition V into sets $V_0 = \{v \in V : \deg(v) < \sqrt{m}\}$ and $V_1 = \{v \in V : \deg(v) \geq \sqrt{m}\}$. By Brooks' Theorem [6], $\chi(G_{|V_0}) \leq \max_{v \in V_0} \deg(v) + 1 \leq \sqrt{m} + 1$. Furthermore, because $\sum_{v \in V_1} \deg(v) \leq 2m$, it follows that $|V_1| \leq 2\sqrt{m}$, and in particular $\chi(G_{|V_1}) \leq 2\sqrt{m}$. The result follows by Lemma 19. ◀

We use a variant of the same partitioning trick in the following lemma.

▶ **Lemma 25.** $\Pr[\chi(G_{\frac{1}{2},e}) \leq k] \geq \alpha \Rightarrow \chi(G) \leq (4 + 2r)km^{1/4}$.

Proof. Note first that if $k \geq m^{1/4}$, then the claimed bound holds by Lemma 24. So we assume henceforth that $k < m^{1/4}$.

Identify subsets of edges E with vectors in $\mathbb{F}_2^m$. Because $\Pr[\chi(G_{\frac{1}{2},e}) \leq k] \geq \alpha$ by Lemma 15 there exist $E_1, E_2 \subseteq E$ such that $G_1 = (V, E_1), G_2 = (V, E_2)$ are k-colorable and $|E_1 \cup E_2| \geq m - (1 + r)\sqrt{m}$.

Let $E_3 = E \setminus (E_1 \cup E_2)$ be the set of edges that are not in $E_1 \cup E_2$, and define the graph $G_3 = (V, E_3)$. Define a partition $U, \overline{U}$ of V, where $U = \{v \in V : \deg_{G_3}(v) < m^{1/4}/k\}$ and $\overline{U} = \{v \in V : \deg_{G_3}(v) \geq m^{1/4}/k\}$. We claim (1) that $\chi(G_{|U}) \leq 2km^{1/4}$ and (2) that $\chi(G_{|\overline{U}}) \leq 2(1 + r)km^{1/4}$. By Lemma 19 we then get the upper bound $\chi(G) \leq \chi(G_{|U}) + \chi(G_{|\overline{U}}) \leq (4 + 2r)km^{1/4}$.

To prove (1) note that by Brooks' Theorem [6] we have $\chi((G_3)_{|U}) \leq 2m^{1/4}/k$, and thus by Lemma 20 $\chi(G_{|U}) \leq \chi(G_1) \cdot \chi(G_2) \cdot \chi((G_3)_{|U}) \leq 2km^{1/4}$. For (2) note that $\sum_{v \in \overline{U}} \deg_{G_3}(v) \leq 2(1 + r)\sqrt{m}$, and hence $\chi(G_{|\overline{U}}) \leq \left|\overline{U}\right| \leq 2(1 + r)km^{1/4}$, as required. ◀

Taking the contrapositive of Lemmas 23 and 25 implies Theorem 6.

Proof of Theorem 7

Finally, we use Theorem 6 to prove the strong robustness result for Gap-Coloring. Let G be an instance of the Gap-Coloring(q, Q) problem. We claim the following:

YES-case: If $\chi(G) \leq q$, then $\chi(G_{\frac{1}{2},e}) < q$.

NO-case: If $\chi(G) \geq Q$, then $\chi(G_{\frac{1}{2},e}) \geq \Omega(Q^{1/3})$ with probability at least 0.99.

The YES-case is clear, since removing edges cannot increase the chromatic number. The NO-case follows from Theorem 6. Thus, the Gap-Coloring(q, Q) problem is strongly-noise-robust to the Gap-Coloring$(q, \Omega(Q^{1/3}))$ problem. The "in particular" part of the theorem follows from the result of Huang [18] showing that Gap-Coloring$(q, 2^{\Omega(q^{1/3})})$ is $\mathcal{NP}$-hard.

Acknowledgements. We thank Itai Benjamini, Uri Feige, Sam Hopkins for useful discussions. We are grateful to the anonymous referees for helpful comments and turning our attention to [7]. Part of this work was done during the visit of the second author at NYU. The second author wishes to thank Subhash Khot for his hospitality.

References

1 N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley-Interscience series in discrete mathematics and optimization. J. Wiley & Sons, New York, 2000.

2 B. Barak, M. Hardt, T. Holenstein, and Steurer D. Subsampling mathematical relaxations and average-case complexity. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, San Francisco, California, USA*, pages 512–531, 2011. doi:10.1137/1.9781611973082.41.

3 B. Bollobás. The chromatic number of random graphs. *Combinatorica*, 8(1):49–55, 1988. doi:10.1007/BF02122551.

4 B. Bollobás. *Random graphs*. Springer, 1998.

5 B. Bollobás, B. P. Narayanan, and A. M. Raigorodskii. On the stability of the erdős-ko-rado theorem. *J. Comb. Theory, Ser. A*, 137:64–78, 2016. doi:10.1016/j.jcta.2015.08.002.

6 R. L. Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37:194–197, 4 1941. doi:10.1017/S030500410002168X.

7 B. Bukh. Interesting problems that I cannot solve. Problem 2. http://www.borisbukh.org/problems.html.

8 I. Dinur, E. Mossel, and O. Regev. Conditional hardness for approximate coloring. *SIAM J. Comput.*, 39(3):843–873, 2009. doi:10.1137/07068062X.

9 I. Dinur and I. Shinkar. On the conditional hardness of coloring a 4-colorable graph with super-constant number of colors. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, Barcelona, Spain, September 1-3, 2010. Proceedings*, pages 138–151, 2010. doi:10.1007/978-3-642-15369-3_11.

10 M. Etscheid and H. Röglin. Smoothed analysis of local search for the maximum-cut problem. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, Portland, Oregon, USA*, pages 882–889, 2014. doi:10.1137/1.9781611973402.66.

11 U. Feige and J. Kilian. Zero knowledge and the chromatic number. *Journal of Computer and System Sciences*, 57(2):187–199, 1998. doi:10.1006/jcss.1998.1587.

12 U. Feige and Daniel Reichman. Recoverable values for independent sets. *Random Struct. Algorithms*, 46(1):142–159, 2015. doi:10.1002/rsa.20492.

13 A. M. Frieze and C. McDiarmid. Algorithmic theory of random graphs. *Random Struct. Algorithms*, 10(1-2):5–42, 1997. doi:10.1002/(SICI)1098-2418(199701/03)10:1/2<5::AID-RSA2>3.0.CO;2-Z.

14 M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *J. ACM*, 23(1):43–49, January 1976. doi:10.1145/321921.321926.

15 G. Grimmett. *Percolation*. Springer, 1999.

16 G. R. Grimmett and C. J. H. McDiarmid. On colouring random graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 77:313–324, 3 1975. doi:10.1017/S0305004100051124.

17 S. Har-Peled. Concentration of Random Variables – Chernoff's Inequality. Available at http://sarielhp.org/teach/13/b_574_rand_alg/lec/07_chernoff.pdf.

18 S. Huang. Improved hardness of approximating chromatic number. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, pages 233–243, 2013. doi:10.1007/978-3-642-40328-6_17.

19 D. R. Karger. Random sampling in cut, flow, and network design problems. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, Montréal, Québec, Canada*, pages 648–657, 1994. doi:10.1145/195058.195422.

20 S. Khot. Improved inapproximability results for maxclique, chromatic number and approximate graph coloring. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 600–609, 2001. doi:10.1109/SFCS.2001.959936.

21 L. Kucera. The greedy coloring is a bad probabilistic algorithm. *J. Algorithms*, 12(4):674–684, 1991.

22 T. Łuczak. The chromatic number of random graphs. *Combinatorica*, 11(1):45–54, 1991. doi:10.1007/BF01375472.

23 M. Mezard and A. Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

24 S. Misra. *Decay of Correlation and Inference in Graphical Models*. PhD thesis, MIT, 2014.

25 E. Mossel, R. O'Donnell, O. Regev, J. E. Steif, and B. Sudakov. Non-interactive correlation distillation, inhomogeneous Markov chains, and the reverse Bonami-Beckner inequality. *Israel Journal of Mathematics*, 154:299–336, 2006.

26 O. Sisask. Discrete fourier analysis. Available at https://people.kth.se/~sisask/Shillong/DFA_2.pdf.

27 A. Sly. Computational transition at the uniqueness threshold. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, Las Vegas, Nevada, USA*, pages 287–296, 2010. doi:10.1109/FOCS.2010.34.

28 D. A. Spielman and S-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3):385–463, 2004. doi:10.1145/990308.990310.

29 D. A. Spielman and S-H. Teng. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Commun. ACM*, 52(10):76–84, 2009. doi:10.1145/1562764.1562785.

30 V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.

Tight Hardness Results for Maximum Weight Rectangles*

Arturs Backurs[†1], Nishanth Dikkala[‡2], and Christos Tzamos[§3]

1 **MIT, Cambridge, USA**
backurs@mit.edu
2 **MIT, Cambridge, USA**
ndikkala@mit.edu
3 **MIT, Cambridge, USA**
tzamos@mit.edu

Abstract

Given n weighted points (positive or negative) in d dimensions, what is the axis-aligned box which maximizes the total weight of the points it contains?

The best known algorithm for this problem is based on a reduction to a related problem, the Weighted Depth problem [Chan, FOCS, 2013], and runs in time $O(n^d)$. It was conjectured [Barbay et al., CCCG, 2013] that this runtime is tight up to subpolynomial factors. We answer this conjecture affirmatively by providing a matching conditional lower bound. We also provide conditional lower bounds for the special case when points are arranged in a grid (a well studied problem known as Maximum Subarray problem) as well as for other related problems.

All our lower bounds are based on assumptions that the best known algorithms for the All-Pairs Shortest Paths problem (APSP) and for the Max-Weight k-Clique problem in edge-weighted graphs are essentially optimal.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Maximum Rectangles, Hardness in P

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.81

1 Introduction

Consider a set of points in the plane. Each point is assigned a real weight that can be either positive or negative. The Max-Weight Rectangle problem asks to find an axis parallel rectangle that maximizes the total weight of the points it contains. This problem (and its close variants) is one of the most basic problems in computational geometry and is used as a subroutine in many applications [17, 20, 23, 7, 6]. Despite significant work over the past two decades, the best known algorithm runs in time quadratic in the number of points [16, 14, 9]. It has been conjectured that there is no strongly subquadratic time algorithm[1] for this problem [9].

An important special case of the Max-Weight Rectangle problem is when the points are arranged in a square grid. In this case the input is given as an $n \times n$ matrix filled with

* The full version of the paper is available at [8].
† A. Backurs is supported by grants from the NSF and the Simons Investigator award.
‡ N. Dikkala is supported by grants from the NSF.
§ C. Tzamos is supported by NSF Award CCF-0953960 (CAREER) and a Simons Award for Graduate Students in Theoretical Computer Science.
1 A strongly subquadratic algorithm runs in time $O(N^{2-\varepsilon})$ for constant $\varepsilon > 0$.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 81; pp. 81:1–81:13


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

Table 1 Upper bounds and conditional lower bounds for the various problems studied. The bounds shown ignore subpolynomial factors.

Problem	**In 2 dimensions**	**In d dimensions**
Max-Weight Rectangle on N weighted points	$O(N^2)$ [9, 12] $\Omega(N^2)$ **[this work]**	$O(N^d)$ [9, 12] $\Omega(N^d)$ **[this work]**
Maximum Subarray on $n \times \cdots \times n$ arrays	$O(n^3)$ [32, 31] $\Omega(n^3)$ **[this work]**	$O(n^{2d-1})$ [Kadane's algorithm] $\Omega(n^{3d/2})$ **[this work]**
Maximum Square Subarray on $n \times \cdots \times n$ arrays	$O(n^3)$ [trivial] $\Omega(n^3)$ **[this work]**	$O(n^{d+1})$ [trivial] $\Omega(n^{d+1})$ **[this work]**
Weighted Depth on N weighted boxes	$O(N)$ [12] $\Omega(N)$ [trivial]	$O(N^{d/2})$ [12] $\Omega(N^{d/2})$ **[this work]**

real numbers and the objective is to compute a subarray that maximizes the sum of its entries [27, 31, 30, 28, 13]. This problem, known as Maximum Subarray problem, has applications in pattern matching [19], data mining and visualization [20] (see [31] for additional references). The particular structure of the Maximum Subarray problem allows for algorithms that run in $O(n^3)$, i.e. $O(N^{3/2})$ with respect to the input size $N = n^2$, as opposed to $O(N^2)$ which is the best algorithm for the more general Max-Weight Rectangle problem.

One interesting question is if this discrepancy between the runtimes of these two very related problems can be avoided. Is it possible to apply ideas from one to improve the runtimes of the other? Despite considerable effort there has been no significant improvement to their runtime other than by subpolynomial factors since they were originally studied.

In this work, we attempt to explain this apparent barrier for faster runtimes by giving evidence of the inherent hardness of the problems. In particular, we show that a strongly subquadratic algorithm for Max-Weight Rectangle would imply a breakthrough for fundamental graph problems. We show similar consequences for $O(N^{3/2-\varepsilon})$ algorithms for the Maximum Subarray problem. Our lower bounds are based on standard hardness assumptions for the All-Pairs Shortest Paths and the Max-Weight k-Clique problems and generalize to the higher-dimensional versions of the problems.

1.1 Related work on the problems

In one dimension, the Max-Weight Rectangle problem and Maximum Subarray problem are identical. The 1-D problem was first posed by Ulf Grenander for pattern detection in images, and a linear time algorithm was found by Jay Kadane [10].

In two dimensions, Dobkin et al [16, 15, 24] studied the Max-Weight Rectangle problem in the case where weights are $+1$ or -1 for its applications to computer graphics and machine learning. They presented the first $O(N^2 \log N)$ algorithm. More recently, Cortés et al [14] studied the problem with arbitrary weights and they developed an algorithm with the same runtime applicable to many variants of the problem. An even faster algorithm was shown by Barbay et al. [9] that runs in $O(N^2)$ time.

For higher dimensions, Barbay et al [9] show a reduction to the related Weighted Depth problem which allows them to achieve runtime $O(N^d)$. Given N axis-parallel rectangular weighted boxes, the Weighted Depth problem asks to find a point that maximizes the total weight of all boxes that contain it. Compared to the Max-Weight Rectangle where we are given points and we aim to find the best box, in this problem, we are given boxes and the aim is to find the best point. The Weighted Depth problem is also related to Klee's

measure problem[2] which has a long line of research. All known algorithms for one problem can be adjusted to work for the other [12]. The Weighted Depth problem was first solved in $O(N^{d/2} \log n)$ by Overmars and Yap [26] and was improved to $O(N^{d/2})$ by Timothy M. Chan [12] who gave a surprisingly simple divide and conquer algorithm.

A different line of work, studies the Maximum Subarray problem. Kadane's algorithm for the 1-dimensional problem can be generalized in higher dimensions for d-dimensional $n \times \cdots \times n$ arrays giving $O(n^{2d-1})$ which implies an $O(n^3)$ algorithm when the array is a $n \times n$ matrix. Tamaki and Tokuyama [32] gave a reduction of the 2-dimensional version of the problem to the distance product problem implying a $O\left(\frac{n^3}{2^{\Omega(\sqrt{\log n})}}\right)$ algorithm by using the latest algorithm for distance product by Ryan Williams [34]. Tamaki and Tokuyama's reduction was further simplified by Tadao Takaoka [31] who also gave a more practical algorithm whose expected time is close to quadratic for a wide range of random data.

1.2 Our results and techniques

Despite significant work on the Max-Weight Rectangle and Maximum Subarray problems, it seems that there is a barrier in improving the best known algorithms for these problems by polynomial factors. Our results indicate that this barrier is inherent by showing connections to well-studied fundamental graph problems. In particular, our first result states that there is no strongly subquadratic algorithm for the Max-Weight Rectangle problem unless the Max-Weight k-Clique problem can be solved in $O(n^{k-\varepsilon})$ time, i.e. substantially faster than the currently best known algorithm. More precisely, we show the following:

▶ **Theorem 1.** *For any constant $\varepsilon > 0$, an $O(N^{2-\varepsilon})$ algorithm for the* Max-Weight Rectangle *problem on N weighted points in the plane implies an $O(n^{k-\varepsilon})$ algorithm for the* Max-Weight k-Clique *problem on a weighted graph with n vertices where $k = 4 \cdot \lceil \varepsilon^{-1} \rceil$.*

Our conditional lower bound generalizes to higher dimensions. Namely, we show that an $O(N^{d-\varepsilon})$ time algorithm for points in d-dimensions implies an $O(n^{k-\varepsilon})$ time algorithm for the Max-Weight k-Clique problem for $k = d^2\lceil \varepsilon^{-1} \rceil$. This matches the best known algorithm [9, 12] for any dimension up to subpolynomial factors. Therefore, because of our reduction, significant improvements in the runtime of the known upper bounds would imply a breakthrough algorithm for finding a k-clique of maximum weight in a graph.

To show this result, we embed an instance of the Max-Weight k-Clique problem to the Max-Weight Rectangle problem, by treating coordinates of the optimal rectangular box as base-n numbers where digits correspond to nodes in the maximum-weight k-clique. In the construction, we place points with appropriate weights so that the weight of any rectangular box corresponds to the weight of the clique it represents. We show that it is sufficient to use only $O(n^{\frac{k}{d}+1})$ points in d-dimensions to represent all weighted k-cliques which gives us the required bound by choosing an appropriately large k.

We also study the special case of the Max-Weight Rectangle problem in the plane where all points are arranged in a square grid, namely the Maximum Subarray problem. Our second result states that for $n \times n$ matrices, there is no strongly subcubic algorithm for the Maximum Subarray problem unless there exists a strongly subcubic algorithm for the All-Pairs Shortest Paths problem. More precisely, we show that:

[2] Klee's measure problem asks for the total volume of the union of N axis-parallel boxes in d dimensions.

▶ **Theorem 2.** *For any constant $\varepsilon > 0$, an $O(n^{3-\varepsilon})$ time algorithm for the* Maximum Subarray *problem on $n \times n$ matrices implies an $O(n^{3 - \varepsilon/10})$ time algorithm for the* All-Pairs Shortest Paths *problem.*

Combined with the fact that the Maximum Subarray problem reduces to the All-Pairs Shortest Paths problem as shown in [32, 31], our result implies that the two problems are equivalent, in the sense that any strongly subcubic algorithm for one would imply a strongly subcubic algorithm for the other.

To extend our lower bound to higher dimensions, we need to make a stronger hardness assumption based on the Max-Weight k-Clique problem. We show that an $O(n^{3d/2 - \varepsilon})$ time algorithm for the Maximum Subarray problem in d-dimensions implies an $O(n^{k-\varepsilon})$ time algorithm for the Max-Weight k-Clique problem. To prove this result, we introduce the following intermediate problem: Given a graph G find a maximum weight subgraph H that is isomorphic to a clique on $2d$ nodes without the edges of a matching (Max-Weight Clique without Matching problem). This graph H contains a large clique of size $3d/2$ as a minor and we show that this implies that no $O(n^{3d/2 - \varepsilon})$ algorithms exist for the Max-Weight Clique without Matching problem. We complete our proof by reducing the Max-Weight Clique without Matching problem to the Maximum Subarray problem in d dimensions.

We note that the best known algorithm for the Maximum Subarray problem runs in $O(n^{2d-1})$ time and is based on Kadane's algorithm for the 1-dimensional problem. It remains an interesting open question to close this gap. To improve either the lower or upper bound, it is necessary to better understand the computational complexity of the Max-Weight Clique without Matching problem.

Another related problem we consider is the Maximum Square Subarray problem: Given an $n \times n$ matrix find a maximum subarray with sides of equal length. This problem and its higher dimensional generalization can be trivially solved in $O(n^{d+1})$ runtime by enumerating over all possible combinations of the $d + 1$ parameters, i.e. the side-length and the location of the hypercube. We give a matching lower bound based on hardness of the Max-Weight k-Clique problem.

Finally, we adapt the reduction for Klee's measure problem shown by Timothy M Chan [11] to show a lower bound for the Weighted Depth problem.

Our results are summarized in Table 1, where we compare the current best upper bounds with the conditional lower bounds that we show.

The conditional hardness results presented above are for the variants of the problems where weights are arbitrary real numbers. We note that all these bounds can be adapted to work for the case where weights are either $+1$ or -1. In this case, we reduce the (unweighted) k-Clique-Detection problem[3] to each of these problems. The k-Clique-Detection problem can be solved in $O(n^{\omega\lfloor k/3 \rfloor + (k \bmod 3)})$ [25] using fast matrix multiplication, where $\omega < 2.372864$ [35, 22] is the fast matrix multiplication exponent.[4] Without using fast matrix multiplication, it is not known whether a purely combinatorial algorithm exists that runs in $O(n^{k-\varepsilon})$ time for any constant $\varepsilon > 0$ and it is a longstanding graph problem. Our lower bounds can be adapted for the $+1$ / -1 versions of the problems obtaining the same runtime exponents for combinatorial algorithms as in Table 1. Achieving better exponents

[3] Given a graph on n vertices, the k-Clique-Detection problem asks whether a k-clique exists in the graph.

[4] There is a slightly faster algorithm for the case when k is not divisible by 3 [18].

for any of these problems would imply a breakthrough combinatorial algorithm for the k-Clique-Detection problem.

There is a vast collection of problems in computation geometry for which conditional lower bounds are based on the assumption of 3-SUM hardness, i.e. that the best known algorithm for the 3-SUM problem[5] can't be solved in time $O(n^{2-\varepsilon})$. This line of research was initiated by [21] (see [33] for more references). Reducing 3-SUM problem to the problems that we study seems hard if possible at all. Our work contributes to the list of interesting geometry problems for which hardness is shown from different assumptions.

1.3 Hardness assumptions

There is a long list of works showing conditional hardness for various problems based on the All-Pairs Shortest Paths problem hardness assumption [29, 36, 4, 2, 3]. Among other results, [36] showed that deciding whether a weighted graph contains a triangle of negative weight is equivalent to the All-Pairs Shortest Paths problem meaning that a strongly subcubic algorithm for the Negative Triangle problem implies a strongly subcubic algorithm for the All-Pairs Shortest Paths problem and the other way around. It is easy to show that the problem of computing the maximum weight triangle in a graph is equivalent to the Negative Triangle problem (by inverting edge-weights of the graph and doing the binary search over the weight of the max-weight triangle). Computing a max-weight triangle is a special case of the problem of computing a max-weight k-clique in a graph for a fixed integer k. This is a very well studied computational problem and despite serious efforts, the best known algorithm for this problem still runs in time $O(n^{k-o(1)})$, which matches the runtime of the trivial algorithm up to subpolynomial factors. The assumption that there is no $O(n^{k-\varepsilon})$ time algorithm for this problem, has served as a basis for showing conditional hardness results for several problems on sequences [1, 5].

2 Preliminaries

2.1 Problems studied in this work

▶ **Definition 3** (Max-Weight Rectangle problem)**.** Given N weighted points (positive or negative) in $d \geq 2$ dimensions, what is the axis-aligned box which maximizes the total weight of the points it contains?

▶ **Definition 4** (Maximum Subarray problem)**.** Given a d-dimensional array M with n^d real-valued entries, find the d-dimensional subarray of M which maximizes the sum of the elements it contains.

▶ **Definition 5** (Max-Weight Square problem)**.** Given a d-dimensional array M with n^d real-valued entries, find the d-dimensional square (hypercube) subarray of M, i.e. a rectangular box with all sides of equal length, which maximizes the sum of the elements it contains.

▶ **Definition 6** (Weighted Depth problem)**.** Given a set of N weighted axis-parallel boxes in d-dimensional space $\mathbb{R}^d$, find a point $p \in \mathbb{R}^d$ that maximizes the sum of the weights of the boxes containing p.

[5] Given a set of integers, decide if there are 3 integers that sum up to 0.

2.2 Hardness assumptions

We use the hardness assumptions of the following problems. Whenever we refer to a weighted graph, we assume that the graph is *edge-weighted* (as opposed to node-weighted).

▶ **Definition 7** (All-Pairs Shortest Paths problem)**.** Given a weighted undirected graph $G = (V, E)$ such that $|V| = n$, find the shortest path between u and v for every $u, v \in V$.

▶ **Definition 8** (Negative Triangle problem)**.** Given a weighted undirected graph $G = (V, E)$ such that $|V| = n$, output yes if there exists a triangle in the graph with negative total edge weight.

▶ **Definition 9** (Max-Weight k-Clique problem)**.** Given an integer k and a weighted graph $G = (V, E)$ with n vertices, output the maximum total edge-weight of a k-clique in the graph. W.l.o.g. we assume that the graph is complete since otherwise we can set the weight of non-existent edges to be equal to a negative integer with large absolute value.

For any fixed k, the best known algorithm for the Max-Weight k-Clique problem runs in time $O(n^{k-o(1)})$.

In Sections 3 and 5, we use the following variant of the Max-Weight k-Clique problem which can be shown to be equivalent to Definition 9:

▶ **Definition 10** (Max-Weight k-Clique problem for k-partite graphs)**.** Given an integer k and a weighted k-partite graph $G = (V_1 \cup \ldots \cup V_k,\ E)$ with kn vertices such that $|V_i| = n$ for all $i \in [k]$. Choose k vertices $v_i \in V_i$ and consider total edge-weight of the k-clique induced by these vertices. Output the maximum total-edge weight of a clique in the graph.

Notation

For any integer n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. For a set S and an integer d, we denote the set $\{(s_1, \ldots, s_d) \mid s_i \in S\}$ by S^d.

3 Hardness of the Max-Weight Rectangle problem

The goal of this section is to show a hardness result for the Max-Weight Rectangle problem making the assumption of Max-Weight k-Clique hardness. We will show the result directly for any constant number of dimensions.

▶ **Theorem 11.** *For any constants $\varepsilon > 0$ and d, an $O(N^{d-\varepsilon})$ time algorithm for the* Max-Weight Rectangle *problem on N weighted points in d-dimensions implies an $O(n^{K-\varepsilon})$ time algorithm for the* Max-Weight K-Clique *problem on a weighted graph with n vertices for $K = d^2\lceil\varepsilon^{-1}\rceil$.*

We set $k = d \cdot \lceil\varepsilon^{-1}\rceil$. To prove the theorem, we will construct an instance of the Max-Weight Rectangle problem whose answer computes a max-weight dk-clique in a $(d \times k)$-partite weighted graph G with n nodes in each of its parts. The Max-Weight dk-Clique problem on general graphs reduces to this case since we can create $d \times k$ copies of the nodes and connect nodes among different parts with edge-weights as in the original graph.

The instance of the Max-Weight Rectangle problem will consist of $N = O(n^{k+1})$ points with integer coordinates $\{-n^k, ..., n^k\}^d$. For such an instance the required runtime

for the Max-Weight Rectangle problem, from the theorem statement, would imply that the maximum weight dk-clique can be computed in

$$O\left(N^{d-\varepsilon}\right) = O\left(n^{(k+1)(d-\varepsilon)}\right) = O\left(n^{dk-\varepsilon+(d-k\varepsilon)}\right) = O\left(n^{dk-\varepsilon}\right)$$

where the last equality follows as $k \geq \frac{d}{\varepsilon}$.

To perform the reduction we introduce the following intermediate problem:

▶ **Definition 12** (Restricted Rectangle problem)**.** Given $N = \Omega(n^k)$ weighted points in an $\{-n^k, ..., n^k\}^d$-grid, compute a rectangular box of a restricted form that maximizes the weight of its enclosed points. The rectangular box $\prod_{i=1}^{d}[-x'_i, x_i]$ must satisfy the following conditions:

1. Both $\vec{x}, \vec{x}' \in \{0, ..., n^k - 1\}^d$, and
2. Treating each coordinate x_i as a k-digit integer $(x_{i1}x_{i2}...x_{ik})_n$ in base n, i.e., $x_i = \sum_{j=1}^{k} x_{ij}n^{k-j}$ and $x_{ij} \in \{0, ..., n-1\}$, we must have $\vec{x}' = (\overline{x_d}, \overline{x_1}, \overline{x_2}, ..., \overline{x_{d-1}})$, where for an integer $z = (z_1z_2...z_k)_n \in \{0, ..., n^k - 1\}$, we denote by $\overline{z} = (z_k...z_2z_1)_n$ the integer that has all the digits reversed.

We show that the Restricted Rectangle problem reduces to the Max-Weight Rectangle problem.

3.1 Restricted Rectangle ⇒ Max-Weight Rectangle

Consider an instance of the Restricted Rectangle problem. We can convert it to an instance of the Max-Weight Rectangle problem by introducing several additional points. Let C be a number greater than twice the sum of absolute values of all weights of the given points. We know that the solution to any rectangular box must have weight in $(-C/2, C/2)$.

The conditions of the Restricted Rectangle require that the rectangular box must contain the origin $\vec{0}$. To satisfy that we introduce a point with weight C at the origin. This forces the optimal rectangle to contain the origin since any rectangle that doesn't include this point gets weight strictly less than C.

The integrality constraint is satisfied since all points in the instance have integer coordinates so without loss of generality the optimal rectangle in the Max-Weight Rectangle problem will also have integer coordinates.

Finally, we can force $x'_2 = \overline{x_1}$, by adding for each $x_1 \in \{0, ..., n^k - 1\}$ the 4 points:

- $(x_1, -\overline{x_1}, 0, 0, .., 0)$ with weight C
- $(x_1 + 1, -\overline{x_1}, 0, 0, .., 0)$ with weight $-C$
- $(x_1, -\overline{x_1} - 1, 0, 0, .., 0)$ with weight $-C$
- $(x_1 + 1, -\overline{x_1} - 1, 0, 0, .., 0)$ with weight C

This creates $4n^k$ points and adds weight C to any rectangle with $x'_2 = \overline{x_1}$ without affecting any of the others. Working similarly for $x_2..., x_d$ we can force that the optimal solution satisfies the constraint that $\vec{x}' = (\overline{x_d}, \overline{x_1}, \overline{x_2}, ..., \overline{x_{d-1}})$.

If x and x' satisfy the conditions of the Restricted Rectangle problem, we collect weight dC for satisfying the constraints on all coordinates and C from including the point at the origin. So the total weight is at least $(d+1)C - \frac{C}{2} = (d + \frac{1}{2})C$ as every rectangle has weight at least $-C/2$ with respect to the original points. On the other hand, if at least one of the conditions is not satisfied, we receive weight strictly less than $(d + \frac{1}{2})C$. Thus, the optimal rectangular box for the Max-Weight Rectangle problem coincides with the optimal rectangular box for the Restricted Rectangle problem. The total number of points is still $O(N)$ since $N = \Omega(n^k)$ and we added $O(n^k)$ points.

3.2 Max-Weight $(d \times k)$-Partite Clique $\Rightarrow$ Restricted Rectangle

Consider a $(d \times k)$-partite weighted graph G. We label each of its parts as P_{ij} for $i \in [d]$ and $j \in [k]$. We associate each dk-clique of the graph G with a corresponding rectangular box in the Restricted Rectangle problem. In particular, for a rectangular box defined by a point $\vec{x} \in \{0, ..., n^k - 1\}^d$, each x_{ij}, i.e. the j-th most significant digit of x_i in the base n representation, corresponds to the index of the node in part P_{ij} (0-indexed).

We now create an instance by adding points so that the total weight of every rectangular box satisfying the conditions of the Restricted Rectangle problem is equal to the weight of its corresponding dk clique. To do that we need to take into account the weights of all the edges. We can easily take care of edges between parts $P_{11}, P_{12}, ..., P_{1k}$ of the graph by adding the following points for each $x_1 \in \{0, ..., n^k - 1\}$.

- $(x_1, 0, 0, 0, .., 0)$ with weight $W(x_1)$ equal to the weight of the k-clique $x_{11}, x_{12}, ..., x_{1k}$ in parts $P_{11}, P_{12}, ..., P_{1k}$
- $(x_1 + 1, 0, 0, 0, .., 0)$ with weight $-W(x_1)$

This creates $2n^k$ points and adds weight $W(x_1)$ to any rectangle whose first coordinate matches x_1 without affecting any of the others. We work similarly for every coordinate i from 2 through d accounting for the weight of all edges between parts P_{ia} and P_{ib} for all $i \in [d]$ and $a \neq b \in [k]$. To take into account the additional edges, we show how to add edges between parts P_{1a} and P_{2b}. For all $x_1 \in n^{k-a}\{0, ..., n^a - 1\}$ and $x_2 \in n^{k-b}\{0, ..., n^b - 1\}$ we add the points:

- $(x_1, x_2, 0, 0, .., 0)$ with weight w equal to the weight of the edge between nodes x_{1a} and x_{2b} in parts P_{1a} and P_{2b}.
- $(x_1 + n^{k-a}, x_2, 0, 0, .., 0)$ with weight $-w$
- $(x_1, x_2 + n^{k-b}, 0, 0, .., 0)$ with weight $-w$
- $(x_1 + n^{k-a}, x_2 + n^{k-b}, 0, 0, .., 0)$ with weight w

This adds weight equal to the weight of the edge between nodes x_{1a} and x_{2b} in parts P_{1a} and P_{2b} for any rectangle with corner $\vec{x}$. This creates $O(n^{a+b})$ points. This number becomes too large if $a + b > k + 1$. However, if this is the case we can instead apply the same construction in the part of the space where the numbers x_1 and x_2 appear reversed, i.e. by working with $x'_2 = \overline{x_1}$ and $x'_3 = \overline{x_2}$. For all $x'_2 \in n^{a-1}\{0, ..., n^{k+1-a} - 1\}$ and $x'_3 \in n^{b-1}\{0, ..., n^{k+1-b} - 1\}$ we add the points:

- $(0, -x'_2, -x'_3, 0, 0, .., 0)$ with weight w equal to the weight of the edge between nodes $x'_{2(k+1-a)}$ and $x'_{3(k+1-b)}$ in parts P_{1a} and P_{2b}.
- $(0, -x'_2 - n^{a-1}, -x'_3, 0, .., 0)$ with weight $-w$
- $(0, -x'_2, -x'_3 - n^{b-1}, 0, .., 0)$ with weight $-w$
- $(0, -x'_2 - n^{a-1}, -x'_3 - n^{b-1}, 0, .., 0)$ with weight w

This produces the identical effect as above creating $O(n^{2k+2-a-b})$ rectangles. If $a + b \geq k + 1$ this adds at most $O(n^{k+1})$ points as desired. We add edges between any other 2 parts $P_{i,\cdot}$ and $P_{i',\cdot}$ by performing a similar construction as above.

The overall number of points in the instance is $O(n^{k+1})$ and this completes the proof of the theorem.

4 Hardness for Maximum Subarray in 2 dimensions

In this section our goal is to show that, if we can solve the Maximum Subarray problem on a matrix of size $n \times n$ in time $O(n^{3-\varepsilon})$, then we can solve the Negative Triangle problem in time $O(n^{3-\varepsilon})$ on n vertex graphs. It is known that a $O(n^{3-\varepsilon})$ time algorithm for the Negative Triangle implies a $O(n^{3 - \varepsilon/10})$ time algorithm for the All-Pairs Shortest

Paths problem [36]. Combining our reduction with the latter one, we obtain Theorem 2 from the introduction, which we restate here:

▶ **Theorem 2.** *For any constant $\varepsilon > 0$, an $O(n^{3-\varepsilon})$ time algorithm for the* Maximum Subarray *problem on $n \times n$ matrices implies an $O(n^{3-\varepsilon/10})$ time algorithm for the* All-Pairs Shortest Paths *problem.*

The generalization of this statement can be found in Section 5. Here we prove 2-dimensional case first because the argument is shorter.

Clearly, the Negative Triangle problem in equivalent to the Positive Triangle problem. In the remainder of this section we therefore reduce the problem of detecting whether a graph has a positive triangle to the Maximum Subarray problem.

We need the following intermediate problem:

▶ **Definition 13** (Maximum 4-Combination)**.** Given a matrix $B \in \mathbb{R}^{m \times m}$, output

$$\max_{i,i',j,j' \in [m] \,:\, i \le i' \text{ and } j \le j'} B[i,j] + B[i',j'] - B[i,j'] - B[i',j].$$

Our reduction consists of two steps:

1. Reduce the Positive Triangle problem on n vertex graph to the Maximum 4-Combination problem on $2n \times 2n$ matrix.
2. Reduce the Maximum 4-Combination problem on $n \times n$ matrix to the Maximum Subarray matrix of size $n \times n$.

4.1 Positive Triangle ⇒ Maximum 4-Combination

Let A be the weighted adjacency matrix of size $n \times n$ of the graph and let M be the largest absolute value of an entry in A. Let $M' := 10M$ and $M'' := 100M$. We define matrix $D \in \mathbb{R}^{n \times n}$:

$$D_{i,j} = \begin{cases} M' + M'' & \text{if } i = j; \\ M'' & \text{otherwise.} \end{cases}$$

We define matrix $B \in \mathbb{R}^{2n \times 2n}$:

$$B := \begin{bmatrix} A & -A^T \\ -A^T & D \end{bmatrix}.$$

The reduction follows from the following lemma.

▶ **Lemma 14.** *Let X be the weight of the max-weight triangle in the graph corresponding to the adjacency matrix A. Let Y be the output of the* Maximum 4-Combination *algorithm when run on matrix B. The following equality holds:*

$$Y = X + M' + M''.$$

Proof. Consider integers i, j, i', j' that achieve a maximum in the Maximum 4-Combination instance as per Definition 13. Our first claim is that $i, j \le n$ and $i', j' \ge n + 1$. If this is not true, we do not collect the weight M'' and the largest output that we can get is $\le 4M' \le 9M''/10$. Note that we can easily achieve a larger output with $i = j = 1$ and $i' = j' = n + 1$.

Our second claim is that $i' = j'$. If this is not so, we do not collect the weight M' and the largest output that we can get is $M'' + 4M \leq M'' + M'/2$. Note that we can easily achieve a larger output with $i = j = 1$ and $i' = j' = n + 1$. Thus, we can denote $i' = j' = k + n$.

Now, by the construction of B, we have

$$B[i,j] + B[i',j'] - B[i,j'] - B[i',j] = A[i,j] + A[j,k] + A[k,i] + M' + M''.$$

We get the equality we need. ◀

4.2 Maximum 4-Combination ⇒ Maximum Subarray

Let $A' \in \mathbb{R}^{(n+1)\times(n+1)}$ be a matrix defined by $A'[i,j] = A[i-1,j-1]$ if $i,j \geq 2$ and $A'[i,j] = 0$ otherwise.

Let $C \in \mathbb{R}^{n\times n}$ be a matrix defined by $C[i,j] = A'[i,j] + A'[i+1,j+1] - A'[i,j+1] - A'[i+1,j]$.

The reduction follows from the claim that the output of the Maximum Subarray on C is equal to the output of the Maximum 4-Combination on A'. The claim follows from the following equality:

$$\sum_{i=i'}^{i''} \sum_{j=j'}^{j''} C[i,j] \;=\; A'[i''+1,j''+1] + A'[i',j'] - A'[i''+1,j'] - A'[i',j''+1].$$

The proofs of the hardness results of the next 3 sections are presented in the Appendix of this paper in the interest of space.

5 Hardness for Maximum Subarray for arbitrary number of dimensions

We state the hardness result we prove for the Maximum Subarray problem on d dimensional arrays.

▶ **Theorem 15.** *For any constant $\varepsilon > 0$, an $O\left(n^{d+\lfloor d/2 \rfloor - \varepsilon}\right)$ time algorithm for the* Maximum Subarray *problem on d-dimensional array, implies an $O\left(n^{d+\lfloor d/2 \rfloor - \varepsilon}\right)$ time algorithm for the* Max-Weight $(d + \lfloor d/2 \rfloor)$-Clique *problem.*

The complete proof of Theorem 15 is given in the full version of the paper [8]. It generalizes the constructions used in the hardness proof of Theorem 2.

6 Hardness for Maximum Square Subarray problem

We state the hardness result we prove for the Maximum Square Subarray problem on d dimensional arrays.

▶ **Theorem 16.** *For any constant $\varepsilon > 0$, an $O\left(n^{d+1-\varepsilon}\right)$ time algorithm for the* Maximum Square Subarray *problem on a d-dimensional array implies an $O\left(n^{d+1-\varepsilon}\right)$ time algorithm for the* Max-Weight $(d+1)$-Clique *problem.*

The proof of Theorem 16 is given in the full version of the paper [8].

7 Hardness for Weighted Depth problem

We state the hardness result we prove for the Weighted Depth problem in d dimensional space.

▶ **Theorem 17.** *For any constant $\varepsilon > 0$, an $O\left(n^{\lfloor d/2 \rfloor - \varepsilon}\right)$ time algorithm for the* Weighted Depth *problem in d dimensional space implies an $O\left(n^{d-2\varepsilon}\right)$ time algorithm for the* Max-Weight (d)-Clique *problem.*

The proof of the above theorem is given in the full version of the paper [8].

Acknowledgements. We thank Piotr Indyk for providing helpful discussions and comments on an earlier version of the paper. We thank Linda Lynch for copy-editing parts of the paper. One of the authors, Nishanth Dikkala, would like to thank Namratha Dikkala for a discussion about efficient algorithms for geometric problems which led to some of questions pursued in this paper.

References

1 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the current clique algorithms are optimal, so is valiant's parser. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 98–117. IEEE, 2015.

2 Amir Abboud, Fabrizio Grandoni, and Virginia Vassilevska Williams. Subcubic equivalences between graph centrality problems, apsp and diameter. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1681–1697. SIAM, 2015.

3 Amir Abboud, Virginia Vassilevska Williams, and Huacheng Yu. Matching triangles and basing hardness on an extremely popular conjecture. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 41–50. ACM, 2015.

4 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 434–443. IEEE, 2014.

5 Amir Abboud, Virginia Vassilevska Williams, and Oren Weimann. Consequences of faster alignment of sequences. In *Automata, Languages, and Programming*, pages 39–51. Springer, 2014.

6 Deepak Agarwal, Jeff M Phillips, and Suresh Venkatasubramanian. The hunting of the bump: on maximizing statistical discrepancy. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 1137–1146. Society for Industrial and Applied Mathematics, 2006.

7 Jonathan Backer and J Mark Keil. The mono-and bichromatic empty rectangle and square problems in all dimensions. In *LATIN 2010: Theoretical Informatics*, pages 14–25. Springer, 2010.

8 Arturs Backurs, Nishanth Dikkala, and Christos Tzamos. Tight hardness results for maximum weight rectangles. *arXiv preprint arXiv:1602.05837*, 2016. URL: `http://arxiv.org/abs/1602.05837`.

9 Jérémy Barbay, Timothy M Chan, Gonzalo Navarro, and Pablo Pérez-Lantero. Maximum-weight planar boxes in o (n^2) time (and better). *Information Processing Letters*, 114(8):437–445, 2014.

10 Jon Bentley. Programming pearls: algorithm design techniques. *Communications of the ACM*, 27(9):865–873, 1984.

11 Timothy M Chan. A (slightly) faster algorithm for klee's measure problem. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 94–100. ACM, 2008.

12 Timothy M Chan. Klee's measure problem made easy. In *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pages 410–419. IEEE, 2013.

13 Chih-Huai Cheng, Kuan-Yu Chen, Wen-Chin Tien, and Kun-Mao Chao. Improved algorithms for the k maximum-sums problems. In *Algorithms and Computation*, pages 799–808. Springer, 2005.

14 C Cortés, José Miguel Díaz-Báñez, Pablo Pérez-Lantero, Carlos Seara, Jorge Urrutia, and Inmaculada Ventura. Bichromatic separability with two boxes: a general approach. *Journal of Algorithms*, 64(2):79–88, 2009.

15 David P Dobkin and Dimitrios Gunopulos. Computing the rectangle discrepancy. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 385–386. ACM, 1994.

16 David P Dobkin, Dimitrios Gunopulos, and Wolfgang Maass. Computing the maximum bichromatic discrepancy, with applications to computer graphics and machine learning. *journal of computer and system sciences*, 52(3):453–470, 1996.

17 Jonathan Eckstein, Peter L Hammer, Ying Liu, Mikhail Nediak, and Bruno Simeone. The maximum box problem and its application to data analysis. *Computational Optimization and Applications*, 23(3):285–298, 2002.

18 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1):57–67, 2004.

19 Paul Fischer, Klaus-U Höffgen, Hanno Lefmann, and Tomasz Luczak. Approximations with axis-aligned rectangles. In *Fundamentals of Computation Theory*, pages 244–255. Springer, 1993.

20 Takeshi Fukuda, Yasukiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. *ACM SIGMOD Record*, 25(2):13–23, 1996.

21 Anka Gajentaan and Mark H Overmars. On a class of o (n 2) problems in computational geometry. *Computational geometry*, 5(3):165–185, 1995.

22 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pages 296–303. ACM, 2014.

23 Ying Liu and Mikhail Nediak. Planar case of the maximum box and related problems. In *CCCG*, pages 14–18, 2003.

24 Wolfgang Maass. Efficient agnostic pac-learning with simple hypothesis. In *Proceedings of the seventh annual conference on Computational learning theory*, pages 67–75. ACM, 1994.

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

26 Mark H Overmars and Chee-Keng Yap. New upper bounds in klee's measure problem. *SIAM Journal on Computing*, 20(6):1034–1045, 1991.

27 Kalyan Perumalla and Narsingh Deo. Parallel algorithms for maximum subsequence and maximum subarray. *Parallel Processing Letters*, 5(03):367–373, 1995.

28 Ke Qiu and Selim G Akl. Parallel maximum sum algorithms on interconnection networks. In *Proceedings of the Eleventh IAESTED Conference on Parallel and Distributed Computing and Systems*, pages 31–38. Citeseer, 1999.

29 Liam Roditty and Uri Zwick. On dynamic shortest paths problems. In *Algorithms–ESA 2004*, pages 580–591. Springer, 2004.

30 Douglas R Smith. Applications of a strategy for designing divide-and-conquer algorithms. *Science of Computer Programming*, 8(3):213–229, 1987.

31 Tadao Takaoka. Efficient algorithms for the maximum subarray problem by distance matrix multiplication. *Electronic Notes in Theoretical Computer Science*, 61:191–200, 2002.

32 Hisao Tamaki and Takeshi Tokuyama. Algorithms for the maxium subarray problem based on matrix multiplication. In *SODA*, volume 1998, pages 446–452, 1998.

33 Virginia Vassilevska Williams. Hardness of easy problems: Basing hardness on popular conjectures such as the strong exponential time hypothesis (invited talk). In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 43. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

34 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 664–673. ACM, 2014.

35 Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.

36 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 645–654. IEEE, 2010.

The Johnson-Lindenstrauss Lemma Is Optimal for Linear Dimensionality Reduction

Kasper Green Larsen[*1] **and Jelani Nelson**[†2]

1 **Aarhus University, Aarhus, Denmark**
larsen@cs.au.dk
2 **Harvard University, Cambridge, MA, USA**
minilek@seas.harvard.edu

Abstract

For any $n > 1$, $0 < \varepsilon < 1/2$, and $N > n^C$ for some constant $C > 0$, we show the existence of an N-point subset X of ℓ_2^n such that any linear map from X to ℓ_2^m with distortion at most $1 + \varepsilon$ must have $m = \Omega(\min\{n, \varepsilon^{-2} \lg N\})$. This improves a lower bound of Alon [Alon, Discre. Mathem., 1999], in the linear setting, by a $\lg(1/\varepsilon)$ factor. Our lower bound matches the upper bounds provided by the identity matrix and the Johnson-Lindenstrauss lemma [Johnson and Lindenstrauss, Contem. Mathem., 1984].

1998 ACM Subject Classification F.0 Theory of Computation

Keywords and phrases dimensionality reduction, lower bounds, Johnson-Lindenstrauss

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.82

1 Introduction

The Johnson-Lindenstrauss lemma [15] states the following.

▶ **Theorem 1** (JL lemma [15, Lemma 1]). *For any N-point subset X of Euclidean space and any $0 < \varepsilon < 1/2$, there exists a map $f : X \to \ell_2^m$ with $m = O(\varepsilon^{-2} \lg N)$ such that*

$$\forall x, y \in X,\ (1-\varepsilon)\|x-y\|_2^2 \leq \|f(x)-f(y)\|_2^2 \leq (1+\varepsilon)\|x-y\|_2^2. \tag{1}$$

We henceforth refer to f satisfying (1) as *having the ε-JL guarantee for X* (often we drop mention of ε when understood from context). The JL lemma has found applications in computer science, signal processing (e.g. compressed sensing), statistics, and mathematics. The main idea in algorithmic applications is that one can transform a high-dimensional problem into a low-dimensional one such that an optimal solution to the lower dimensional problem can be lifted to a nearly optimal solution to the original problem. Due to the decreased dimension, the lower dimensional problem requires fewer resources (time, memory, etc.) to solve. We refer the reader to [12, 28, 21] for a list of further applications.

All known proofs of the JL lemma with target dimension as stated above in fact provide such a map f which is *linear*. This linearity property is important in several applications. For example in the turnstile model of streaming [22], a vector $x \in \mathbb{R}^n$ receives a stream of coordinate-wise updates each of the form $x_i \leftarrow x_i + \Delta$, where $\Delta \in \mathbb{R}$. The goal is to

* KGL was supported by Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, grant DNRF84, a Villum Young Investigator Grant and an AUFF Starting Grant.
† JN was supported by NSF CAREER award CCF-1350670, NSF grant IIS-1447471, ONR grant N00014-14-1-0632 and Young Investigator award N00014-15-1-2388, and a Google Faculty Research Award.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 82; pp. 82:1–82:11


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

process x using $m \ll n$ memory. Thus if one wants to perform dimensionality reduction in a stream, which occurs for example in streaming linear algebra applications [7], this can be achieved with linear f since $f(x + \Delta \cdot e_i) = f(x) + \Delta \cdot f(e_i)$. In compressed sensing, another application where linearity of f is inherent, one wishes to (approximately) recover (approximately) sparse signals using few linear measurements [8, 6]. The map f sending a signal to the vector containing some fixed set of linear measurements of it is known to allow for good signal recovery as long as f satisfies the JL guarantee for the set of all k-sparse vectors [6]. Linear f is also inherent in model-based compressed sensing, which is similar but where one assumes the sparsity pattern cannot be an arbitrary one of $\binom{n}{k}$ sparsity patterns, but rather comes from a smaller, structured set [5].

Given the widespread use of dimensionality reduction across several domains, it is a natural and often-asked question whether the JL lemma is tight: does there exist some X of size N such that any such map f must have $m = \Omega(\min\{n, \varepsilon^{-2} \lg N\})$? The paper [15] introducing the JL lemma provided the first lower bound of $m = \Omega(\lg N)$ when ε is smaller than some constant. This was improved by Alon [3], who showed that if $X = \{0, e_1, \ldots, e_n\} \subset \mathbb{R}^n$ is the simplex (thus $N = n + 1$) and $0 < \varepsilon < 1/2$, then any JL map f must embed into dimension $m = \Omega(\min\{n, \varepsilon^{-2} \lg n / \lg(1/\varepsilon)\})$. Note the first term in the min is achieved by the identity map. Furthermore, the $\lg(1/\varepsilon)$ term cannot be removed for this particular X since one can use Reed-Solomon codes to obtain embeddings with $m = O(1/\varepsilon^2)$ (superior to the JL lemma) once $\varepsilon \leq n^{-\Omega(1)}$ [3] (see [23] for details). Specifically, for this X it is possible to achieve $m = O(\varepsilon^{-2} \min\{\lg N, ((\lg N)/\lg(1/\varepsilon))^2\})$. Note also for this choice of X we can assume that any f is in fact linear. This is because first we can assume $f(0) = 0$ by translation. Then we can form a matrix $A \in \mathbb{R}^{m \times n}$ such that the ith column of A is $f(e_i)$. Then trivially $Ae_i = f(e_i)$ and $A0 = 0 = f(0)$.

The fact that the JL lemma is not optimal for the simplex for small ε begs the question: is the JL lemma suboptimal for all point sets? This is a major open question in the area of dimensionality reduction, and it has been open since the paper of Johnson and Lindenstrauss 30 years ago.

Our Main Contribution: For any $n > 1$, $0 < \varepsilon < 1/2$, and $N > n^C$ for some constant $C > 0$, there is an N-point subset X of ℓ_2^n such that any embedding $f : X \to \ell_2^m$ providing the JL guarantee, and where f *is linear*, must have $m = \Omega(\min\{n, \varepsilon^{-2} \lg N\})$. In other words, the JL lemma is optimal in the case where f must be linear.

Our lower bound is optimal: the identity map achieves the first term in the min, and the JL lemma the second. It carries the restriction of only being against linear embeddings, but we emphasize that since the original JL paper [15] 31 years ago, every known construction achieving the JL guarantee has been linear. Thus, in light of our new contribution, the JL lemma cannot be improved without developing ideas that are radically different from those developed in the last three decades of research on the problem.

It is worth mentioning there have been important works on non-linear embeddings into Euclidean space, such as Sammon's mapping [14], Locally Linear Embeddings [26], ISOMAP [27], and Hessian eigenmaps [9]. None of these methods, however, is relevant to the current task. Sammon's mapping minimizes the *average* squared relative error of the embedded point distances, as opposed to the maximum relative error (see [14, Eqn. 1]). Locally linear embeddings, ISOMAP, and Hessian eigenmaps all assume the data lies on a d-dimensional manifold $\mathcal{M}$ in $\mathbb{R}^n$, $d \ll n$, and try to recover the d-dimensional parametrization given a few points sampled from $\mathcal{M}$. Furthermore, various other assumptions are made about the input, e.g. the analysis of ISOMAP assumes that geodesic distance on $\mathcal{M}$ is isometrically

embeddable into ℓ_2^d. Also, the way error in these works is meausured is again via some form of average squared error and not worst case relative error (e.g. [26, Eqn. 2]). The point in all these works is then not to show the *existence* of a good embedding into low dimensional Euclidean space (in fact these works study promise problems where one is promised to exist), but rather to show that a good embedding can be recovered, in some squared loss sense, if the input data is sampled sufficiently densely from $\mathcal{M}$. There has also been other work outside the manifold setting on providing good worst case distortion via non-linear embeddings in the TCS community [10], but this work (1) provides an embedding for the snowflake metric $\ell_2^{1/2}$ and not ℓ_2, and (2) does not achieve $1+\varepsilon$ distortion. Furthermore, differently from our focus, [10] assumes the input has bounded doubling dimension D, and the goal is to achieve target dimension and distortion being functions of D.

▶ Remark. It is worth noting that the JL lemma is different from the *distributional* JL (DJL) lemma that often appears in the literature, sometimes with the same name (though the lemmas are different!). In the DJL problem, one is given an integer $n > 1$ and $0 < \varepsilon, \delta < 1/2$, and the goal is to provide a distribution $\mathcal{F}$ over maps $f : \ell_2^n \to \ell_2^m$ with m as small as possible such that for any fixed $x \in \mathbb{R}^n$

$$\mathop{\mathbb{P}}_{f \leftarrow \mathcal{F}}(\|f(x)\|_2 \notin [(1-\varepsilon)\|x\|_2, (1+\varepsilon)\|x\|_2]) < \delta.$$

The existence of such $\mathcal{F}$ with small m implies the JL lemma by taking $\delta < 1/\binom{N}{2}$. Then for any $z \in X - X$, a random $f \leftarrow \mathcal{F}$ fails to preserve the norm of z with probability δ. Thus the probability that there exists $z \in X - X$ which f fails to preserve the norm of is at most $\delta \cdot \binom{N}{2} < 1$, by a union bound. In other words, a random map provides the desired JL guarantee with high probability (and in fact this map is chosen completely obliviously of the input vectors).

The optimal m for the DJL lemma when using linear maps is understood. The original paper [15] provided a linear solution to the DJL problem with $m = O(\min\{n, \varepsilon^{-2}\lg(1/\delta)\})$, and this was later shown to be optimal for the full range of $\varepsilon, \delta \in (0, 1/2)$ [13, 16]. Thus when δ is set as above, one obtains the $m = O(\varepsilon^{-2}\lg N)$ guarantee of the JL lemma. However, this does not imply that the JL lemma is tight. Indeed, it is sometimes possible to obtain smaller m by avoiding the DJL lemma, such as the Reed-Solomon based embedding construction for the simplex mentioned above (which involves zero randomness).

It is also worth remarking that DJL is desirable for one-pass streaming algorithms, since no properties of X are known when the map f is chosen at the beginning of the stream, and thus the DJL lower bounds of [13, 16] are relevant in this scenario. However when allowed two passes or more, one could imagine estimating various properties of X in the first pass(es) then choosing some f more efficiently based on these properties to perform dimensionality reduction in the last pass. The approach of using the first pass(es) to estimate characteristics of a stream to then more efficiently select a linear sketch to use in the last pass is in fact a common technique in streaming algorithms. For example, [18] used such an approach to design a nearly optimal two-pass algorithm for L_0-estimation in turnstile streams, which consumes nearly a logarithmic factor less memory than the one-pass lower bound for the same problem. In fact all known turnstile streaming algorithms, even those using multiple passes, maintain *linear* maps applied to the input stream (with linear maps in subsequent passes being functions of data collected from applying linear maps in previous passes). It is even reasonable to conjecture that the most space-efficient algorithm for *any* multi-pass turnstile streaming problem for ℓ_2 dimensionality reduction must be of this form, since the recent works [20, 2] give evidence in this direction: namely that if a multi-pass algorithm is viewed as a sequence of finite automata (one for each pass), where the ith automaton is

generated solely from the output of the $(i-1)$st automaton, then it can be assumed that all automata represent *linear maps* with at most a logarithmic factor loss in space. They give examples where this logarithmic factor loss is necessary, but for many problems we know that no loss is necessary when requiring linear maps [4, 17]. Our new lower bound thus gives evidence that one cannot improve dimensionality reduction in the streaming setting even when given multiple passes.

1.1 Proof overview

For any $n > 1$ and $\varepsilon \in (0, 1/2)$ and $N > poly(n)$, we prove the existence of $X \subset \mathbb{R}^n$, $|X| = N$, s.t. if for $A \in \mathbb{R}^{m\times n}$

$$(1-\varepsilon)\|x\|_2^2 \le \|Ax\|_2^2 \le (1+\varepsilon)\|x\|_2^2 \text{ for all } x \in X, \tag{2}$$

then $m = \Omega(\min\{n, \varepsilon^{-2}\lg N\})$. Providing the JL guarantee on $X \cup \{0\}$ implies satisfying (2), and therefore also requires $m = \Omega(\min\{n, \varepsilon^{-2}\lg N\})$. We show such X exists via the probabilistic method, by letting X be the union of all n standard basis vectors together with several independent gaussian vectors. Gaussian vectors were also the hard case in the DJL lower bound proof of [16], though the details were different. Note we can assume $N < \exp(C\varepsilon^2 n)$ for any $C > 0$ we choose, since for larger N the n in the minimum defining m takes effect.

We now give the idea of the lower bound proof to achieve (2). First, we include in X the vectors $e_1, \ldots, e_n$. Then if $A \in \mathbb{R}^{m\times n}$ for $m \le n$ satisfies (2), this forces every column of A to have roughly unit norm. Then by standard results in covering and packing (see Eqn. (5.7) of [25]), there exists some family of matrices $\mathcal{F} \subset \cup_{t=1}^{n}\mathbb{R}^{t\times n}$, $|\mathcal{F}| = e^{O(n^2\lg n)}$, such that

$$\inf_{\hat{A}\in\mathcal{F}\cap\mathbb{R}^{m\times n}} \|A - \hat{A}\|_F \le \frac{1}{n^C} \tag{3}$$

for $C > 0$ a constant as large as we like, where $\|\cdot\|_F$ denotes Frobenius norm. Also, by a theorem of Latała [19], for any $\hat{A} \in \mathcal{F}$ and for a random gaussian vector g,

$$\mathbb{P}_g(|\|\hat{A}g\|_2^2 - \mathrm{tr}(\hat{A}^T\hat{A})| \ge \Omega(\sqrt{\lg(1/\delta)}\cdot\|\hat{A}^T\hat{A}\|_F)) \ge \delta/2 \tag{4}$$

for any $0 < \delta < 1/2$, where $\mathrm{tr}(\cdot)$ is trace. This is a (weaker version of the) statement that for gaussians, the Hanson-Wright inequality [11] not only provides an upper bound on the tail of degree-two gaussian chaos, but also is a lower bound. (The strong form of the previous sentence, without the parenthetical qualifier, was proven in [19], but we do not need this stronger form for our proof – essentially the difference is that in stronger form, (4) is replaced with a stronger inequality also involving the operator norm $\|\hat{A}^T\hat{A}\|$.)

It also follows by standard results that a random gaussian vector g satisfies

$$\mathbb{P}_g(|\|g\|_2^2 - n| > C\sqrt{n\lg(1/\delta)}) < \delta/2 \tag{5}$$

Thus by a union bound, the events of (4), (5) happen simultaneously with probability $\Omega(\delta)$. Thus if we take N random gaussian vectors, the probability that the events of (4), (5) never happen simultaneously for any of the N gaussians is at most $(1-\Omega(\delta))^N = e^{-\Omega(\delta N)}$. By picking N sufficiently large and $\delta = 1/\mathrm{poly}(n)$, a union bound over $\mathcal{F}$ shows that for every $\hat{A} \in \mathcal{F}$, one of the N gaussians satisfies the events of (4) and (5) simultaneously. Specifically, for $N > n^3$ there exist N vectors $\{v_1, \ldots, v_N\} = V \subset \mathbb{R}^n$ such that

- Every $v \in V$ has $\|v\|_2^2 = n \pm O(\sqrt{n\lg N}) = (1 \pm O(\varepsilon))n$.
- For any $\hat{A} \in \mathcal{F}$ there exists some $v \in V$ such that $|\|\hat{A}v\|_2^2 - \mathrm{tr}(\hat{A}^T\hat{A})| = \Omega(\sqrt{\lg N}\cdot\|\hat{A}^T\hat{A}\|_F)$.

Here $\pm B$ represents a value in $[-B, B]$. The final definition of X is $\{e_1, \ldots, e_n\} \cup V$. Then, using (2) and (3), we show that the second bullet implies

$$\mathrm{tr}(\hat{A}^T\hat{A}) = n \pm O(\varepsilon n), \text{ and } |\|Av\|_2^2 - n| = \Omega(\sqrt{\lg N} \cdot \|\hat{A}^T\hat{A}\|_F) - O(\varepsilon n). \tag{6}$$

But then by the triangle inequality, the first bullet above, and (2),

$$\left|\|Av\|_2^2 - n\right| \leq \left|\|Av\|_2^2 - \|v\|_2^2\right| + \left|\|v\|\|_2^2 - n\right| = O(\varepsilon n). \tag{7}$$

Combining (6) and (7) implies

$$\mathrm{tr}(\hat{A}^T\hat{A}) = \sum_{i=1}^{n} \hat{\lambda}_i \geq (1 - O(\varepsilon))n, \text{ and } \|\hat{A}^T\hat{A}\|_F^2 = \sum_{i=1}^{n} \hat{\lambda}_i^2 = O\left(\frac{\varepsilon^2 n^2}{\lg N}\right)$$

where $(\hat{\lambda}_i)$ are the eigenvalues of $\hat{A}^T\hat{A}$. With bounds on $\sum_i \hat{\lambda}_i$ and $\sum_i \hat{\lambda}_i{}^2$ in hand, a lower bound on $\mathrm{rank}(\hat{A}^T\hat{A}) \leq m$ follows by Cauchy-Schwarz (this last step is also common to the proof of [3]).

▶ Remark. It is not crucial in our proof that N be at least n^3. Our techniques straightforwardly extend to show that N can be any value which is $\Omega(n^{2+\gamma})$ for any constant $\gamma > 0$, or even $\Omega(n^{1+\gamma}/\varepsilon^2)$.

2 Preliminaries

Henceforth a *standard gaussian* random variable $g \in \mathbb{R}$ is a gaussian with mean 0 and variance 1. If we say $g \in \mathbb{R}^n$ is standard gaussian, then we mean that g is a multivariate gaussian with identity covariance matrix (i.e. its entries are independent standard gaussian). Also, the notation $\pm B$ denotes a value in $[-B, B]$. For a real matrix $A = (a_{i,j})$, $\|A\|$ is the $\ell_2 \to \ell_2$ operator norm, and $\|A\|_F = (\sum_{i,j} a_{i,j}^2)^{1/2}$ is Frobenius norm.

In our proof we depend on some previous work. The first theorem is due to Latała [19] and says that, for gaussians, the Hanson-Wright inequality is not only an upper bound but also a lower bound.

▶ **Theorem 2** ([19, Corollary 2]). *There exists universal $c > 0$ such that for $g \in \mathbb{R}^n$ standard gaussian and $A = (a_{i,j})$ an $n \times n$ real symmetric matrix with zero diagonal,*

$$\forall t \geq 1,\ \mathop{\mathbb{P}}_{g}\left(|g^T A g| > c(\sqrt{t} \cdot \|A\|_F + t \cdot \|A\|)\right) \geq \min\{c, e^{-t}\}.$$

Theorem 2 implies the following corollary.

▶ **Corollary 3.** *Let g, A be as in Theorem 2, but where A is no longer restricted to have zero diagonal. Then*

$$\forall t \geq 1,\ \mathop{\mathbb{P}}_{g}\left(|g^T A g - tr(A)| > c(\sqrt{t} \cdot \|A\|_F + t \cdot \|A\|)\right) \geq \min\{c, e^{-t}\}.$$

Proof. Let N be a positive integer. Define $\tilde{g} = (\tilde{g}_{1,1}, \tilde{g}_{1,2}, \ldots, \tilde{g}_{1,N}, \ldots, \tilde{g}_{n,1}, \tilde{g}_{n,2}, \ldots, \tilde{g}_{n,N})$ a standard gaussian vector. Then g_i is equal in distribution to $N^{-1/2}\sum_{j=1}^{N} \tilde{g}_{i,j}$. Define $\tilde{A}_N$ as the $nN \times nN$ matrix formed by converting each entry $a_{i,j}$ of A into an $N \times N$ block with each entry being $a_{i,j}/N$. Then

$$g^T A g - \mathrm{tr}(A) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j} g_i g_j - \mathrm{tr}(A) \overset{d}{=} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{r=1}^{N}\sum_{s=1}^{N} \frac{a_{i,j}}{N} \tilde{g}_{i,r}\tilde{g}_{j,s} - \mathrm{tr}(A) \overset{\text{def}}{=} \tilde{g}^T \tilde{A}_N \tilde{g} - \mathrm{tr}(\tilde{A}_N)$$

where $\stackrel{d}{=}$ denotes equality in distribution (note $\mathrm{tr}(A) = \mathrm{tr}(\tilde{A}_N)$). By the weak law of large numbers

$$\forall \lambda > 0,\ \lim_{N\to\infty} \mathbb{P}\left(|\tilde{g}^T \tilde{A}_N \tilde{g} - \mathrm{tr}(\tilde{A}_N)| > \lambda\right) = \lim_{N\to\infty} \mathbb{P}\left(|\tilde{g}^T (\tilde{A}_N - \tilde{D}_N)\tilde{g}| > \lambda\right) \tag{8}$$

where $\tilde{D}_N$ is diagonal containing the diagonal elements of $\tilde{A}_N$. Note $\|\tilde{A}_N\| = \|A\|$. This follows since if we have the singular value decomposition $A = \sum_i \sigma_i u_i v_i^T$ (where the $\{u_i\}$ and $\{v_i\}$ are each orthonormal, $\sigma_i > 0$, and $\|A\|$ is the largest of the σ_i), then $\tilde{A}_N = \sum_i \sigma_i u_i^{(N)} (v_i^{(N)})^T$ where $u_i^{(N)}$ is equal to u_i but where every coordinate is replicated N times and divided by $\sqrt{N}$. This implies $|\|\tilde{A}_N - \tilde{D}_N\| - \|A\|| \le \|\tilde{D}_N\| = \max_i |a_{i,i}|/N = o_N(1)$ by the triangle inequality. Therefore $\lim_{N\to\infty} \|\tilde{A}_N - \tilde{D}_N\| = \|A\|$. Also $\lim_{N\to\infty} \|\tilde{A}_N - \tilde{D}_N\|_F = \|A\|_F$. Our corollary follows by applying Theorem 2 to the right side of (8). ◀

The next lemma follows from gaussian concentration of Lipschitz functions [24, Corollary 2.3]. It also follows from the Hanson-Wright inequality [11] (which is the statement of Corollary 3, but with the inequality reversed). Ultimately we will apply it with $t \in \Theta(\lg n)$, in which case the e^{-t} term will dominate.

▶ **Lemma 4.** *For a universal $c > 0$, and $g \in \mathbb{R}^n$ standard gaussian, $\forall t > 0$ $\mathbb{P}(|\|g\|_2^2 - n| > c\sqrt{nt}) < e^{-t} + e^{-\sqrt{nt}}$.*

The following corollary summarizes the above in a form that will be useful later.

▶ **Corollary 5.** *For $A \in \mathbb{R}^{m\times n}$ let $\lambda_1 \ge \cdots \ge \lambda_n \ge 0$ be the eigenvalues of $A^T A$. Let $g^{(1)}, \ldots, g^{(N)} \in \mathbb{R}^n$ be independent standard gaussian vectors. For some universal constants $c_1, c_2, \delta_0 > 0$ and any $0 < \delta < \delta_0$*

$$\mathbb{P}\left(\nexists j \in [N] : \left\{ \left| \|Ag^{(j)}\|_2^2 - \sum_{i=1}^n \lambda_i \right| \ge c_1 \sqrt{\lg(1/\delta)} \left(\sum_{i=1}^n \lambda_i^2 \right)^{1/2} \right\} \wedge \left\{ |\|g^{(j)}\|_2^2 - n| \le c_2 \sqrt{n \lg(1/\delta)} \right\} \right) \le e^{-N\delta}. \tag{9}$$

Proof. We will show that for any fixed $j \in [N]$ it holds that

$$\mathbb{P}\left(\left\{ \left| \|Ag^{(j)}\|_2^2 - \sum_{i=1}^n \lambda_i \right| \ge c_1 \sqrt{\lg(1/\delta)} \left(\sum_{i=1}^n \lambda_i^2 \right)^{1/2} \right\} \wedge \left\{ \|g^{(j)}\|_2^2 \le n + c_2 \sqrt{n \lg(1/\delta)} \right\} \right) > \delta \tag{10}$$

Then, since the g_j are independent, the left side of (9) is at most $(1-\delta)^N \le e^{-\delta N}$.

Now we must show (10). It suffices to show that

$$\mathbb{P}\left(|\|g^{(j)}\|_2^2 - n| \le c_2 \sqrt{n \lg(1/\delta)} \right) > 1 - \delta/2 \tag{11}$$

and

$$\mathbb{P}\left(\left| \|Ag^{(j)}\|_2^2 - \sum_{i=1}^n \lambda_i \right| \ge c_1 \sqrt{\lg(1/\delta)} \left(\sum_{i=1}^n \lambda_i^2 \right)^{1/2} \right) > \delta/2 \tag{12}$$

since (10) would then follow from a union bound. Eqn. (11) follows immediately from Lemma 4 for c_2 chosen sufficiently large. For Eqn. (12), note $\|Ag^{(j)}\|_2^2 = g^T A^T A g$. Then $\sum_i \lambda_i = \mathrm{tr}(A^T A)$ and $(\sum_i \lambda_i^2)^{1/2} = \|A^T A\|_F$. Then (12) frollows from Corollary 3 for δ smaller than some sufficiently small constant δ_0. ◀

We also need a standard estimate on entropy numbers (covering the unit ℓ_∞^{mn} ball by ℓ_2^{mn} balls).

▶ **Lemma 6.** *For any parameter $0 < \alpha < 1$, there exists a family $\mathcal{F}_\alpha \subseteq \bigcup_{m=1}^n \mathbb{R}^{m\times n}$ of matrices with the following two properties:*

1. *For any matrix $A \in \bigcup_{m=1}^n \mathbb{R}^{m\times n}$ having all entries bounded in absolute value by 2, there is a matrix $\hat{A} \in \mathcal{F}_\alpha$ such that A and $\hat{A}$ have the same number of rows and $B = A - \hat{A}$ satisfies $tr(B^TB) \le \alpha/100$.*
2. *$|\mathcal{F}_\alpha| = e^{O(n^2 \lg(n/\alpha))}$.*

Proof. We construct $\mathcal{F}_\alpha$ as follows: For each integer $1 \le m \le n$, add all $m \times n$ matrices having entries of the form $i\frac{\sqrt{\alpha}}{10n}$ for integers $i \in \{-20n/\sqrt{\alpha}, \ldots, 20n/\sqrt{\alpha}\}$. Then for any matrix $A \in \bigcup_{m=1}^n \mathbb{R}^{m\times n}$ there is a matrix $\hat{A} \in \mathcal{F}_\alpha$ such that A and $\hat{A}$ have the same number of rows and every entry of $B = A - \hat{A}$ is bounded in absolute value by $\frac{\sqrt{\alpha}}{10n}$. This means that every diagonal entry of B^TB is bounded by $n\alpha/(100n^2)$ and thus $\mathrm{tr}(B^TB) \le \alpha/100$. The size of $\mathcal{F}_\alpha$ is bounded by $n(40n/\sqrt{\alpha})^{n^2} = e^{O(n^2 \lg(n/\alpha))}$. ◀

3 Proof of main theorem

▶ **Lemma 7.** *Let $\mathcal{F}_\alpha$ be as in Lemma 6 with $1/\,poly(n) \le \alpha < 1$. Then there for any $N > n^3$ there exists a set of N vectors $v_1, \ldots, v_N$ in $\mathbb{R}^n$ such that for every matrix $A \in \mathcal{F}_\alpha$, there is an index $j \in [N]$ such that*

(i) $|\|Av_j\|_2^2 - \sum_i \lambda_i| = \Omega\left(\sqrt{\lg N \sum_i \lambda_i^2}\right)$.

(ii) $|\|v_j\|_2^2 - n| = O(\sqrt{n \lg N})$.

Proof. Let $g^{(1)}, \ldots, g^{(N)} \in \mathbb{R}^n$ be independent standard gaussian. Let $A \in \mathcal{F}_\alpha$ and apply Corollary 5 with $\delta = N^{-1/12} \le n^{-1/4}$. With probability $1 - e^{-\Omega(n^{3-1/4})}$, one of the $g^{(j)}$ for $j \in [N]$ satisfies (i) and (ii) for A. Since $|\mathcal{F}_\alpha| = e^{O(n^2 \lg(n/\alpha))}$, the claim follows by a union bound over all matrices in $\mathcal{F}_\alpha$. ◀

▶ **Theorem 8.** *For any $0 < \varepsilon < 1/2$, $n > 1$, and $n' > n^3$, there exists a set $X \subset \mathbb{R}^n$, $|X| = N = n' + n$, such that if A is a matrix in $\mathbb{R}^{m\times n}$ satisfying $\|Av_i\|_2^2 \in (1 \pm \varepsilon)\|v_i\|_2^2$ for all $v_i \in X$, then $m = \Omega(\min\{n, \varepsilon^{-2} \lg N\})$.*

Proof. We can assume $\varepsilon > 1/\sqrt{n}$ since otherwise an $m = \Omega(n)$ lower bound already follows from [3]. We also can assume $N < \exp(C\varepsilon^2 n)$, since otherwise $\min\{n, \varepsilon^{-2} \lg N\} = n$. To construct X, we first invoke Lemma 7 with $\alpha = \varepsilon^2/n^2$ to find n' vectors $w_1, \ldots, w_{n'}$ such that for all matrices $\tilde{A} \in \mathcal{F}_{\varepsilon^2/n^2}$, there exists an index $j \in [n']$ for which:

1. $|\|\tilde{A}w_j\|_2^2 - \sum_i \tilde{\lambda}_i| \ge \Omega\left(\sqrt{(\lg N)\sum_i \tilde{\lambda}_i^2}\right)$.
2. $|\|w_j\|_2^2 - n| = O(\sqrt{n \lg N}) = O(\varepsilon n)$.

where $\tilde{\lambda}_1 \ge \cdots \ge \tilde{\lambda}_n \ge 0$ denote the eigenvalues of $\tilde{A}^T\tilde{A}$. We let $X = \{e_1, \ldots, e_n, w_1, \ldots, w_{n'}\}$ and claim this set of $N = n' + n$ vectors satisfies the theorem. Here e_i denotes the i'th standard unit vector.

To prove this, let $A \in \mathbb{R}^{m\times n}$ be a matrix with $m \le n$ satisfying $\|Av\|_2^2 \in (1 \pm \varepsilon)\|v\|_2^2$ for all $v \in X$. Now observe that since $e_1, \ldots, e_n \in X$, A satisfies $\|Ae_i\|_2^2 \in (1 \pm \varepsilon)\|e_i\|_2^2 = (1 \pm \varepsilon)$ for all e_i. Hence all entries $a_{i,j}$ of A must have $a_{i,j}^2 \le (1 + \varepsilon) < 2$ (and in fact, all columns of A have ℓ_2 norm at most $\sqrt{2}$). This implies that there is an $m \times n$ matrix $\hat{A} \in \mathcal{F}_{\varepsilon^2/n^2}$ such

that $B = A - \hat{A} = (b_{i,j})$ satisfies $\operatorname{tr}(B^T B) \leq \varepsilon^2/(100n^2)$. Since $\operatorname{tr}(B^T B) = \|B\|_F^2$, this also implies $\|B\|_F \leq \varepsilon/(10n)$. Then by Cauchy-Schwarz,

$$\begin{aligned}
\sum_{i=1}^{n} \hat{\lambda}_i &= \operatorname{tr}(\hat{A}^T \hat{A}) \\
&= \operatorname{tr}((A-B)^T(A-B)) \\
&= \operatorname{tr}(A^T A) + \operatorname{tr}(B^T B) - \operatorname{tr}(A^T B) - \operatorname{tr}(B^T A) \\
&= \sum_{i=1}^{n} \|Ae_i\|_2^2 + \operatorname{tr}(B^T B) - \operatorname{tr}(A^T B) - \operatorname{tr}(B^T A) \\
&= n \pm (O(\varepsilon n) + 2n \cdot \max_j (\sum_i b_{i,j}^2)^{1/2} \cdot \max_k (\sum_i a_{i,k}^2)^{1/2}) \\
&= n \pm (O(\varepsilon n) + 2n \cdot \|B\|_F \cdot \sqrt{2}) \\
&= n \pm O(\varepsilon n).
\end{aligned}$$

Thus from our choice of X there exists a vector $v^* \in X$ such that

(i) $|\|\hat{A}v^*\|_2^2 - n| \geq \Omega\left(\sqrt{(\lg N)\sum_i \hat{\lambda}_i^2}\right) - O(\varepsilon n)$.

(ii) $|\|v^*\|_2^2 - n| = O(\sqrt{n \lg N}) = O(\varepsilon n)$.

Note $\|B\|^2 \leq \|B\|_F^2 = \operatorname{tr}(B^T B) \leq \varepsilon^2/(100n^2)$ and $\|\hat{A}\|^2 \leq \|\hat{A}\|_F^2 \leq (\|A\|_F + \|B\|_F)^2 = O(n^2)$. Then by (i)

(iii)

$$\begin{aligned}
|\|Av^*\|_2^2 - n| &= |\|\hat{A}v^*\|_2^2 + \|Bv^*\|_2^2 + 2\langle \hat{A}v^*, Bv^*\rangle - n| \\
&\geq \Omega\left(\sqrt{(\lg N)\sum_i \hat{\lambda}_i^2}\right) - \|Bv^*\|_2^2 - 2|\langle \hat{A}v^*, Bv^*\rangle| - O(\varepsilon n) \\
&\geq \Omega\left(\sqrt{(\lg N)\sum_i \hat{\lambda}_i^2}\right) - \|B\|^2 \cdot \|v^*\|_2^2 - 2\|B\| \cdot \|A\| \cdot \|v^*\|_2^2 - O(\varepsilon n) \\
&= \Omega\left(\sqrt{(\lg N)\sum_i \hat{\lambda}_i^2}\right) - O(\varepsilon n).
\end{aligned}$$

We assumed $|\|Av^*\|_2^2 - \|v^*\|_2^2| = O(\varepsilon \|v^*\|_2^2) = O(\varepsilon n)$. Therefore by (ii),

$$\left|\|Av^*\|_2^2 - n\right| \leq \left|\|Av^*\|_2^2 - \|v^*\|_2^2\right| + \left|\|v^*\|_2^2 - n\right| = O(\varepsilon n)$$

which when combined with (iii) implies

$$\sum_{i=1}^{n} \hat{\lambda}_i^2 = O\left(\frac{\varepsilon^2 n^2}{\lg N}\right).$$

To complete the proof, by Cauchy-Schwarz since exactly $\operatorname{rank}(\hat{A}^T \hat{A})$ of the $\hat{\lambda}_i$ are non-zero,

$$\frac{n^2}{2} \leq \left(\sum_{i=1}^{n} \hat{\lambda}_i\right)^2 \leq \operatorname{rank}(\hat{A}^T \hat{A}) \cdot \left(\sum_{i=1}^{n} \hat{\lambda}_i{}^2\right) \leq m \cdot O\left(\frac{\varepsilon^2 n^2}{\lg N}\right).$$

Rearranging gives $m = \Omega(\varepsilon^{-2} \lg N)$. Note we assumed $N < \exp(C\varepsilon^2 n)$. Thus considering N larger, we obtain the lower bound $m = \Omega(\min\{n, \varepsilon^{-2} \lg N\})$ as desired. ◀

4 Discussion

One obvious future goal is to obtain an $m = \Omega(\min\{n, \varepsilon^{-2}\lg N\})$ lower bound that also applies to non-linear maps. Unfortunately, such a lower bound cannot be obtained by using the hard set X from Theorem 8. If X is the union of $\{e_1, \ldots, e_n\}$ with $n^{O(1)}$ independent gaussian vectors normalized to each have squared unit norm in expectation, then it is not hard to show (e.g. via a decoupled Hanson-Wright inequality) that X will be ε-incoherent with high probability for any $\varepsilon \in \Omega(\sqrt{\lg n/n})$, where we say a set X is ε-incoherent if (1) for all $x \in X$, $\|x\|_2 = 1 \pm \varepsilon$, and (2) for all $x \neq y \in X$, $|\langle x, y\rangle| \leq \varepsilon$. It is known that any ε-incoherent set of N vectors can be *non-linearly* embedded into dimension $O(\varepsilon^{-2}(\lg N/(\lg\lg N + \lg(1/\varepsilon)))^2)$ by putting each vector in correspondence with a Reed-Solomon codeword (see [23] for details). This upper bound is $o(\varepsilon^{-2}\lg N)$ for any $\varepsilon \in 2^{-\omega(\sqrt{\lg N})}$. Thus, one cannot prove an $\Omega(\varepsilon^{-2}\lg N)$ lower bound against non-linear maps for our hard set X for the full range of $\varepsilon \in [\sqrt{(\lg n)/n}, 1/2]$.

One potential avenue for generalizing our lower bound to the non-linear setting is to shrink $|X|$. Our hard set X contains $N = O(n^3)$ points in $\mathbb{R}^n$ (though as remarked earlier, our techniques easily imply $N = O(n^{1+\gamma}/\varepsilon^2)$ points suffice). Any embedding f could be assumed linear without loss of generality if the elements of X were linearly independent, at which point one would only need to prove a lower bound against linear embeddings. However, clearly $X \subset \mathbb{R}^n$ cannot be linearly independent if $N > n$, as is the case for our X. Thus a first step toward a lower bound against non-linear embeddings is to obtain a hard X with N as small as possible. Alternatively, one could hope to extend the aforementioned non-linear embedding upper bound for incoherent sets of vectors to arbitrary sets of vectors, though such a result if true seems to require ideas very different from all known constructions of JL transforms to date.

Finally, we mention an alternative but similar proof strategy that leads to the same result as proved above. In [16], the authors proved the following theorem (see their Theorem 9):

▶ **Theorem 9** ([16]). *If $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear transformation with $n \geq 2m$ and $\varepsilon > 0$ is sufficiently small, then for g a randomly chosen vector in $\mathcal{S}^{n-1}$, $\mathbb{P}(|\|Ag\|_2^2 - 1| > \varepsilon) \geq \exp(-O(m\varepsilon^2 + 1))$.*

With this theorem in mind, we can redo our proof steps, first showing that for a matrix $A \in \mathbb{R}^{m\times n}$ and N randomly chosen vectors $g^{(1)}, \ldots, g^{(N)}$, at least one of them will have $|\|Ag^{(j)}\|_2^2 - 1| > \varepsilon$ with probability $1 - \exp(-N\exp(O(m\varepsilon^2 + 1)))$. If $m = o(\varepsilon^{-2}\lg N)$, we can prove an analog of our Lemma 7, showing that there exists a set $X \subset \mathcal{S}^{n-1}$ of $N > n^3$ vectors $v_1, \ldots, v_N$, such that for every matrix $A \in \mathcal{F}_\alpha$, there is an index $j \in [N]$ with $|\|Av_j\|_2^2 - 1| > \varepsilon$. Finally, we could redo the steps in the proof of Theorem 8, showing that any JL-matrix for $X \cup \{e_1, \ldots, e_n\}$ must be sufficiently "close" to a matrix in $\mathcal{F}_\alpha$ and hence there is a vector v_j in X whose norm is distorted by too much. In summary, their theorem would replace our Corollary 3. The proof of their theorem is slightly more involved than the proof of Corollary 3 (once one assumes Theorem 2), albeit not by much. We believe there is value in knowing both proofs and we hope the underlying ideas may be useful in other applications.

Acknowledgments. We thank Radosław Adamczak for pointing out how to derive Corollary 3 from Theorem 2, and for pointing out the reference [1], which uses a more involved but similar argument. We also thank Yi Li for helpful conversation.

References

1. Radosław Adamczak and Paweł Wolff. Concentration inequalities for non-Lipschitz functions with bounded derivatives of higher order. *CoRR*, abs/1304.1826, 2013.
2. Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New characterizations in turnstile streams with applications. In *Proceedings of the 31st Annual Conference on Computational Complexity (CCC)*, 2016.
3. Noga Alon. Problems and results in extremal combinatorics–I. *Discrete Mathematics*, 273(1-3):31–53, 2003.
4. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.
5. Richard G. Baraniuk, Volkan Cevher, Marco F. Duarte, and Chinmay Hegde. Model-based compressive sensing. *IEEE Trans. Inf. Theory*, 56:1982–2001, 2010.
6. Emmanuel Candès and Terence Tao. Decoding by linear programming. *IEEE Trans. Inf. Theory*, 51(12):4203–4215, 2005.
7. Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, (STOC)*, pages 205–214, 2009.
8. David Donoho. Compressed sensing. *IEEE Trans. Inf. Theory*, 52(4):1289–1306, 2006.
9. David L. Donoho and Carrie Grimes. Hessian eigenmaps: locally linear embedding techniques for high-dimensional data. *PNAS*, 100(10):5591–5596, 2003.
10. Lee-Ad Gottlieb and Robert Krauthgamer. A nonlinear approach to dimension reduction. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 888–899, 2011.
11. David Lee Hanson and Farroll Tim Wright. A bound on tail probabilities for quadratic forms in independent random variables. *Ann. Math. Statist.*, 42(3):1079–1083, 1971.
12. Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 10–33, 2001.
13. T. S. Jayram and David P. Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms*, 9(3):26, 2013.
14. Jr. John W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, 18:401–409, 1969.
15. William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.
16. Daniel M. Kane, Raghu Meka, and Jelani Nelson. Almost optimal explicit Johnson-Lindenstrauss families. In *Proceedings of the 15th International Workshop on Randomization and Computation (RANDOM)*, pages 628–639, 2011.
17. Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1161–1178, 2010.
18. Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 41–52, 2010.
19. Rafał Latała. Tail and moment estimates for some types of chaos. *Studia Math.*, 135:39–53, 1999.
20. Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the 46th ACM Symposium on Theory of Computing (STOC)*, pages 174–183, 2014.

21 Jirí Matousek. On variants of the Johnson-Lindenstrauss lemma. *Random Struct. Algorithms*, 33(2):142–156, 2008.

22 S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

23 Jelani Nelson, Huy L. Nguyễn, and David P. Woodruff. On deterministic sketching and streaming for sparse recovery and norm estimation. *Linear Algebra and its Applications, Special Issue on Sparse Approximate Solution of Linear Systems*, 441:152–167, 2014.

24 Gilles Pisier. Probabilistic methods in the geometry of Banach spaces. *Probability and Analysis, Lecture Notes in Math.*, 1206:167–241, 1986.

25 Gilles Pisier. *The volume of convex bodies and Banach space geometry*, volume 94 of *Cambridge Tracts in Mathematics*. Cambridge University Press, 1989.

26 Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

27 Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

28 Santosh Vempala. *The random projection method*, volume 65 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 2004.

Impossibility of Sketching of the 3D Transportation Metric with Quadratic Cost*

Alexandr Andoni[†1], **Assaf Naor**[‡2], **and Ofer Neiman**[§3]

1 **Computer Science Department, Columbia University, New York, NY, USA**
andoni@cs.columbia.edu
2 **Mathematics Department, Princeton University, Princeton, NJ, USA**
naor@math.princeton.edu
3 **Department of Computer Science, Ben-Gurion University of the Negev, Be'er Sheva, Israel**
neimano@cs.bgu.ac.il

Abstract

Transportation cost metrics, also known as the Wasserstein distances W_p, are a natural choice for defining distances between two pointsets, or distributions, and have been applied in numerous fields. From the computational perspective, there has been an intensive research effort for understanding the W_p metrics over $\mathbb{R}^k$, with work on the W_1 metric (a.k.a *earth mover distance*) being most successful in terms of theoretical guarantees. However, the W_2 metric, also known as the *root-mean square* (RMS) bipartite matching distance, is often a more suitable choice in many application areas, e.g. in graphics. Yet, the geometry of this metric space is currently poorly understood, and efficient algorithms have been elusive. For example, there are no known non-trivial algorithms for nearest-neighbor search or sketching for this metric.

In this paper we take the first step towards explaining the lack of efficient algorithms for the W_2 metric, even over the three-dimensional Euclidean space $\mathbb{R}^3$. We prove that there are no meaningful embeddings of W_2 over $\mathbb{R}^3$ into a wide class of normed spaces, as well as that there are no efficient sketching algorithms for W_2 over $\mathbb{R}^3$ achieving constant approximation. For example, our results imply that: 1) any embedding into L_1 must incur a distortion of $\Omega(\sqrt{\log n})$ for pointsets of size n equipped with the W_2 metric; and 2) any sketching algorithm of size s must incur $\Omega\left(\sqrt{\log n}/\sqrt{s}\right)$ approximation. Our results follow from a more general statement, asserting that W_2 over $\mathbb{R}^3$ contains the 1/2-snowflake of *all* finite metric spaces with a uniformly bounded distortion. These are the first non-embeddability/non-sketchability results for W_2.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems – Geometrical problems and computations

Keywords and phrases Transportation metric, embedding, snowflake, sketching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.83

1 Introduction

Transportation metrics provide a natural distance on sets of points, or probability measures more generally, and as such have applications in numerous fields, such as computer science, as

* An extended version of this paper is available at http://arxiv.org/abs/1509.08677. The present paper contains some material that is not covered in that version.
† A. A. was supported in part by the NSF.
‡ A. N. was supported in part by the BSF, the Packard Foundation and the Simons Foundation.
§ O. N. was supported in part by the ISF and the European Union's Seventh Framework Programme.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 83; pp. 83:1–83:14


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

well as statistical physics, mathematical economics, automated control, shape optimization, applied probability, partial differential equations, metric geometry and many more, see [44, 39]. These metrics are also known as Wasserstein distance, Kantorovich-Rubinstein distance, Prokhorov distance, or the earth mover distance. We now recall basic notation and terminology from the theory of transportation cost metrics [57]. For a metric space (X, d_X) and $p \in (0, \infty)$, let $\mathcal{P}_p(X)$ denote the space of all (Borel) probability measures μ on X satisfying $\int_X d_X(x, x_0)^p \mathrm{d}\mu(x) < \infty$ for some (hence all) $x_0 \in X$. The *Wasserstein-p distance* between $\mu, \nu \in \mathcal{P}_p(X)$ is then

$$\mathsf{W}_p(\mu, \nu) \stackrel{\text{def}}{=} \inf_{\pi \in \Pi(\mu,\nu)} \left(\iint_{X \times X} d_X(x, y)^p \mathrm{d}\pi(x, y) \right)^{\frac{1}{p}},$$

where $\Pi(\mu, \nu)$ is the set of all couplings (matchings) π between (μ, ν) on X, i.e., probability measures π on $X \times X$ such that $\mu(A) = \pi(A \times X)$ and $\nu(A) = \pi(X \times A)$ for every $A \subseteq X$. W_p on $\mathcal{P}_p(X)$ is a metric whenever $p \geqslant 1$. Here we consider the classic setting of X being $\mathbb{R}^k$, for $k \geqslant 2$, endowed with the standard Euclidean distance.

In computer science, the transportation metrics on $\mathbb{R}^k$ play an important role in computer vision [58, 46, 21, 22, 28, 42, 38, 33], machine learning [20], information retrieval [45], and mechanism design [16], among others. For example, an image can be represented as a set of pixels in a color space $\mathbb{R}^3$; the transportation cost between such sets yields an accurate measure of dissimilarity between color characteristics of the images [47, 25].

These applications motivated a lot of research into the *computational* properties of transportation metrics. In particular, typical problems are to develop efficient algorithms for: computing the distance between two pointsets (finitely-supported measures), nearest neighbor search under these metrics, as well as problems in the streaming and sketching context.

So far, most of the rigorous algorithmic results have been developed for the W_1 metric, often refered to as the Earth Mover Distance (EMD). There is a long line of work on approximation algorithms for computing EMD between two pointsets in $\mathbb{R}^k$ [55, 2, 56, 1, 24, 49], culminating in a near-linear time algorithm achieving a $(1 + \varepsilon)$-approximation [50, 3, 7]. Nearest neighbor search algorithms all proceed via either embedding EMD into L_1 or sketching. Understanding the embeddability of EMD over $\mathbb{R}^k$ into L_1 is a well-known open problem [30], and the best distortion is currently known [14, 25, 26, 37, 5] to be between $O(k \log n)$ and $\Omega(k + \sqrt{\log n})$ for pointsets in $[n]^k = \{1, 2, \dots n\}^k$. Similarly, designing sketching algorithms for EMD over $\mathbb{R}^k$ is also a well-known open problem [40, 41]. Some of the sketching bounds for W_1 follow from the aforementioned L_1 embeddings, and some others are proved directly [4, 6].

Yet, in a number of applications the Wasserstein-2 distance W_2 is a *more natural* distance than Wasserstein-1 (EMD), and indeed other communities have paid more attention to W_2 [53]. Specifically, W_2 (a.k.a., *root-mean square bipartite matching distance*) corresponds to the "ℓ_2 error" between two pointsets, in contrast to the "ℓ_1 error" measured by W_1; as such they have better regularity properties and also have a differential interpretation [53]. See [34, 18] for a further discussion of why using W_2 gives results of a better quality than W_1. W_2 is used in graphics [51, 52, 54, 53], for shape interpolation [12], for barycenter computation [15, 11], shape reconstruction [19], blue noise generation [18], triangulations [34], among others.

Surprisingly, the algorithmic results for W_2 have been much more elusive. The best algorithms for computing W_2 distance between two pointsets follow from [43, 3], who obtain $\tilde{O}(n^2)$ time for exact and $\tilde{O}(n^{3/2})$ for approximate computation (in contrast to the near-linear

time algorithms for W_1). Beyond these results, there are no known non-trivial algorithms for embedding, nearest neighbor search, or sketching for W_2! This discrepancy raises the question of why there has been such a dire lack of progress on algorithms for W_2.

Here we address this question by proving the first explicit lower bounds for W_2 over $\mathbb{R}^3$, establishing that it is a very rich space that cannot be represented faithfully even with weak guarantees in a large class of normed spaces (that includes all L_q spaces for finite q, and much more). In particular, focusing on W_2 on measures over $\mathbb{R}^3$ supported on at most n points, we show that $\Omega(\sqrt{\log n})$ distortion is required for either: 1) embedding of W_2 into L_1, and 2) constant-size sketching. To contrast these results to those known for W_1 over the same set of measures, while W_1 has a similar non-embeddability into L_1 [37], it does not translate into *sketching* lower bounds. In fact, it was only recently established [6] that the approximation for sketching W_1 must be super-constant (without giving an explicit bound). Besides stronger sketching lower bounds, our results for W_2 are stronger than any known W_1 non-embeddability results since they apply to a larger class of Banach space targets (nontrivial type), and also rule out embeddings that are much weaker than bi-Lipschitz, like coarse embeddings. Finally, our results also apply to W_p space for $p \in (1,2)$, yielding a $\Omega((\log n)^{1/p})$ distortion lower bound, which is asymptotically stronger than the distortion lower bound known for embedding W_1 into L_1.

Our results apply to measures over $\mathbb{R}^3$ only, and the validity of analogous results for measures over $\mathbb{R}^2$ remains an open question. The only progress has been obtained in the forthcoming work [8], where the authors establish the first lower bound for embedding $\mathsf{W}_2(\mathbb{R}^2)$ into L_1, showing that the distortion goes to infinity (without an explicit bound). However, [8] does not yield the full strength of our results in terms of ruling out embeddings into spaces with nontrivial type, as well as, say, coarse embeddings.

1.1 Main Results

We now present our results on non-existence of good embedding and sketching methods for W_2 over $\mathbb{R}^3$. We then show that these results follow from a more general principle: that W_2 over $\mathbb{R}^3$ is snowflake-universal, and hence, say, we can embed the square-root of a shortest path metric on an expander graph into it with distortion arbitrarily close to 1. Our results apply to all W_p for $p > 1$, but not to W_1.

Non-embeddability results. We now introduce the standard notion of embeddings.

▶ **Definition 1.** Fix two metric spaces (X, d_X) and (Y, d_Y), and $D \in [1, \infty]$. A mapping $f : X \to Y$ is an embedding with distortion at most D if there exists $s \in (0, \infty)$ such that every $x, y \in X$ satisfy $s \cdot d_X(x, y) \leqslant d_Y(f(x), f(y)) \leqslant Ds \cdot d_X(x, y)$. The infimum over those $D \in [1, \infty]$ for which this holds true is called the distortion of f and is denoted $\mathbf{dist}(f)$. If there exists a mapping $f : X \to Y$ with distortion at most D then we say that (X, d_X) embeds with distortion D into (Y, d_Y). The infimum of $\mathbf{dist}(f)$ over all $f : X \to Y$ is denoted $c_{(Y,d_Y)}(X, d_X)$, or $c_Y(X)$ if the metrics are clear from the context.

We prove the following theorem.

▶ **Theorem 2.** *For any fixed $p \in (1, \infty)$ and $n \in \mathbb{N}$, consider the metric space X consisting of all the measures on $\mathbb{R}^3$ that are supported on at most n points, equipped with the W_p metric. Then any embedding of X into L_1 must incur distortion $\Omega(((p-1)\log n)^{1/p})$.*

Theorem 2 implies a $\Omega(\sqrt{\log n})$ approximation for any algorithmic approach proceeding via embedding W_2 over measures on $\mathbb{R}^3$ whose support is of size at most n into L_1. While

embedding into L_1 is a common algorithmic technique for high-dimensional metric spaces, it is not the only one. In particular, despite non-embeddability into L_1, a metric could admit a better embedding into, say, $L_{1/2}$, which would imply efficient sketches and nearest neighbor search algorithms. We rule out such weaker embeddings as well.

In fact, our work actually yields impossibility results that are much stronger than the bi-Lipschitz nonembeddability statement that corresponds to Theorem 2. Our most general results are contained in the full version of this paper, but here is one illustrative example. Let X be either L_1 or a Banach space of nontrivial type.[1] Then for $p \in (1, \infty)$ there do not exist any nondecreasing functions $\alpha, \beta : [0, \infty) \to [0, \infty)$ with $\lim_{t\to\infty} \alpha(t) = \infty$ for which there is a mapping $f : \mathcal{P}_p(\mathbb{R}^3) \to X$ that satisfies

$$\forall\, \mu, \nu \in \mathcal{P}_p(\mathbb{R}^3), \qquad \alpha(\mathsf{W}_p(\mu, \nu)) \leqslant \|f(\mu) - f(\nu)\|_X \leqslant \beta(\mathsf{W}_p(\mu, \nu)).$$

Theorem 2 corresponds to the special case when the function α, β are linear and X is L_1. In common metric geometry jargon, the above statement asserts that $\mathcal{P}_p(\mathbb{R}^3)$ fails to admit a coarse embedding into any normed space of nontrivial type.

Sketching. We can also state our results using the language of the sketching algorithms. The notion of sketching is defined as follows [48].

▶ **Definition 3.** Fix a metric (X, d_X), and approximation $D \geqslant 1$. We say (X, d_X) has sketching complexity $s \geqslant 1$ if, for any threshold $r > 0$, there exists a distribution over sketching maps $\mathrm{sk} : X \to \{0,1\}^s$ and reconstruction algorithms $R : \{0,1\}^s \times \{0,1\}^s \to \{\text{close}, \text{far}\}$, satisfying the following. For any $x, y \in X$, with at least $2/3$ probability of success:

- if $d_X(x, y) \leqslant r$, then $R(\mathrm{sk}(x), \mathrm{sk}(y)) = \text{close}$;
- if $d_X(x, y) > Dr$, then $R(\mathrm{sk}(x), \mathrm{sk}(y)) = \text{far}$.

We are now ready to state our sketching lower bound for W_p for $p > 1$.

▶ **Theorem 4.** *Fix $p \in (1, \infty)$ and let $n, s \in \mathbb{N}$. Consider the metric space X consisting of all the measures on $\mathbb{R}^3$ that are supported on at most n points, equipped with the W_p metric. Then any sketching algorithm for X with sketching complexity s must have an approximation guarantee of $D = \Omega\left(\left(\frac{(p-1)\log n}{s}\right)^{1/p}\right)$.*

We note that, for comparison, standard ℓ_1, ℓ_2 metrics have constant sketching complexity [27, 48, 9]. Also, for W_1 over $\mathbb{R}^3$ (or $\mathbb{R}^2$), the only known lower bound is that $Ds = \omega(1)$, shown recently in [6], based on [37].

Snowflake universality. Our results follow from a more general phenomenon, captured by the following theorem.

▶ **Theorem 5.** *If $p \in (1, \infty)$ then for every finite metric space (X, d_X) we have*

$$c_{(\mathcal{P}_p(\mathbb{R}^3), \mathsf{W}_p)}\left(X, d_X^{\frac{1}{p}}\right) = 1.$$

[1] The correct class of Banach spaces here could even be all those Banach spaces that do not contain *every* finite metric space with distortion arbitrarily close to 1, but currently this stronger version of the ensuing statement holds true conditionally on a well-known open question in metric geometry; see the full version of this paper for more details.

For a metric space (X, d_X) and $\theta \in (0,1]$, the metric space (X, d_X^θ) is commonly called the *θ-snowflake* of (X, d_X); see e.g. [17]. Thus Theorem 5 asserts that the θ-snowflake of any finite metric space (X, d_X) embeds with distortion $1+\varepsilon$ into $\mathcal{P}_p(\mathbb{R}^3)$ for every $\varepsilon \in (0,\infty)$ and $\theta \in (0, 1/p]$.[2] Our techniques fall short of proving a longstanding conjecture of Bourgain [13], who asked whether $(\mathcal{P}_1(\mathbb{R}^2), \mathsf{W}_1)$ is not universal (i.e., does not contain all finite metrics).[3] Bourgain proved in [13] that $(\mathcal{P}_1(\ell_1), \mathsf{W}_1)$ is universal (despite the fact that ℓ_1 is not universal), but it remains an intriguing open question to determine whether or not $(\mathcal{P}_1(\mathbb{R}^k), \mathsf{W}_1)$ is universal for any finite $k \in \mathbb{N}$, the case $k = 2$ being most challenging.

Theorem 6 below implies that Theorem 5 is sharp if $p \in (1,2]$, and yields a nontrivial, though probably non-sharp, restriction on the embeddability of snowflakes into $\mathcal{P}_p(\mathbb{R}^3)$ also for $p \in (2,\infty)$.

▶ **Theorem 6.** *For arbitrarily large $n \in \mathbb{N}$ there exists an n-point metric space (X_n, d_{X_n}) such that for every $\alpha \in (0,1]$ we have*

$$c_{(\mathcal{P}_p(\mathbb{R}^3), \mathsf{W}_p)}(X_n, d_{X_n}^\alpha) \gtrsim \begin{cases} (\log n)^{\alpha - \frac{1}{p}} & \text{if } p \in (1,2], \\ (\log n)^{\alpha + \frac{1}{p} - 1} & \text{if } p \in (2,\infty). \end{cases}$$

Here, and in what follows, we use standard asymptotic notation, i.e., for $a, b \in [0,\infty)$ the notation $a \gtrsim b$ (respectively $a \lesssim b$) stands for $a \geqslant cb$ (respectively $a \leqslant cb$) for some universal constant $c \in (0,\infty)$. The notation $a \asymp b$ stands for $(a \lesssim b) \wedge (b \lesssim a)$.

The rest of the paper is organized as follows. We give the proof of Theorem 5 in Section 2, and its consequences, Theorem 2 and 4, in Section 2.1. We then present some future research directions suggested by our results in Section 3. We defer the proof of Theorem 6 to the full version.

2 Proof of Theorem 5

To establish the theorem, we will construct an explicit embedding of an n-point metric into $\mathsf{W}_2(\mathbb{R}^3)$. In what follows fix $n \in \mathbb{N}$ and an n-point metric space (X, d_X).

We start by presenting the intuition behind the construction. In particular, let us demonstrate a fundamental difference between W_1 and W_p for $p > 1$ for a simple transportation instance. We will exploit this construction in our embedding. Fix a positive integer k, and consider the optimal transport between the sets $A = \{0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}\}$ and $B = \{\frac{1}{k}, \frac{2}{k}, \ldots, 1\}$. While under the W_1 metric the optimal cost is simply 1, under W_p the optimal transport would send every $x \in A$ to $x + \frac{1}{k} \in B$, which incurs a cost of $\left(\sum_{i=1}^k \left(\frac{1}{k}\right)^p\right)^{1/p} = k^{1/p-1} \xrightarrow[k\to\infty]{} 0$. Note that for any $0 \leqslant \varepsilon < 1$, we can increase the transport cost to ε by introducing a "gap" of size εk. E.g., for some i, define $A = \{0, \frac{1}{k}, \ldots, \frac{i}{k}, \frac{i+\varepsilon k}{k}, \frac{i+\varepsilon k+1}{k}, \ldots, \frac{k-1}{k}\}$ and $B = A \setminus \{0\} \cup \{1\}$. Then the optimal transport cost under W_p would be

$$\left(\left(\frac{\varepsilon k}{k}\right)^p + \sum_{i=1}^{k-\varepsilon k}\left(\frac{1}{k}\right)^p\right)^{1/p} \xrightarrow[k\to\infty]{} \varepsilon\,.$$

[2] Formally, Theorem 5 makes this assertion when $\theta = 1/p$, but for general $\theta \in (0, 1/p]$ one can then apply Theorem 5 to the metric space $(X, d_X^{\theta p})$ to deduce the seemingly more general statement.

[3] Bourgain actually formulated this question as asking whether a certain Banach space (namely, the dual of the Lipschitz functions on the square $[0,1]^2$) has finite Rademacher cotype, but this is equivalent to the above formulation.

We shall use the fact that any graph, in particular the complete graph, can be realized in $\mathbb{R}^3$, so that if every edge is represented by a wire, there are no wire crossings (except at vertices). Imagine that each wire is replaced by a set of points with distances $1/k$ between neighboring points. We then introduce a gap of length proportional to $d_X(u,v)^{1/p}$ on the wire connecting u and v. The embedding of $u \in X$ will be into a uniform measure over the point realizing u, and all the points in all the wires. Then the transport from u to v must move the mass at u to the mass of v. By the simple example above, this can be done at cost proportional to $d_X(u,v)^{1/p}$, when k is sufficiently large. The trickier part is showing no better transport exist. To this end, we require that all the wires are sufficiently far apart, so any transport plan that does not move along the wires will have a huge cost. Finally, the triangle inequality ensures that the cost of a plan using the wires between the points $u = u_0, u_1, \ldots, u_q = v$ is at least $d_X(u,v)^{1/p}$ (this is the reason why we make the gaps proportional to the p-th roots).

We now proceed with the formal proof of the theorem. Write $X = \{x_1, \ldots, x_n\}$ and fix $\phi : \{1,\ldots,n\} \times \{1,\ldots,n\} \to \{1,\ldots,n^2\}$ to be an arbitrary bijection between $\{1,\ldots,n\} \times \{1,\ldots,n\}$ and $\{1,\ldots,n^2\}$. Below it will be convenient to use the following notation.

$$m \stackrel{\text{def}}{=} \min_{\substack{x,y\in X\\ x\neq y}} d_X(x,y)^{\frac{1}{p}} \qquad \text{and} \qquad M \stackrel{\text{def}}{=} \max_{x,y\in X} d_X(x,y)^{\frac{1}{p}}. \tag{1}$$

Fix $K \in \mathbb{N}$. Denoting the standard basis of $\mathbb{R}^3$ by $e_1 = (1,0,0)$, $e_2 = (0,1,0)$, $e_3 = (0,0,1)$, for every $i,j \in \{1,\ldots,n\}$ with $i < j$ define five families of points in $\mathbb{R}^3$ by setting for $s \in \{0,\ldots,K\}$,

$$Q^1_s(i,j) \stackrel{\text{def}}{=} \frac{Mi}{m}e_1 + \frac{M\phi(i,j)s}{mK}e_2, \tag{2}$$

$$Q^2_s(i,j) \stackrel{\text{def}}{=} \frac{Mi}{m}e_1 + \frac{M\phi(i,j)}{m}e_2 + \frac{Ms}{mK}e_3, \tag{3}$$

$$Q^3_s(i,j) \stackrel{\text{def}}{=} \frac{M(s(j-i)+Ki) + (K-s)d_X(x_i,x_j)^{\frac{1}{p}}}{mK}e_1 + \frac{M\phi(i,j)}{m}e_2 + \frac{M}{m}e_3, \tag{4}$$

$$Q^4_s(i,j) \stackrel{\text{def}}{=} \frac{Mj}{m}e_1 + \frac{M\phi(i,j)}{m}e_2 + \frac{M(K-s)}{mK}e_3, \tag{5}$$

$$Q^5_s(i,j) \stackrel{\text{def}}{=} \frac{Mj}{m}e_1 + \frac{M(K-s)\phi(i,j)}{mK}e_2. \tag{6}$$

Then $Q^1_K(i,j) = Q^2_0(i,j)$, $Q^3_K(i,j) = Q^4_0(i,j)$ and $Q^4_K(i,j) = Q^5_0(i,j)$, so the total number of points thus obtained equals $5(K+1) - 3 = 5K + 2$.

Define $\mathcal{B} \subseteq \mathbb{R}^3$ by setting

$$\mathcal{B} \stackrel{\text{def}}{=} \bigcup_{\substack{i,j\in\{1,\ldots,n\}\\ i<j}} \mathcal{B}_{ij}, \tag{7}$$

where for every $i,j \in \{1,\ldots,n\}$ with $i < j$ we write

$$\mathcal{B}_{ij} \stackrel{\text{def}}{=} \bigcup_{s=0}^{K} \left\{Q^1_s(i,j), Q^2_s(i,j), Q^3_s(i,j), Q^4_s(i,j), Q^5_s(i,j)\right\}. \tag{8}$$

Hence $|\mathcal{B}_{ij}| = 5K + 2$. We also define $\mathcal{C} \subseteq \mathbb{R}^3$ by

$$\mathcal{C} \stackrel{\text{def}}{=} \mathcal{B} \setminus \left\{\frac{Mi}{m}e_1 :\ i \in \{1,\ldots,n\}\right\}. \tag{9}$$

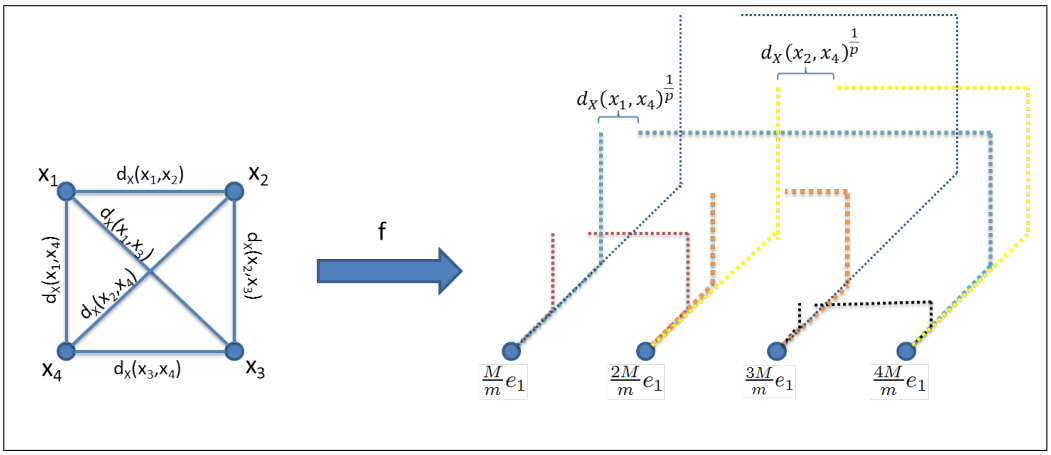


Figure 1 A schematic depiction of the embedding $f : X \to \mathcal{P}_p(\mathbb{R}^3)$ for a four-point metric space $(X, d_X) = (\{x_1, x_2, x_3, x_4\}, d_X)$. Here the x-axis is the horizontal direction, the z-axis is the vertical direction and the y-axis is perpendicular to the page plane. Recall that m and M are defined in (1).

Note that by (2) we have $(Mi/m)e_1 = Q_0^1(i,j)$ if $i, j \in \{1, \ldots, n\}$ satisfy $i < j$, and by (6) we have $(Mi/m)e_1 = Q_K^5(\ell, i)$ if $\ell, i \in \{1, \ldots, n\}$ satisfy $\ell < i$. Thus $\mathcal{C}$ corresponds to removing from $\mathcal{B}$ those points that lie on the x-axis. In what follows, we denote $N = |\mathcal{C}| + 1$. Finally, for every $i \in \{1, \ldots, n\}$ we define $\mathcal{C}_i \subseteq \mathbb{R}^3$ by

$$\mathcal{C}_i \stackrel{\text{def}}{=} \mathcal{C} \cup \left\{ \frac{Mi}{m} e_1 \right\}. \tag{10}$$

Hence $|\mathcal{C}_i| = N$. Our embedding $f : X \to \mathcal{P}_p(\mathbb{R}^3)$ will be given by

$$\forall\, j \in \{1, \ldots, n\}, \qquad f(x_j) \stackrel{\text{def}}{=} \frac{1}{N} \sum_{u \in \mathcal{C}_j} \delta_u, \tag{11}$$

where, as usual, δ_u is the point mass at u. Thus $f(x_j)$ is the uniform probability measure over $\mathcal{C}_j$. A schematic depiction of the above construction appears in Figure 1 below.

Lemma 7 below estimates the distortion of f, proving Theorem 5.

▶ **Lemma 7.** *Fix $\varepsilon \in (0,1)$ and $p \in (1, \infty)$. Let $f : X \to \mathcal{P}_p(\mathbb{R}^3)$ be the mapping appearing in (11), considered as a mapping from the snowflaked metric space $(X, d_X^{1/p})$ to the metric space $(\mathcal{P}_p(\mathbb{R}^3), \mathsf{W}_p)$. Then, recalling the definitions of m and M in (1), we have*

$$K \geqslant \left(\frac{5M^p n^{2p}}{pm^p \varepsilon} \right)^{\frac{1}{p-1}} \implies \mathbf{dist}(f) \leqslant 1 + \varepsilon. \tag{12}$$

Proof. We shall show that under the assumption on K that appears in (12) we have

$$\forall\, i, j \in \{1, \ldots, n\}, \qquad \left(\frac{d_X(x_i, x_j)}{m^p N} \right)^{\frac{1}{p}} \leqslant \mathsf{W}_p(f(x_i), f(x_j)) \leqslant (1+\varepsilon) \left(\frac{d_X(x_i, x_j)}{m^p N} \right)^{\frac{1}{p}}, \tag{13}$$

where we recall that we defined N to be equal to $|\mathcal{C}| + 1$ for $\mathcal{C}$ given in (9). Clearly (13) implies that $\mathbf{dist}(f) \leqslant 1 + \varepsilon$, as required.

To prove the right hand inequality in (13), suppose that $i, j \in \{1, \ldots, n\}$ satisfy $i < j$ and consider the coupling $\pi \in \Pi(f(x_i), f(x_j))$ given by

$$\pi \stackrel{\text{def}}{=} \frac{1}{N} \left(\sum_{t=1}^{5} \sum_{s=0}^{K-1} \delta_{\left(Q_s^t(i,j), Q_{s+1}^t(i,j)\right)} + \delta_{\left(Q_K^2(i,j), Q_0^3(i,j)\right)} + \sum_{u \in \mathcal{C} \setminus \mathcal{B}_{ij}} \delta_{(u,u)} \right), \tag{14}$$

where for (14) recall (8) and (9). The meaning of (14) is simple: the supports of $f(x_i)$ and $f(x_j)$ equal $\mathcal{C}_i$ and $\mathcal{C}_j$, respectively, where we recall (10). Note that $\mathcal{C}_i \setminus \mathcal{C}_j = \{Q_0^1(i,j)\}$ and $\mathcal{C}_j \setminus \mathcal{C}_i = \{Q_K^5(i,j)\}$, where we recall (2) and (6). So, the coupling π in (14) corresponds to shifting the points in $\mathcal{B}_{ij}$ from the support of $f(x_i)$ to the support of $f(x_j)$ while keeping the points in $\mathcal{C} \setminus \mathcal{B}_{ij}$ unchanged.

Now, recalling the definitions (2), (3), (4), (5) and (6),

$$\begin{aligned}\mathsf{W}_p(f(x_i),f(x_j))^p &\leqslant \iint_{\mathbb{R}^3\times\mathbb{R}^3} \|x-y\|_2^p \mathrm{d}\pi(x,y)\\ &= \frac{1}{N}\sum_{t=1}^{5}\sum_{s=0}^{K-1} \left\|Q_s^t(i,j)-Q_{s+1}^t(i,j)\right\|_2^p + \frac{\|Q_K^2(i,j)-Q_0^3(i,j)\|_2^p}{N}. \end{aligned} \tag{15}$$

Note that if $s \in \{0,\ldots,K-1\}$ then by (2), (3), (5), (6) we have

$$\begin{aligned} &t \in \{1,5\} \implies \left\|Q_s^t(i,j)-Q_{s+1}^t(i,j)\right\|_2 = \frac{M\phi(i,j)}{mK} \leqslant \frac{Mn^2}{mK},\\ &t \in \{2,4\} \implies \left\|Q_s^t(i,j)-Q_{s+1}^t(i,j)\right\|_2 = \frac{M}{mK}. \end{aligned} \tag{16}$$

Also, by (3) and (4) we have

$$\left\|Q_K^2(i,j)-Q_0^3(i,j)\right\|_2 = \frac{d_X(x_i,x_j)^{\frac{1}{p}}}{m}. \tag{17}$$

Finally, by (4) for every $s \in \{0,\ldots,K-1\}$ we have

$$\left\|Q_s^3(i,j)-Q_{s+1}^3(i,j)\right\|_2 = \frac{M(j-i)}{mK} - \frac{d_X(x_i,x_j)^{\frac{1}{p}}}{mK} \leqslant \frac{Mn}{mK}, \tag{18}$$

where we used the fact that $M(j-i)-d_X(x_i,x_j)^{1/p} \geqslant 0$, which holds true by the definition of M in (1) because $j-i \geqslant 1$. A substitution of (16), (17) and (18) into (15) yields the estimate

$$\begin{aligned}\mathsf{W}_p(f(x_i),f(x_j))^p &\leqslant \frac{d_X(x_i,x_j)}{m^pN} + \frac{5K}{N}\left(\frac{Mn^2}{mK}\right)^p\\ &= \left(1+\frac{5M^pn^{2p}}{K^{p-1}d_X(x_i,x_j)}\right)\frac{d_X(x_i,x_j)}{m^pN} \leqslant (1+p\varepsilon)\frac{d_X(x_i,x_j)}{m^pN},\end{aligned}$$

where we used the fact that by the definition of m in (1) we have $m^p \leqslant d_X(x_i,x_j)$, and the lower bound on K that is assumed in (12). This implies the right hand inequality in (13) because $1+p\varepsilon \leqslant (1+\varepsilon)^p$.

Passing now to the proof of the left hand inequality in (13), we need to prove that for every $i,j \in \{1,\ldots,n\}$ with $i<j$ we have

$$\forall\, \pi \in \Pi(f(x_i),f(x_j)), \qquad \iint_{\mathbb{R}^3\times\mathbb{R}^3} \|x-y\|_2^p \mathrm{d}\pi(x,y) \geqslant \frac{d_X(x_i,x_j)}{m^pN}. \tag{19}$$

Note that we still did not use the triangle inequality for d_X, but this will be used in the proof of (19). Also, the reason why we are dealing with $\mathcal{P}_p(\mathbb{R}^3)$ rather than $\mathcal{P}_p(\mathbb{R}^2)$ will become clear in the ensuing argument.

Recall that the measures $f(x_i)$ and $f(x_j)$ are uniformly distributed over sets of the same size, and their supports $\mathcal{C}_i$ and $\mathcal{C}_j$ (respectively) satisfy $\mathcal{C}_i \triangle \mathcal{C}_j = \{(Mi/m)e_1, (Mj/m)e_1\}$.

Since the set of all doubly stochastic matrices is the convex hull of the permutation matrices, and every permutation is a product of disjoint cycles, it follows that it suffices to establish the validity of (19) when $\pi = \frac{1}{N}\sum_{\ell=1}^{L}\delta_{(u_{\ell-1},u_\ell)}$ for some $L \in \{1,\ldots,n\}$ and $u_1,\ldots u_{L-1} \in \mathcal{C}$, where we set $u_0 = (Mi/m)e_1$ and $u_L = (Mj/m)e_1$. With this notation, our goal is to show that

$$\frac{1}{N}\sum_{\ell=1}^{L}\|u_\ell - u_{\ell-1}\|_2^p \geqslant \frac{d_X(x_i,x_j)}{m^p N}. \tag{20}$$

For every $a \in \{1,\ldots,n\}$ define $\mathcal{S}_a \subseteq \mathbb{R}^3$ by $\mathcal{S}_a \stackrel{\text{def}}{=} \mathcal{S}_a^1 \cup \mathcal{S}_a^2$, where

$$\mathcal{S}_a^1 \stackrel{\text{def}}{=} \bigcup_{b=a+1}^{n}\bigcup_{s=0}^{K}\left\{Q_s^1(a,b), Q_s^2(a,b)\right\}, \tag{21}$$

and

$$\mathcal{S}_a^2 \stackrel{\text{def}}{=} \bigcup_{c=1}^{a-1}\bigcup_{s=0}^{K}\left\{Q_s^3(c,a), Q_s^4(c,a), Q_s^5(c,a)\right\}. \tag{22}$$

Thus, recalling (7), the sets $\mathcal{S}_1,\ldots,\mathcal{S}_n$ form a partition of $\mathcal{B}$ and $a \in \mathcal{S}_a$ for every $a \in \{1,\ldots,n\}$. For every $\ell \in \{0,\ldots,L\}$ let $a(\ell)$ be the unique element of $\{1,\ldots,n\}$ for which $u_\ell \in \mathcal{S}_{a(\ell)}$. Then $a(0) = i$ and $a(L) = j$. The left hand side of (20) can be bounded from below as follows

$$\frac{1}{N}\sum_{\ell=1}^{L}\|u_\ell - u_{\ell-1}\|_2^p \geqslant \frac{1}{N}\sum_{\ell=1}^{L}\min_{\substack{u\in\mathcal{S}_{a(\ell-1)}\\ v\in\mathcal{S}_{a(\ell)}}}\|u-v\|_2^p. \tag{23}$$

We shall show that

$$\forall\, a,b \in \{1,\ldots,n\},\ \forall (u,v) \in \mathcal{S}_a \times \mathcal{S}_b, \qquad \|u-v\|_2^p \geqslant \frac{d_X(x_a,x_b)}{m^p}. \tag{24}$$

The validity of (24) implies the required estimate (20) because, by (23), it follows from (24) and the triangle inequality for d_X that

$$\frac{1}{N}\sum_{\ell=1}^{L}\|u_\ell - u_{\ell-1}\|_2^p \geqslant \frac{1}{N}\sum_{\ell=1}^{L}\frac{d_X\left(x_{a(\ell-1)},x_{a(\ell)}\right)}{m^p} \geqslant \frac{d_X(x_i,x_j)}{m^p N}.$$

It remains to justify (24). Suppose that $a,b \in \{1,\ldots,n\}$ satisfy $a < b$ and $(u,v) \in \mathcal{S}_a \times \mathcal{S}_b$. Write $u = Q_s^t(c,d)$ and $v = Q_\sigma^\tau(\gamma,\delta)$ for some $s,\sigma \in \{0,\ldots,K\}$, $t,\tau \in \{1,\ldots,5\}$ and $c,d,\gamma,\delta \in \{1,\ldots,n\}$.

We shall check below, via a direct case analysis, that the absolute value of one of the three coordinates of $u-v$ is either at least M/m or at least $d_X(x_a,x_b)^{1/p}/m$. Since by the definition of M in (1) we have $M \geqslant d_X(x_a,x_b)^{1/p}$, this assertion will imply (24).

Suppose first that $t,\tau \in \{1,2,4,5\}$. By comparing (21), (22) with (2), (3), (4), (5) we see that $\langle u,e_1\rangle = Ma/m$ and $\langle v,e_1\rangle = Mb/m$. Since $b-a \geqslant 1$, this implies that $\langle u-v,e_1\rangle \geqslant M/m$, as required.

If $t = \tau = 3$ then by (22) we necessarily have $d = a$ and $\delta = b$. Hence $(c,d) \neq (\gamma,\delta)$ and therefore $|\phi(c,d) - \phi(\gamma,\delta)| \geqslant 1$, since ϕ is a bijection between $\{1,\ldots,n\}\times\{1,\ldots,n\}$ and $\{1,\ldots,n^2\}$. By (4) we therefore have $|\langle u-v,e_2\rangle| \geqslant M/m$, as required.

It remains to treat the case $t \neq \tau$ and $3 \in \{t, \tau\}$. If $\{t, \tau\} \subseteq \{1, 3, 5\}$ then by contrasting (4) with (2) and (6) we see that the third coordinate of one of the vectors u, v vanishes while the third coordinate of the other vector equals M/m. Therefore $|\langle u - v, e_3\rangle| \geqslant M/m$, as required. The only remaining case is $\{t, \tau\} \subseteq \{2, 3, 4\}$. In this case $|\langle u-v, e_2\rangle| = M|\phi(c, d)-\phi(\gamma, \delta)|/m$, by (4), (3), (5). So, if $(c, d) \neq (\gamma, \delta)$ then $|\phi(c, d) - \phi(\gamma, \delta)| \geqslant 1$, and we are done. We may therefore assume that $c = \gamma$ and $d = \delta$. Observe that by (22) if $\{t, \tau\} = \{3, 4\}$ then $\{d, \delta\} = \{a, b\}$, which contradicts $d = \delta$. So, we also necessarily have $\{t, \tau\} = \{2, 3\}$, in which case, since $a < b$, by (21) and (22) we see that $c = \gamma = a$ and $d = \delta = b$. By interchanging the labels s and σ if necessary, we may assume that $u = Q^2_\sigma(a, b)$ and $v = Q^3_s(a, b)$. By (3) and (4) we therefore have

$$\begin{aligned}\langle v - u, e_1\rangle &= \frac{M(s(b-a)+Ka)}{mK} + \frac{(K-s)d_X(x_a,x_b)^{\frac{1}{p}}}{mK} - \frac{Ma}{m} \\ &= \frac{d_X(x_a,x_b)^{\frac{1}{p}}}{m} + \frac{sM(b-a) - sd_X(x_a,x_b)^{\frac{1}{p}}}{mK} \geqslant \frac{d_X(x_a,x_b)^{\frac{1}{p}}}{m},\end{aligned}$$

where we used the fact that by (1) we have $M \geqslant d_X(x_a, x_b)^{1/p}$, and that $b - a \geqslant 1$. This concludes the verification of the remaining case of (24), and hence the proof of Lemma 7 is complete. ◀

2.1 Implications: Theorems 2 and 4

Theorem 2 follows from the fact that the shortest path metric on an expander graph on N nodes has $\Omega(\log N)$ distortion lower bound for embedding it into L_1 [29]. Note that in the proof above we obtain measures supported on n points where $n \leqslant N^{O(1)} \cdot \left(\frac{5M^pN^{2p}}{pm^p}\right)^{\frac{1}{p-1}}$ for a $1 + \varepsilon = 2$ approximation. Hence, any embedding of W_p on $\mathbb{R}^3$ pointsets of size n into L_1 has a distortion lower bound of $\Omega((\log N)^{1/p}) = \Omega(((p-1)\log n)^{1/p})$.

Similarly, Theorem 4 follows by considering X to be the N-point subset of $(\mathcal{P}_1(\{0,1\}^{O(\log N)}), \mathsf{W}_1)$ introduced in [26, Section 3]. Any sketching algorithm for this metric X requires $\Omega(\frac{\log N}{s})$ approximation for sketching complexity s [5, Theorem 4.1]. Since we can embed X into the square of W_2 with constant distortion, we obtain a $\Omega\left(\left(\frac{(p-1)\log n}{s}\right)^{1/p}\right)$ approximation lower bound for any W_p sketch with sketching complexity s.

3 Future Directions

As discussed in the Introduction, it seems plausible that Theorem 5 and Theorem 6 are not sharp when $p \in (2, \infty)$. Specifically, we conjecture that there exist $D_p \in [1, \infty)$ such that for every finite metric space (X, d_X) we have

$$c_{\mathcal{P}_p(\mathbb{R}^3)}\left(X, \sqrt{d_X}\right) \leqslant D_p. \tag{25}$$

Perhaps (25) even holds true with $D_p = 1$. Since L_2 admits an isometric embedding into L_p (see e.g. [59]), the perceived analogy between Wasserstein p spaces and L_p spaces makes it natural to ask whether or not $(\mathcal{P}_2(\mathbb{R}^3), \mathsf{W}_2)$ admits a bi-Lipschitz embedding into $(\mathcal{P}_p(\mathbb{R}^3), \mathsf{W}_p)$. If the answer to this question were positive then (25) would hold true by virtue of the case $p = 2$ of Theorem 5. We also conjecture that the lower bound of Theorem 6 could be improved when $p > 2$ to state that for arbitrarily large $n \in \mathbb{N}$ there exists an n-point metric space (Y, d_Y) such that for every $\alpha \in (1/2, 1]$,

$$c_{(\mathcal{P}_p(\mathbb{R}^3), \mathsf{W}_p)}(Y, d_Y^\alpha) \gtrsim_p (\log n)^{\alpha - \frac{1}{2}}. \tag{26}$$

It was shown in [36] that L_p has Markov type 2 for every $p \in (2,\infty)$. We therefore ask whether or not $(\mathcal{P}_p(\mathbb{R}^3), \mathsf{W}_p)$ has Markov type 2 for every $p \in (2,\infty)$. A positive answer to this question would imply that the lower bound (26) is indeed achievable. For this purpose it would also suffice to show that for every $p \in (2,\infty)$ and $k \in \mathbb{N}$ we have

$$M_p((\mathcal{P}_p(\mathbb{R}^3), \mathsf{W}_p); 2^k) \lesssim_p 2^{k\left(\frac{1}{2}-\frac{1}{p}\right)}. \tag{27}$$

Proving (27) may be easier than proving that $M_2(\mathcal{P}_p(\mathbb{R}^3), \mathsf{W}_p) < \infty$, since the former involves arguing about the pth powers of Wasserstein p distances while the latter involves arguing about Wasserstein p distances squared. Note that $M_p(L_p; m) \lesssim \sqrt{p} m^{1/2-1/p}$ by [36] (see also [35, Theorem 4.3]), so the L_p-version of (27) is indeed valid.

Another natural direction to pursue concerns with the distortion of embedding finite metric spaces into Wasserstein spaces.

▶ **Question 1.** *Is it true that for $p \in (1,2]$ and $n \in \mathbb{N}$ every n-point metric space (X, d_X) satisfies*

$$c_{\mathcal{P}_p(\mathbb{R}^3)}(X) \lesssim_p (\log n)^{1-\frac{1}{p}}?$$

A positive answer to Question (1) would resolve the *metric cotype dichotomy problem* [31] (see the full version for more details). We believe that Question 1 is an especially intriguing challenge in embedding theory (for a concrete and natural target space) because a positive answer would require an interesting new construction, and a negative answer would require devising a new bi-Lipschitz invariant that would serve as an obstruction for embeddings into Wasserstein spaces.

Focusing for concreteness on the case $p = 2$, Question 1 asks whether $c_{\mathcal{P}_2(\mathbb{R}^3)}(X) \lesssim \sqrt{\log n}$ for every n-point metric space (X, d_X). Note that Theorem 5 implies that (X, d_X) embeds into $\mathcal{P}_2(X)$ with distortion at most the square root of the *aspect ratio* of (X, d_X), i.e.,

$$c_{(\mathcal{P}_2(\mathbb{R}^3), \mathsf{W}_2)}(X, d_X) \leqslant \sqrt{\frac{\operatorname{diam}(X, d_X)}{\min_{\substack{x,y\in X\\ x\neq y}} d_X(x,y)}}, \tag{28}$$

but we are asking here for the largest possible growth rate of the distortion of X into $\mathcal{P}_2(X)$ in terms of the cardinality of X. While for certain embedding results there are standard methods (see e.g. [10, 23, 32]) for replacing the dependence on the aspect ratio of a finite metric space by a dependence on its cardinality, these methods do not seem to apply to our embedding in (28). See the full version for further discussion.

References

1. Pankaj Agarwal and Kasturi Varadarajan. A near-linear constant-factor approximation for euclidean bipartite matching? In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG'04, pages 247–252, New York, NY, USA, 2004. ACM. doi:10.1145/997817.997856.
2. Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. *SIAM Journal on Computing*, 29(3):912–953, 2000.
3. Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC'14, pages 555–564, New York, NY, USA, 2014. ACM. doi:10.1145/2591796.2591844.

4 Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. Efficient sketches for Earth-Mover Distance, with applications. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2009.

5 Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 343–352, 2008. Previously ECCC Report TR07-048.

6 Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn. Sketching and embedding are equivalent for norms. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2015. Full version at `http://arxiv.org/abs/1411.2577`.

7 Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2014. Full version at `http://arxiv.org/abs/1401.0042`. `doi:10.1145/2591796.2591805`.

8 T. Austin and A. Naor. On the bi-Lipschitz structure of Wasserstein spaces. In preparation, 2016.

9 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004. Previously in FOCS'02.

10 Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science (Burlington, VT, 1996)*, pages 184–193. IEEE Comput. Soc. Press, Los Alamitos, CA, 1996. `doi:10.1109/SFCS.1996.548477`.

11 Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

12 Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. Displacement interpolation using lagrangian mass transport. *ACM Transactions on Graphics (TOG)*, 30(6):158, 2011.

13 J. Bourgain. The metrical interpretation of superreflexivity in Banach spaces. *Israel J. Math.*, 56(2):222–230, 1986. `doi:10.1007/BF02766125`.

14 Moses Charikar. Similarity estimation techniques from rounding. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 380–388, 2002.

15 Marco Cuturi and Arnaud Doucet. Fast computation of wasserstein barycenters. In *Proceedings of The 31st International Conference on Machine Learning*, pages 685–693, 2014.

16 Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. Mechanism design via optimal transport. In *Proc. of the 14th ACM Conf. on Electronic Commerce*, EC'13, pages 269–286, New York, NY, USA, 2013. ACM. `doi:10.1145/2482540.2482593`.

17 Guy David and Stephen Semmes. *Fractured fractals and broken dreams*, volume 7 of *Oxford Lecture Series in Mathematics and its Applications*. The Clarendon Press, Oxford University Press, New York, 1997. Self-similar geometry through metric and measure.

18 Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. Blue noise through optimal transport. *ACM Transactions on Graphics (TOG)*, 31(6):171, 2012.

19 Fernando De Goes, David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. An optimal transport approach to robust reconstruction and simplification of 2d shapes. *Computer Graphics Forum*, 30(5):1593–1602, 2011.

20 Norm Ferns, Pablo Samuel Castro, Doina Precup, and Prakash Panangaden. Methods for computing state similarity in markov decision processes. In *UAI'06, Proc. of the 22nd Conf. in Uncertainty in Artificial Intelligence, Cambridge, MA, USA, July 13-16, 2006*, 2006.

21 Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Beijing, China, October 2005.

22 Kristen Grauman and Trevor Darrell. Approximate correspondences in high dimensions. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2006.

23 Sariel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184 (electronic), 2006. doi:10.1137/S0097539704446281.

24 Piotr Indyk. A near linear time constant factor approximation for euclidean bichromatic matching (cost). In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2007.

25 Piotr Indyk and Nitin Thaper. Fast color image retrieval via embeddings. *Workshop on Statistical and Computational Theories of Vision (at ICCV)*, 2003.

26 Subhash Khot and Assaf Naor. Nonembeddability theorems via Fourier analysis. *Math. Ann.*, 334(4):821–852, 2006. doi:10.1007/s00208-005-0745-0.

27 Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000. Preliminary version appeared in STOC'98.

28 Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

29 N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

30 Jiří Matoušek and Assaf Naor. Open problems on embeddings of finite metric spaces, August 2011. Available at http://kam.mff.cuni.cz/~matousek/metrop.ps.

31 Manor Mendel and Assaf Naor. Metric cotype. *Ann. of Math. (2)*, 168(1):247–298, 2008. doi:10.4007/annals.2008.168.247.

32 Manor Mendel and Assaf Naor. Maximum gradient embeddings and monotone clustering. *Combinatorica*, 30(5):581–615, 2010. doi:10.1007/s00493-010-2302-z.

33 David M. Mount, Nathan S. Netanyahu, and San Ratanasanya. New approaches to robust, point-based image registration. In Jacqueline Le Moigne, Nathan S. Netanyahu, and Roger D. Eastman, editors, *Image Registration for Remote Sensing*, pages 179–199. Cambridge University Press, 2011. Cambridge Books Online. doi:10.1017/CBO9780511777684.009.

34 Patrick Mullen, Pooran Memari, Fernando de Goes, and Mathieu Desbrun. Hot: Hodge-optimized triangulations. *ACM Transactions on Graphics (TOG)*, 30(4):103, 2011.

35 Assaf Naor. Comparison of metric spectral gaps. *Anal. Geom. Metr. Spaces*, 2:1–52, 2014. doi:10.2478/agms-2014-0001.

36 Assaf Naor, Yuval Peres, Oded Schramm, and Scott Sheffield. Markov chains in smooth Banach spaces and Gromov-hyperbolic metric spaces. *Duke Math. J.*, 134(1):165–197, 2006. doi:10.1215/S0012-7094-06-13415-4.

37 Assaf Naor and Gideon Schechtman. Planar earthmover is not in L_1. *SIAM J. Comput. (SICOMP)*, 37(3):804–826, 2007. An extended abstract appeared in FOCS'06.

38 Kangyu Ni, Xavier Bresson, Tony F. Chan, and Selim Esedoglu. Local histogram based segmentation using the wasserstein distance. *International Journal of Computer Vision*, 84(1):97–111, 2009. doi:10.1007/s11263-009-0234-0.

39 Yann Ollivier, Herve Pajot, and Cedric Villani. *Optimal Transport, Theory and Applications*. Cambridge University Press, New York, NY, USA, 2014.

40 List of open problems in sublinear algorithms: Problem 7. http://sublinear.info/7.

41 List of open problems in sublinear algorithms: Problem 49. http://sublinear.info/49.

42 Ofir Pele and Michael Werman. Fast and robust earth mover's distances. In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 – October 4, 2009*, pages 460–467, 2009. doi:10.1109/ICCV.2009.5459199.

43 Jeff M. Phillips and Pankaj K. Agarwal. On bipartite matching under the RMS distance. In *Proceedings of the 18th Annual Canadian Conference on Computational Geometry, CCCG 2006, August 14-16, 2006, Queen's University, Ontario, Canada*, 2006. URL: http://www.cs.queensu.ca/cccg/papers/cccg37.pdf.

44 Svetlozar T. Rachev and Ludger Rüschendorf. *Mass transportation problems. Vol. I.* Probability and its Applications (New York). Springer-Verlag, New York, 1998. Theory.

45 Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *ICCV*, pages 59–66, 1998.

46 Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, November 2000. doi:10.1023/A:1026543900054.

47 Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

48 Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 360–369, 2002. doi:10.1145/509907.509963.

49 R. Sharathkumar and Pankaj K. Agarwal. Algorithms for the transportation problem in geometric settings. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 306–317, 2012.

50 R. Sharathkumar and Pankaj K. Agarwal. A near-linear time approximation algorithm for geometric bipartite matching. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 385–394, 2012.

51 Justin Solomon, Fernando de Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4):66:1–66:11, July 2015. In SIGGRAPH'15. doi:10.1145/2766963.

52 Justin Solomon, Leonidas Guibas, and Adrian Butscher. Dirichlet energy for analysis and synthesis of soft maps. *Computer Graphics Forum*, 32(5):197–206, 2013.

53 Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. Earth mover's distances on discrete surfaces. *ACM Trans. Graph.*, 33(4):67:1–67:12, July 2014. doi:10.1145/2601097.2601175.

54 Justin Solomon, Raif Rustamov, Leonidas Guibas, and Adrian Butscher. Wasserstein propagation for semi-supervised learning. In *Proceedings of The 31st International Conference on Machine Learning*, pages 306–314, 2014.

55 Pravin M. Vaidya. Geometry helps in matching. *SIAM Journal on Computing*, 18(6):1201–1225, 1989.

56 Kasturi R. Varadarajan and Pankaj K. Agarwal. Approximation algorithms for bipartite and non-bipartite matching in the plane. In *Proc. of the 10th annual ACM-SIAM Symp. on Discrete algorithms*, pages 805–814. SIAM, 1999.

57 Cédric Villani. *Topics in optimal transportation*, volume 58 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2003.

58 Michael Werman, Shmuel Peleg, and Azriel Rosenfeld. A distance metric for multidimensional histograms. *Computer Vision, Graphics, and Image Processing*, 32(3):328–336, 1985. doi:10.1016/0734-189X(85)90055-6.

59 P. Wojtaszczyk. *Banach spaces for analysts*, volume 25 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 1991. doi:10.1017/CBO9780511608735.

Simple Average-Case Lower Bounds for Approximate Near-Neighbor from Isoperimetric Inequalities

Yitong Yin*

State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China
yinyt@nju.edu.cn

Abstract

We prove an $\Omega(d/\log\frac{sw}{nd})$ lower bound for the average-case cell-probe complexity of deterministic or Las Vegas randomized algorithms solving approximate near-neighbor (ANN) problem in d-dimensional Hamming space in the cell-probe model with w-bit cells, using a table of size s. This lower bound matches the highest known worst-case cell-probe lower bounds for any static data structure problems.

This average-case cell-probe lower bound is proved in a general framework which relates the cell-probe complexity of ANN to isoperimetric inequalities in the underlying metric space. A tighter connection between ANN lower bounds and isoperimetric inequalities is established by a stronger richness lemma proved by cell-sampling techniques.

1998 ACM Subject Classification E.1 Data Structures

Keywords and phrases nearest neighbor search, approximate near-neighbor, cell-probe model, isoperimetric inequality

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.84

1 Introduction

The nearest neighbor search (NNS) problem is a fundamental problem in Computer Science. In this problem, a database $y = (y_1, y_2, \ldots, y_n)$ of n points from a metric space (X, dist) is preprocessed to a data structure, and at the query time given a query point x from the same metric space, we are asked to find the point y_i in the database which is closest to x according to the metric.

In this paper, we consider a decision and approximate version of NNS, the approximate near-neighbor (ANN) problem, where the algorithm is asked to distinguish between the two cases: (1) there is a point in the databases that is λ-close to the query point for some radius λ, or (2) all points in the database are $\gamma\lambda$-far away from the query point, where $\gamma \geq 1$ is the approximation ratio.

The complexity of nearest neighbor search has been extensively studied in the cell-probe model, a classic model for data structures. In this model, the database is encoded to a table consisting of memory cells. Upon each query, a cell-probing algorithm answers the query by making adaptive cell-probes to the table. The complexity of the problem is measured by the tradeoff between the time cost (in terms of number of cell-probes to answer a query) and the space cost (in terms of sizes of the table and cells). There is a substantial body of work on the cell-probe complexity of NNS for various metric space [6, 7, 5, 11, 8, 14, 3, 2, 16, 17, 12, 20].

* This work was done in part while the author was visiting the Simons Institute for the Theory of Computing. This work was partially supported by NSFC grants 61272081 and 61321491.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 84; pp. 84:1–84:13

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

It is widely believed that NNS suffers from the "curse of dimensionality" [10]: The problem may become intractable to solve when the dimension of the metric space becomes very high. Consider the most important example, d-dimensional Hamming space $\{0,1\}^d$ with $d \geq C \log n$ for a sufficiently large constant C. The conjecture is that NNS in this metric remains hard to solve when either approximation or randomization is allowed individually.

In a series of pioneering works [6, 5, 11, 14, 3], by a rectangle-based technique of asymmetric communication complexity known as the richness lemma [15], cell-probe lower bounds in form of $\Omega(d/\log s)$, where s stands for the number of cells in the table, were proved for deterministic approximate near-neighbor (due to Liu [14]) and randomized exact near-neighbor (due to Barkol and Rabani [5]). Such lower bound is the highest possible lower bound one can prove in the communication model. This fundamental barrier was overcome by an elegant self-reduction technique introduced in the seminal work of Pătraşcu and Thorup [18], in which the cell-probe lower bounds for deterministic ANN and randomized exact near-neighbor were improved to $\Omega(d/\log \frac{sw}{n})$, where w represents the number of bits in a cell. More recently, in a previous work of us [20], by applying the technique of Pătraşcu and Thorup to the certificates in data structures, the lower bound for deterministic ANN was further improved to $\Omega(d/\log \frac{sw}{nd})$. This last lower bound behaves differently for the polynomial space where $sw = \text{poly}(n)$, near-linear space where $sw = n \cdot \text{polylog}(n)$, and linear space where $sw = O(nd)$. In particular, the bound becomes $\Omega(d)$ when the space cost is strictly linear in the entropy of the database, i.e. when $sw = O(nd)$.

When both randomization and approximation are allowed, the complexity of NNS is substantially reduced. With polynomial-size tables, a $\Theta(\log\log d/\log\log\log d)$ tight bound was proved for randomized approximate NNS in d-dimensional Hamming space [7, 8]. If we only consider the decision version, the randomized ANN can be solved with $O(1)$ cell-probes on a table of polynomial size [8]. For tables of near-linear size, a technique called cell-sampling was introduced by Panigrahy *et al.* [16, 17] to prove $\Omega(\log n/\log \frac{sw}{n})$ lower bounds for randomized ANN. This was later extended to general asymmetric metrics [1].

Among these lower bounds, the randomized ANN lower bounds of Panigrahy *et al.* [16, 17] were proved explicitly for *average-case* cell-probe complexity. The significance of average-case complexity for NNS was discussed in their papers. A recent breakthrough in upper bounds [4] also attributes to solving the problem on a random database. Retrospectively, the randomized exact near-neighbor lower bounds due to the density version of richness lemma [6, 5, 11] also hold for random inputs. All these average-case lower bounds hold for Monte Carlo randomized algorithms with fixed worst-case cell-probe complexity. This leaves open an important case: the average-case cell-probe complexity for the deterministic or Las Vegas randomized algorithms for ANN, where the number of cell-probes may vary for different inputs.

1.1 Our contributions

We study the average-case cell-probe complexity of deterministic or Las Vegas randomized algorithms for the approximate near-neighbor (ANN) problem, where the number of cell-probes to answer a query may vary for different query-database pairs and the average is taken with respect to the distribution over input queries and databases.

For ANN in Hamming space $\{0,1\}^n$, the hard distribution over inputs is very natural: Every point y_i in the database $y = (y_1, y_2, \ldots, y_n)$ is sampled uniformly and independently from the Hamming space $\{0,1\}^d$, and the query point x is also a point sampled uniformly and independently from $\{0,1\}^d$. According to earlier average-case lower bounds [16, 17] and the recent data-dependent LSH algorithm [4], this input distribution seems to capture the

hardest case for nearest neighbor search and is also a central obstacle to overcome for efficient algorithms.

By a simple proof, we show the following lower bound for the average-case cell-probe complexity of ANN in Hamming space with this very natural input distribution.

▶ **Theorem 1.** *For $d \geq 32 \log n$ and $d < n^{o(1)}$, any deterministic or Las Vegas randomized algorithm solving (γ, λ)-approximate near-neighbor problem in d-dimensional Hamming space in the cell-probe model with w-bit cells for $w < n^{o(1)}$, using a table of size $s < 2^d$, must have expected cell-probe complexity $t = \Omega\left(\frac{d}{\gamma^2 \log \frac{sw\gamma^2}{nd}}\right)$, where the expectation is taken over both the uniform and independent input database and query and the random bits of the algorithm.*

This lower bound matches the highest known worst-case cell-probe lower bounds for *any* static data structure problems. Such lower bound was only known for polynomial evaluation [19, 13] and also worst-case deterministic ANN due to our previous work [20].

We also prove an average-case cell-probe lower bound for ANN under ℓ_∞-distance. The lower bound matches the highest known worst-case lower bound for the problem [2].

In fact, we prove these lower bounds in a unified framework that relates the average-case cell-probe complexity of ANN to isoperimetric inequalities regarding an expansion property of the metric space.

Inspired by the notions of metric expansion defined in [17], we define the following notion of expansion for metric space. Let (X, dist) be a metric space. The λ-neighborhood of a point $x \in X$, denoted as $N_\lambda(x)$ is the set of all points in X within distance λ from x. Consider a distribution μ over X. We say the λ-neighborhoods are **weakly independent** under distribution μ, if for any point $x \in X$, the measure of the λ-neighborhood $\mu(N_\lambda(x)) < \frac{\beta}{n}$ for a constant $\beta < 1$. We say the λ-neighborhoods are (Φ, Ψ)**-expanding** under distribution μ, if for any point set $A \subseteq X$ with $\mu(A) \geq \frac{1}{\Phi}$, we have $\mu(N_\lambda(A)) \geq 1 - \frac{1}{\Psi}$, where $N_\lambda(A)$ denotes the set of all points within distance λ from some point in A.

Consider the database $y = (y_1, y_2, \ldots, y_n) \in X^n$ with every point y_i sampled independently from μ, and the query $x \in X$ sampled independently from μ. We denote this input distribution as $\mu \times \mu^n$. We prove the following lower bound.

▶ **Theorem 2.** *For a metric space (X, dist), assume the followings:*

- *the $\gamma\lambda$-neighborhoods are weakly independent under distribution μ;*
- *the λ-neighborhoods are (Φ, Ψ)-expanding under distribution μ.*

Then any deterministic or Las Vegas randomized algorithm solving (γ, λ)-approximate near-neighbor problem in (X, dist) in the cell-probe model with w-bit cells, using a table of size s, must have expected cell-probe complexity

$$t = \Omega\left(\frac{\log \Phi}{\log \frac{sw}{n \log \Psi}}\right) \quad \text{or} \quad t = \Omega\left(\frac{n \log \Psi}{w + \log s}\right)$$

under input distribution $\mu \times \mu^n$.

The key step to prove such a theorem is a stronger version of the richness lemma that we prove in Section 3. The proof of this stronger richness lemma uses an idea called "cell-sampling" introduced by Panigrahy *et al.* [17] and later refined by Larsen [13]. This new richness lemma as well as this connection between the rectangle-based techniques (such as the richness lemma) and information-theory-based techniques (such as cell-sampling) are of interests by themselves.

2 Preliminary

Let (X, dist) be a metric space. Let $\gamma \geq 1$ and $\lambda \geq 0$. The **(γ, λ)-approximate near-neighbor problem** (γ, λ)-ANN_X^n is defined as follows: A database $y = (y_1, y_2, \ldots, y_n) \in X^n$ of n points from X is preprocessed and stored as a data structure. Upon each query $x \in X$, by accessing the data structure we want to distinguish between the following two cases: (1) there is a point y_i in the database such that $\mathrm{dist}(x, z) \leq \lambda$; (2) for all points y_i in the database we have $\mathrm{dist}(x, z) > \gamma\lambda$. For all other cases the answer can be arbitrary.

More abstractly, given a universe X of queries and a universe Y of all databases, a **data structure problem** is a function $f : X \times Y \to Z$ that maps every pair of **query** $x \in X$ and **database** $y \in Y$ to an answer $f(x, y) \in Z$. In our example of (γ, λ)-ANN_X^n, the query universe is the metric space X, the database universe is the set $Y = X^n$ of all tuples of n points from X, and f maps each query $x \in X$ and database $y \in Y$ to an Boolean answer: $f(x, y) = 0$ if there is a λ-near neighbor of x in the database y; $f(x, y) = 1$ if no points in the database y is a $\gamma\lambda$-near neighbor of x; and $f(x, y)$ can be arbitrary if otherwise. Note that due to a technical reason, we usually use 1 to indicate the "no near-neighbor" case.

Given a data structure problem $f : X \times Y \to Z$, a code $T : Y \to \Sigma^s$ with alphabet $\Sigma = \{0, 1\}^w$ encodes every database $y \in Y$ to a **table** T_y of s **cells** with each cell storing a word of w bits. We use $[s] = \{1, 2, \ldots, s\}$ to denote the set of indices of cells. For each $i \in [s]$, we use $T_y[i]$ to denote the content of the i-th cell of table T_y; and for $S \subseteq [s]$, we write $T_y[S] = (T_y[i])_{i \in S}$ for the tuple of the contents of the cells in S. Upon each query $x \in X$, a **cell-probing algorithm** adaptive retrieves the contents of the cells in the table T_y (which is called **cell-probes**) and outputs the answer $f(x, y)$ at last. Being adaptive means that the cell-probing algorithm is actually a decision tree: In each round of cell-probing the address of the cell to probe next is determined by the query x as well as the contents of the cells probed in previous rounds. Together, this pair of code and decision tree is called a **cell-probing scheme**.

For randomized cell-probing schemes, the cell-probing algorithm takes a sequence of random bits as its internal random coin. In this paper we consider only deterministic or Las Vegas randomized cell-probing algorithms, therefore the algorithm is guaranteed to output a correct answer when it terminates.

When a cell-probing scheme is fixed, the size s of the table as well as the length w of each cell are fixed. These two parameters together give the space complexity. And the number of cell-probes may vary for each pair of inputs (x, y) or may be a random variable if the algorithm is randomized. Given a distribution $\mathcal{D}$ over $X \times Y$, the **average-case cell-probe complexity** for the cell-probing scheme is given by the expected number of cell-probes to answer $f(\boldsymbol{x}, \boldsymbol{y})$ for a $(\boldsymbol{x}, \boldsymbol{y})$ sampled from $\mathcal{D}$, where the expectation is taken over both the input distribution $\mathcal{D}$ and the internal random bits of the cell-probing algorithm.

3 A richness lemma for average-case cell-probe complexity

The richness lemma (or the rectangle method) introduced in [15] is a classic tool for proving cell-probe lower bounds. A data structure problem $f : X \times Y \to \{0, 1\}$ is a natural communication problem, and a cell-probing scheme can be interpreted as a communication protocol between the cell-probing algorithm and the table, with cell-probes as communications.

Given a distribution $\mathcal{D}$ over $X \times Y$, a data structure problem $f : X \times Y \to \{0, 1\}$ is **α-dense** under distribution $\mathcal{D}$ if $\mathbb{E}_{\mathcal{D}}[f(\boldsymbol{x}, \boldsymbol{y})] \geq \alpha$. A combinatorial rectangle $A \times B$ for $A \subseteq X$ and $B \subseteq Y$ is a monochromatic 1-rectangle in f if $f(x, y) = 1$ for all $(x, y) \in A \times B$.

The richness lemma states that if a problem f is dense enough (i.e. being rich in 1's) and is easy to solve by communication, then f contains large monochromatic 1-rectangles. Specifically, if an α-dense problem f can be solved by Alice sending a bits and Bob sending b bits in total, then f contains a monochromatic 1-rectangle of size $\alpha \cdot 2^{-O(a)} \times \alpha \cdot 2^{-O(a+b)}$ in the uniform measure. In the cell-probe model with w-bit cells, tables of size s and cell-probe complexity t, it means the monochromatic 1-rectangle is of size $\alpha \cdot 2^{-O(t \log s)} \times \alpha \cdot 2^{-O(t \log s + tw)}$. The cell-probe lower bounds can then be proved by refuting such large 1-rectangles for specific data structure problems f.

We prove the following richness lemma for average-case cell-probe complexity.

▶ **Lemma 3.** *Let μ, ν be distributions over X and Y respectively, and let $f : X \times Y \to \{0,1\}$ be α-dense under the product distribution $\mu \times \nu$. If there is a deterministic or randomized Las Vegas cell-probing scheme solving f on a table of s cells, each cell containing w bits, with expected t cell-probes under input distribution $\mu \times \nu$, then for any $\Delta \in \left[32t/\alpha^2, s\right]$, there is a monochromatic 1-rectangle $A \times B \subseteq X \times Y$ in f such that $\mu(A) \geq \alpha \cdot \left(\frac{\Delta}{s}\right)^{O(t/\alpha^2)}$ and $\nu(B) \geq \alpha \cdot 2^{-O(\Delta \ln \frac{s}{\Delta} + \Delta w)}$.*

Compared to the classic richness lemma, this new lemma has the following advantages:

- It holds for average-case cell-probe complexity.
- It gives stronger result even restricted to worst-case complexity. The newly introduced parameter Δ should not be confused as an overhead caused by the average-case complexity argument, rather, it strengthens the result even for the worst-case lower bounds. When $\Delta = t$ it gives the bound in the classic richness lemma.
- The lemma claims the existence of a *family* of rectangles parameterized by Δ, therefore to prove a cell-probe lower bound it is enough to refute any one rectangle from this family. As we will see, this gives us a power to prove the highest lower bounds (even for the worst case) known to any static data structure problems.

The proof of this lemma uses an argument called "cell-sampling" introduced by Panigrahy *et al.* [16, 17] for approximate nearest neighbor search and later refined by Larsen [13] for polynomial evaluation. Our proof is greatly influenced by Larsen's approach.

The rest of this section is dedicated to the proof of this lemma.

3.1 Proof of the average-case richness lemma (Lemma 3)

By fixing random bits, it is sufficient to consider only deterministic cell-probing algorithms.

The high level idea of the proof is simple. Fix a table T_y. A procedure called the "cell-sampling procedure" chooses the subset Γ of Δ many cells that resolve the maximum amount of positive queries. This associates each database y to a string $\omega = (\Gamma, T_y[\Gamma])$, which we call a **certificate**, where $T_y[\Gamma] = (T_y[i])_{i \in \Gamma}$ represent the contents of the cells in Γ. Due to the nature of the cell-probing algorithm, once the certificate is fixed, the set of queries it can resolve is fixed. We also observe that if the density of 1's in the problem f is $\Omega(1)$, then there is a $\Omega(1)$-fraction of good databases y such that amount of positive queries resolved by the certificate ω constructed by the cell-sampling procedure is at least an $(\frac{\Delta}{s})^{O(t)}$-fraction of all queries. On the other hand, since $\omega \in \binom{[s]}{\Delta} \times \{0,1\}^{\Delta w}$ there are at most $\binom{s}{\Delta} 2^{\Delta w} = 2^{O(\Delta \ln \frac{s}{\Delta} + \Delta w)}$ many certificates ω. Therefore, at least $2^{-O(\Delta \ln \frac{s}{\Delta} + \Delta w)}$-fraction of good databases (which is at least $2^{-O(\Delta \ln \frac{s}{\Delta} + \Delta w)}$-fraction of all databases) are associated with the same ω. Pick this popular certificate ω, the positive queries that ω resolves together with the good databases that ω is associated with form the large monochromatic 1-rectangle.

Now we proceed to the formal parts of the proof. Given a database $y \in Y$, let $X_y^+ = \{x \in X \mid f(x,y) = 1\}$ denote the set of positive queries on y. We use $\mu_y^+ = \mu_{X_y^+}$ to denote the distribution induced by μ on X_y^+.

Let $P_{xy} \subseteq [s]$ denote the set of cells probed by the algorithm to resolve query x on database y. Fix a database $y \in Y$. Let $\Gamma \subseteq [s]$ be a subset of cells. We say a query $x \in X$ is resolved by Γ if x can be resolved by probing only cells in Γ on the table storing database y, i.e. if $P_{xy} \subseteq \Gamma$. We denote by

$$X_y^+(\Gamma) = \{x \in X_y^+ \mid P_{xy} \subseteq \Gamma\}$$

the set of positive queries resolved by Γ on database y. Assume two databases y and y' are *indistinguishable* over Γ: meaning that for the tables T_y and $T_{y'}$ storing y and y' respectively, the cell contents $T_y[i] = T_{y'}[i]$ for all $i \in \Gamma$. Then due to the determinism of the cell-probing algorithm, we have $X_y^+(\Gamma) = X_{y'}^+(\Gamma)$, i.e. Γ resolve the same set of positive queries on both databases.

The cell-sampling procedure

Fix a database $y \in Y$ and any $\Delta \in \left[32t/\alpha^2, s\right]$. Suppose we have a *cell-sampling procedure* which does the following: The procedure deterministically[1] chooses a unique $\Gamma \subseteq [s]$ such that $|\Gamma| = \Delta$ and the measure $\mu(X_y^+(\Gamma))$ of positive queries resolved by Γ is maximized (and if there are more than one such Γ, the procedure chooses an arbitrary one of them). We use Γ_y^* to denote this set of cells chosen by the cell-sampling procedure. We also denote by $X_y^* = X_y^+(\Gamma_y^*)$ the set of positive queries resolved by this chosen set of cells.

On each database y, the cell-sampling procedure chooses for us the most informative set Γ of cells of size $|\Gamma| = \Delta$ that resolve the maximum amount of positive queries. We use $\omega_y = (\Gamma_y^*, T_y[\Gamma_y^*])$ to denote the contents (along with addresses) of the cells chosen by the cell-sampling procedure for database y. We call such ω_y a **certificate** chosen by the cell-sampling procedure for y.

Let y and y' be two databases. A simple observation is that if two databases y and y' have the same certificate $\omega_y = \omega_{y'}$ chosen by the cell-sampling procedure, then the respective sets $X_y^*, X_{y'}^*$ of positive queries resolved on the certificate are going to be the same as well.

▶ **Proposition 4.** *For any databases $y, y' \in Y$, if $\omega_y = \omega_{y'}$ then $X_y^* = X_{y'}^*$.*

Let $\tau(x,y) = |P(x,y)|$ denote the number of cell-probes to resolve query x on database y. By the assumption of the lemma, $\mathbb{E}_{\mu\times\nu}[\tau(\boldsymbol{x},\boldsymbol{y})] \le t$ for the inputs $(\boldsymbol{x},\boldsymbol{y})$ sampled from the product distribution $\mu \times \nu$. We claim that there are many "good" columns (databases) with high density of 1's and low average cell-probe costs.

▶ **Claim 5.** *There is a collection $Y_{\mathsf{good}} \subseteq Y$ of substantial amount of good databases, such that $\nu(Y_{\mathsf{good}}) \ge \frac{\alpha}{4}$ and for every $y \in Y_{\mathsf{good}}$, the followings are true:*

- *the amount of positive queries is large: $\mu(X_y^+) \ge \frac{\alpha}{2}$;*
- *the average cell-probe complexity* among positive queries *is bounded:*

$$\mathbb{E}_{\boldsymbol{x}\sim\mu_y^+}[\tau(\boldsymbol{x},y)] \le \frac{8t}{\alpha^2}.$$

[1] Being deterministic here means that the chosen set Γ_y^* is a function of y.

Proof. The claim is proved by a series of averaging principles. First consider $Y_{\mathsf{dense}} = \{y \in Y \mid \mu(X_y^+) \geq \frac{\alpha}{2}\}$ the set of databases with at least $\frac{\alpha}{2}$-density of positive queries. By the averaging principle, we have $\nu(Y_{\mathsf{dense}}) \geq \alpha/2$. Since $\mathbb{E}[\tau(\boldsymbol{x}, \boldsymbol{y})] \geq \nu(Y_{\mathsf{dense}})\mathbb{E}[\tau(\boldsymbol{x}, \boldsymbol{y}) \mid y \in Y_{\mathsf{dense}}]$, we have $\mathbb{E}_{\mu \times \nu_{\mathsf{dense}}}[\tau(\boldsymbol{x}, \boldsymbol{y})] \leq \frac{2t}{\alpha}$, where $\nu_{\mathsf{dense}} = \nu_{Y_{\mathsf{dense}}}$ is the distribution induced by ν on Y_{dense}. We then construct $Y_{\mathsf{good}} \subseteq Y_{\mathsf{dense}}$ as the set of $y \in Y_{\mathsf{dense}}$ with average cell-probe complexity bounded as $\mathbb{E}_{\boldsymbol{x} \sim \mu}[\tau(\boldsymbol{x}, y)] \leq \frac{4t}{\alpha}$. By Markov inequality $\nu_{\mathsf{dense}}(Y_{\mathsf{good}}) \geq \frac{1}{2}$ and hence $\nu(Y_{\mathsf{good}}) \geq \frac{\alpha}{4}$. Note that $\mathbb{E}_{\boldsymbol{x} \sim \mu}[\tau(\boldsymbol{x}, y)] \geq \mathbb{E}_{\boldsymbol{x} \sim \mu_y^+}[\tau(\boldsymbol{x}, y)]\mu(X_y^+)$. We have $\mathbb{E}_{\boldsymbol{x} \sim \mu_y^+}[\tau(\boldsymbol{x}, y)] \leq \mathbb{E}_{\boldsymbol{x} \sim \mu}[\tau(\boldsymbol{x}, y)]/\mu(X_y^+) \leq \frac{8t}{\alpha^2}$ for all $y \in Y_{\mathsf{good}}$. ◀

For the rest, we consider only these good databases. Fix any $\Delta \in \left[32t/\alpha^2, s\right]$. We claim that for every good database $y \in Y_{\mathsf{good}}$, the cell-sampling procedure always picks a subset $\Gamma_y^* \subseteq [s]$ of Δ many cells, which can resolve a substantial amount of positive queries:

▶ **Claim 6.** *For every $y \in Y_{\mathsf{good}}$, it holds that $\mu(X_y^*) \geq \frac{\alpha}{4}\left(\frac{\Delta}{2s}\right)^{8t/\alpha^2}$.*

Proof. Fix any good database $y \in Y_{\mathsf{good}}$. We only need to prove there exists a $\Gamma \subseteq [s]$ with $|\Gamma| = \Delta$ that resolve positive queries $\mu(X_y^+(\Gamma)) \geq \frac{\alpha}{4}\left(\frac{\Delta}{2s}\right)^{8t/\alpha^2}$. The claims follows immediately.

We construct a hypergraph $\mathcal{H} \subseteq 2^{[s]}$ with vertex set $[s]$ as $\mathcal{H} = \{P_{xy} \mid x \in X_y^+\}$, so that each positive queries $x \in X_y^+$ on database y is associated (many-to-one) to a hyperedge $e \in \mathcal{H}$ such that $e = P_{xy}$ is precisely the set of cells probed by the cell-probing algorithm to resolve query x on database y.

We also define a measure $\tilde{\mu}$ over hyperedges $e \in \mathcal{H}$ as the total measure (in μ_y^+) of the positive queries x associated to e. Formally, for every $e \in \mathcal{H}$,

$$\tilde{\mu}(e) = \sum_{x \in X_y^+ : P_{xy} = e} \mu_y^+(x).$$

Since $\sum_{e \in \mathcal{H}} \tilde{\mu}(e) = \sum_{x \in X_y^+} \mu_y^+(x) = 1$, this $\tilde{\mu}$ is a well-defined probability distribution over hyperedges in $\mathcal{H}$. Moreover, recalling that $\tau(x, y) = |P_{xy}|$, the the average size of hyperedges

$$\mathbb{E}_{\boldsymbol{e} \sim \tilde{\mu}}[|\boldsymbol{e}|] = \mathbb{E}_{\boldsymbol{x} \sim \mu_y^+}[\tau(\boldsymbol{x}, y)] \leq \frac{8t}{\alpha^2}.$$

By the probabilistic method (whose proof is in the full paper [21]), there must exist a $\Gamma \subseteq [s]$ of size $|\Gamma| = \Delta$, such that the sub-hypergraph $\mathcal{H}_\Gamma$ induced by Γ has

$$\tilde{\mu}(\mathcal{H}_\Gamma) \geq \frac{1}{2}\left(\frac{\Delta}{2s}\right)^{8t/\alpha^2}.$$

By our construction of $\mathcal{H}$, the positive queries associated (many-to-one) to the hyperedges in the induced sub-hypergraph $\mathcal{H}_\Gamma = \{P_{xy} \mid x \in X_y^+ \wedge P_{xy} \subseteq \Gamma\}$ are precisely those positive queries in $X_y^+(\Gamma) = \{x \in X_y^+ \mid P_{xy} \subseteq \Gamma\}$. Therefore,

$$\mu_y^+(X_y^+(\Gamma)) = \sum_{x \in X_y^+, P_{xy} \subseteq \Gamma} \mu_y^+(x) = \tilde{\mu}(\mathcal{H}_\Gamma) \geq \frac{1}{2}\left(\frac{\Delta}{2s}\right)^{8t/\alpha^2}.$$

Recall that $\mu(X_y^+) \geq \frac{\alpha}{2}$ for every $y \in Y_{\mathsf{good}}$. And since $X_y^+(\Gamma) \subseteq X_y^+$, we have

$$\mu(X_y^+(\Gamma)) = \mu_y^+(X_y^+(\Gamma))\mu(X_y^+) \geq \frac{\alpha}{4}\left(\frac{\Delta}{2s}\right)^{8t/\alpha^2}.$$

The claim is proved. ◀

Recall that the certificate $\omega_y = (\Gamma_y^*, T_y[\Gamma_y^*])$ is constructed by the cell-sampling procedure for database y. For every possible assignment $\omega \in \binom{[s]}{\Delta} \times \{0,1\}^{\Delta w}$ of certificate, let Y_ω denote the set of good databases $y \in Y_{\mathsf{good}}$ with this certificate $\omega_y = \omega$. Due to the determinism of the cell-sampling procedure, this classifies the Y_{good} into at most $\binom{s}{\Delta}2^{\Delta w}$ many disjointed subclasses Y_ω. Recall that $\nu(Y_{\mathsf{good}}) \geq \frac{\alpha}{4}$. By the averaging principle, the following proposition is natural.

▶ **Proposition 7.** *There exists a certificate $\omega \in \binom{[s]}{\Delta} \times \{0,1\}^{\Delta w}$, denoted as ω^*, such that*

$$\nu(Y_{\omega^*}) \geq \frac{\alpha}{4\binom{s}{\Delta}2^{\Delta w}}.$$

On the other hand, fixed any ω, since all databases $y \in Y_\omega$ have the same ω_y^*, by Proposition 4 they must have the same X_y^*. We can abuse the notation and write $X_\omega = X_y^*$ for all $y \in Y_\omega$.

Now we let $A = X_{\omega^*}$ and $B = Y_{\omega^*}$, where ω^* satisfies Proposition 7. Due to Claim 6 and Proposition 7, we have

$$\mu(A) \geq \frac{\alpha}{4}\left(\frac{\Delta}{2s}\right)^{8t/\alpha^2} = \alpha \cdot \left(\frac{\Delta}{s}\right)^{O(t/\alpha^2)} \quad \text{and} \quad \nu(B) \geq \frac{\alpha}{4\binom{s}{\Delta}2^{\Delta w}} = \alpha \cdot 2^{-O\left(\Delta \ln \frac{s}{\Delta} + \Delta w\right)}.$$

Note for every $y \in B = Y_{\omega^*}$, the $A = X_{\omega^*} = X_y^+(\Gamma_y^*)$ is a set of positive queries on database y, thus $A \times B$ is a monochromatic 1-rectangle in f. This finishes the proof of Lemma 3.

4 Rectangles in conjunction problems

Many natural data structure problems can be expressed as a conjunction of point-wise relations between the query point and database points. Consider data structure problem $f : X \times Y \to \{0,1\}$. Let $Y = \mathcal{Y}^n$, so that each database $y \in Y$ is a tuple $y = (y_1, y_2, \ldots, y_n)$ of n points from $\mathcal{Y}$. A **point-wise function** $g : X \times \mathcal{Y} \to \{0,1\}$ is given. The data structure problem f is defined as the conjunction of these subproblems:

$$\forall x \in X, \forall y = (y_1, y_2, \ldots, y_n) \in Y, \quad f(x,y) = \bigwedge_{i=1}^{n} g(x, y_i)\,.$$

Many natural data structure problems can be defined in this way, for example:

- Membership query: $X = \mathcal{Y}$ is a finite domain. The point-wise function $g(\cdot,\cdot)$ is $\neq$ that indicates whether the two points are unequal.
- (γ,λ)-approximate near-neighbor (γ,λ)-ANN_X^n: $X = \mathcal{Y}$ is a metric space with distance $\mathrm{dist}(\cdot,\cdot)$. The point-wise function g is defined as: for $x, z \in X$, $g(x,z) = 1$ if $\mathrm{dist}(x,z) > \gamma\lambda$, or $g(x,z) = 0$ if $\mathrm{dist}(x,z) \leq \lambda$. The function value can arbitrary for all other cases.
- Partial match $\mathsf{PM}_\Sigma^{d,n}$: Σ is an alphabet, $\mathcal{Y} = \Sigma^d$ and $X = (\Sigma \cup \{\star\})^d$. The point-wise function g is defined as: for $x \in X$ and $z \in \mathcal{Y}$, $g(x,z) = 1$ if there is an $i \in [d]$ such that $x_i \notin \{\star, z_i\}$, or $g(x,z) = 0$ if otherwise.

We show that refuting the large rectangles in the point-wise function g can give us lower bounds for the conjunction problem f.

Let μ, ν be distributions over X and $\mathcal{Y}$ respectively, and let ν^n be the product distribution on $Y = \mathcal{Y}^n$. Let $g : X \times \mathcal{Y} \to \{0,1\}$ be a point-wise function and $f : X \times Y \to \{0,1\}$ a data structure problem defined by the conjunction of g as above.

▶ **Lemma 8.** *For f, g, μ, ν defined as above, assume that there is a deterministic or randomized Las Vegas cell-probing scheme solving f on a table of s cells, each cell containing w bits, with expected t cell-probes under input distribution $\mu \times \nu^n$. If the followings are true:*

- *the density of 0's in g is at most $\frac{\beta}{n}$ under distribution $\mu \times \nu$ for some constant $\beta < 1$;*
- *g does not contain monochromatic 1-rectangle of measure at least $\frac{1}{\Phi} \times \frac{1}{\Psi}$ under distribution $\mu \times \nu$;*

then

$$\left(\frac{sw}{n \log \Psi}\right)^{O(t)} \geq \Phi \quad \text{or} \quad t = \Omega\left(\frac{n \log \Psi}{w + \log s}\right).$$

Proof. By union bound, the density of 0's in f under distribution $\mu \times \nu^n$ is:

$$\Pr_{\substack{x \sim \mu \\ y=(y_1,\ldots,y_n)\sim\nu^n}}\left[\bigwedge_{i=1}^{n} g(x, y_i) = 0\right] \leq n \cdot \Pr_{\substack{x\sim\mu \\ z\sim\nu}}[g(x, z) = 0] \leq n \cdot \frac{\beta}{n} = \beta.$$

By Lemma 3, the $\Omega(1)$-density of 1's in f and the assumption of existing a cell-probing scheme with parameters s, w and t, altogether imply that for any $4t \leq \Delta \leq s$, f has a monochromatic 1-rectangle $A \times B$ such that

$$\mu(A) \geq \left(\frac{\Delta}{s}\right)^{c_1 t} \quad \text{and} \quad \nu^n(B) \geq 2^{-c_2 \Delta(\ln \frac{s}{\Delta} + w)}, \tag{1}$$

for some constants $c_1, c_2 > 0$ depending only on β.

Let $C \subset \mathcal{Y}$ be the largest set of columns in g to form a 1-rectangle with A. Formally,

$$C = \{z \in \mathcal{Y} \mid \forall x \in A, g(x, z) = 1\}.$$

Clearly, for any monochromatic 1-rectangle $A \times D$ in g, we must have $D \subseteq C$. By definition of f as a conjunction of g, it must hold that for all $y = (y_1, y_2, \ldots, y_n) \in B$, none of $y_i \in y$ has $g(x, y_i) = 0$ for any $x \in A$, which means $B \subseteq C^n$, and hence

$$\nu^n(B) \leq \nu^n(C^n) = \nu(C)^n.$$

Recall that $A \times C$ is monochromatic 1-rectangle in g. Due to the assumption of the lemma, either $\mu(A) < \frac{1}{\Phi}$ or $\nu(C) < \frac{1}{\Psi}$. Therefore, either $\mu(A) < \frac{1}{\Phi}$ or $\nu^n(B) < \frac{1}{\Psi^n}$.

We can always choose a Δ such that $\Delta = O\left(\frac{n \log \Psi}{w}\right)$ and $\Delta = \Omega\left(\frac{n \log \Psi}{w + \log s}\right)$ to satisfy

$$2^{-c_2 \Delta(\ln \frac{s}{\Delta} + w)} > \frac{1}{\Psi^n}.$$

If such Δ is less than $32t/(1-\beta)^2$, then we immediately have a lower bound

$$t = \Omega\left(\frac{n \log \Psi}{w + \log s}\right).$$

Otherwise, due to (1), $A \times B$ is monochromatic 1-rectangle in f with $\nu^n(B) > \frac{1}{\Psi^n}$, therefore it must hold that $\mu(A) < \frac{1}{\Phi}$, which by (1) gives us

$$\frac{1}{\Phi} > \mu(A) \geq \left(\frac{\Delta}{s}\right)^{O(t)} = \left(\frac{n \log \Psi}{sw}\right)^{O(t)},$$

which gives the lower bound

$$\left(\frac{sw}{n \log \Psi}\right)^{O(t)} \geq \Phi.$$

◀

5 Isoperimetry and ANN lower bounds

Given a metric space X with distance $\mathrm{dist}(\cdot,\cdot)$ and $\lambda \geq 0$, we say that two points $x, x' \in X$ are λ-close if $\mathrm{dist}(x,x') \leq \lambda$, and λ-far if otherwise. The λ-neighborhood of a point $x \in X$, denoted by $N_\lambda(x)$, is the set of all points from X which are λ-close to x. Given a point set $A \subseteq X$, we define $N_\lambda(A) = \bigcup_{x\in A} N_\lambda(x)$ to be the set of all points which are λ-close to some point in A.

In [17], a natural notion of metric expansion was introduced.

▶ **Definition 9** (metric expansion [17])**.** Let X be a metric space and μ a probability distribution over X. Fix any radius $\lambda > 0$. Define

$$\Phi(\delta) \triangleq \min_{A\subset X, \mu(A)\leq\delta} \frac{\mu(N_\lambda(A))}{\mu(A)}.$$

The expansion Φ of the λ-neighborhoods in X under distribution μ is defined as the largest k such that for all $\delta \leq \frac{1}{2k}$, $\Phi(\delta) \geq k$.

We now introduce a more refined definition of metric expansion using two parameters Φ and Ψ.

▶ **Definition 10** ((Φ,Ψ)-expanding)**.** Let X be a metric space and μ a probability distribution over X. The λ-neighborhoods in X are **(Φ,Ψ)-expanding** under distributions μ if we have $\mu(N_\lambda(A)) \geq 1 - 1/\Psi$ for any $A \subseteq X$ that $\mu(A) \geq 1/\Phi$.

The metric expansion defined in [17] is actually a special case of (Φ,Ψ)-expanding: The expansion of λ-neighborhoods in a metric space X is Φ means the λ-neighborhoods are $(\Phi,2)$-expanding. The notion of (Φ,Ψ)-expanding allows us to describe a more extremal expanding situation in metric space: The expanding of λ-neighborhoods does not stop at measure $1/2$, rather, it can go all the way to be very close to measure 1. This generality may support higher lower bounds for approximate near-neighbor.

Given a radius $\lambda > 0$ and an approximation ratio $\gamma > 1$, recall that the (γ,λ)-approximate near neighbor problem (γ,λ)-ANN^n_X can be defined as a conjunction $f(x,y) = \bigwedge_i g(x,y_i)$ of point-wise function $g : X \times X \to \{0,1\}$ where $g(x,z) = 0$ if x is λ-close to z; $g(x,z) = 1$ if x is $\gamma\lambda$-far from z; and $g(x,z)$ is arbitrary for all other cases. Observe that g is actually (γ,λ)-ANN^1_X, the point-to-point version of the (γ,λ)-approximate near neighbor.

The following proposition gives an intrinsic connection between the expansion of metric space and size of monochromatic rectangle in the point-wise near-neighbor relation.

▶ **Proposition 11.** *If the λ-neighborhoods in X are (Φ,Ψ)-expanding under distribution μ, then the function g defined as above does not contain a monochromatic 1-rectangle of measure $\geq \frac{1}{\Phi} \times \frac{1.01}{\Psi}$ under distribution $\mu \times \mu$.*

Proof. Since the λ-neighborhoods in X are (Φ,Ψ)-expanding, for any $A \subseteq X$ with $\mu(A) \geq \frac{1}{\Phi}$, we have $\mu(N_\lambda(A)) \geq 1 - \frac{1}{\Psi}$. And by definition of g, for any monochromatic $A \times B$, it must hold that $B \cap N_\lambda(A) = \emptyset$, i.e. $B \subseteq X \setminus N_\lambda(A)$. Therefore, either $\mu(A) < \frac{1}{\Phi}$, or $\mu(B) = 1 - \mu(N_\lambda(A)) \leq \frac{1}{\Psi} < \frac{1.01}{\Psi}$. ◀

The above proposition together with Lemma 8 immediately gives us the following corollary which reduces lower bounds for near-neighbor problems to the isoperimetric inequalities.

▶ **Corollary 12.** *Let μ be a distribution over a metric space X. Let $\lambda > 0$ and $\gamma \geq 1$. Assume that there is a deterministic or randomized Las Vegas cell-probing scheme solving (γ,λ)-ANN^n_X on a table of s cells, each cell containing w bits, with expected t cell-probes under input distribution $\mu \times \mu^n$. If the followings are true:*

- $\mathbb{E}_{x\sim\mu}[\mu(N_{\gamma\lambda}(x))] \le \frac{\beta}{n}$ *for a constant* $\beta < 1$*;*
- *the* λ*-neighborhoods in* X *are* (Φ, Ψ)*-expanding under distribution* μ*;*

then

$$\left(\frac{sw}{n\log\Psi}\right)^{O(t)} \ge \Phi \quad \text{or} \quad t = \Omega\left(\frac{n\log\Psi}{w+\log s}\right).$$

▶ Remark. In [17], a lower bound for (γ,λ)-ANN^n_X was proved with the following form:

$$\left(\frac{swt}{n}\right)^t \ge \Phi.$$

In our Corollary 12, unless the cell-size w is unrealistically large to be comparable to n, the corollary always gives the first lower bound

$$\left(\frac{sw}{n\log\Psi}\right)^{O(t)} \ge \Phi.$$

This strictly improves the lower bound in [17]. For example, when the metric space is $\left(2^{\Theta(d)}, 2^{\Theta(d)}\right)$-expanding, this would give us a lower bound $t = \Omega\left(\frac{d}{\log\frac{sw}{nd}}\right)$, which in particular, when the space is linear ($sw = O(nd)$), becomes $t = \Omega(d)$.

5.1 Lower bound for ANN in Hamming space

Let $X = \{0,1\}^d$ be the Hamming space with Hamming distance $\mathrm{dist}(\cdot,\cdot)$. Recall that $N_\lambda(x)$ represents the λ-neighborhood around x, in this case, the Hamming ball of radius λ centered at x; and for a set $A \subset X$, the $N_\lambda(A)$ is the set of all points within distance λ to any point in A. For any $0 \le r \le d$ $B(r) = |N_r(\bar{0})|$ denote the volume of Hamming ball of radius r, where $\bar{0} \in \{0,1\}^d$ is the zero vector. Obviously $B(r) = \sum_{k\le r}\binom{d}{k}$.

The following isoperimetric inequality of Harper is well known.

▶ **Lemma 13** (Harper's theorem [9]). *Let* $X = \{0,1\}^d$ *be the* d*-dimensional Hamming space. For* $A \subset X$*, let* r *be such that* $|A| \ge B(r)$*. Then for every* $\lambda > 0$*,* $|N_\lambda(A)| \ge B(r+\lambda)$*.*

In words, Hamming balls have the worst vertex expansion.

For $0 < r < \frac{d}{2}$, the following upper bound for the volume of Hamming ball is well known:

$$2^{(1-o(1))dH(r/d)} \le \binom{d}{r} \le B(r) \le 2^{dH(r/d)},$$

where $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$ is the Boolean entropy function.

Consider the Hamming (γ,λ)-approximate near-neighbor problem (γ,λ)-ANN^n_X. The hard distribution for this problem is just the uniform and independent distribution: For the database $y = (y_1, y_2, \ldots, y_n) \in X^n$, each database point y_i is sampled uniformly and independently from $X = \{0,1\}^n$; and the query point x is sampled uniformly and independently from X.

▶ **Theorem 14.** *Let* $d \ge 32\log n$*. For any* $\gamma \ge 1$*, there is a* $\lambda > 0$ *such that if* (γ,λ)-ANN^n_X *can be solved by a deterministic or Las Vegas randomized cell-probing scheme on a table of* s *cells, each cell containing* w *bits, with expected* t *cell-probes for uniform and independent database and query, then* $t = \Omega\left(\frac{d}{\gamma^2\log\frac{sw\gamma^2}{nd}}\right)$ *or* $t = \Omega\left(\frac{nd}{\gamma^2(w+\log s)}\right)$.

Proof. Choose λ to satisfy $\gamma\lambda = \frac{d}{2} - \sqrt{2d\ln(2n)}$. Let μ be uniform distribution over X. We are going to show:

- $\mathbb{E}_{x\sim\mu}[\mu(N_{\gamma\lambda}(x))] \le \frac{1}{2n}$;
- the λ-neighborhoods in X are (Φ, Ψ)-expanding under distribution μ for some $\Phi = 2^{\Omega(d/\gamma^2)}$ and $\Psi = 2^{\Omega(d/\gamma^2)}$.

Then the cell-probe lower bounds follows directly from Corollary 12.

First, by the Chernoff bound, $\mu(N_{\gamma\lambda}(x)) \le \frac{1}{2n}$ for any point $x \in X$. Thus trivially $\mathbb{E}_{x\sim\mu}[\mu(N_{\gamma\lambda}(x))] \le \frac{1}{2n}$.

On the other hand, for $d \ge 32\log n$ and n being sufficiently large, it holds that $\lambda \ge \frac{d}{4\gamma}$. Let $r = \frac{d}{2} - \frac{d}{8\gamma}$. And consider any $A \subseteq X$ with $\mu(A) \ge 2^{-(1-H(r/d))d}$. We have $|A| \ge 2^{dH(r/d)} \ge B(r)$. Then by Harper's theorem,

$$|N_\lambda(A)| \ge B\left(r+\lambda\right) \ge B\left(\tfrac{d}{2} + \tfrac{d}{8\gamma}\right) \ge 2^d - B\left(\tfrac{d}{2} - \tfrac{d}{8\gamma}\right) = 2^d - B(r) \ge 2^d - 2^{dH(r/d)},$$

which means $\mu(N_\lambda(A)) \ge 1 - 2^{-(1-H(r/d))d}$. In other words, the λ-neighborhoods in X are (Φ, Ψ)-expanding under distribution μ for $\Phi = \Psi = 2^{(1-H(r/d))d}$, where $r/d = \frac{1}{2} - \frac{1}{8\gamma}$. Apparently $1 - H(\frac{1}{2} - x) = \Theta(x^2)$ for small enough $x > 0$. Hence, $\Phi = \Psi = 2^{\Theta(d/\gamma^2)}$. ◀

5.2 Lower bound for ANN under L-infinity norm

Let $\Sigma = \{0, 1, \ldots, m\}$ and the metric space is $X = \Sigma^d$ with ℓ_∞ distance $\mathrm{dist}(x, y) = \|x - y\|_\infty$ for any $x, y \in X$.

Let μ be the distribution over X as defined in [2]: First define a distribution π over Σ as $p(i) = 2^{-(2\rho)^i}$ for all $i > 0$ and $\pi(0) = 1 - \sum_{i>0}\pi(i)$; and then μ is defined as $\mu(x_1, x_2, \ldots, x_d) = \pi(x_1)\pi(x_2)\ldots\pi(x_d)$.

The following isoperimetric inequality is proved in [2].

▶ **Lemma 15** (Lemma 9 of [2]). *For any $A \subseteq X$, it holds that $\mu(N_1(A)) \ge (\mu(A))^{1/\rho}$.*

Consider the (γ, λ)-approximate near-neighbor problem (γ, λ)-$\mathsf{ANN}^n_{\ell_\infty}$ defined in the metric space X under ℓ_∞ distance. The hard distribution for this problem is $\mu \times \mu^n$: For the database $y = (y_1, y_2, \ldots, y_n) \in X^n$, each database point y_i is sampled independently according to μ; and the query point x is sampled independently from X according to μ. The following lower bound has been proved in [2] and [12].

Fix any $\epsilon > 0$ and $0 < \delta < \frac{1}{2}$. Assume $\Omega\left(\log^{1+\epsilon} n\right) \le d \le o(n)$. For $3 < c \le O(\log\log d)$, define $\rho = \frac{1}{2}(\frac{\epsilon}{4}\log d)^{1/c} > 10$. Now we choose $\gamma = \log_\rho \log d$ and $\lambda = 1$.

▶ **Theorem 16.** *With d, γ, λ, ρ and the metric space X defined as above, if (γ, λ)-$\mathsf{ANN}^n_{\ell_\infty}$ can be solved by a deterministic or Las Vegas randomized cell-probing scheme on a table of s cells, each cell containing $w \le n^{1-2\delta}$ bits, with expected $t \le \rho$ cell-probes under input distribution $\mu \times \mu^n$, then $sw = n^{\Omega(\rho/t)}$.*

Proof. The followings are true

- $\mu(N_{\gamma\lambda}(x)) = \frac{e^{-\log^{1+\epsilon/3} n}}{n} \le \frac{1}{2n}$ for any $x \in X$ (Claim 6 in [2]);
- the λ-neighborhoods in X are $(n^{\delta\rho}, \frac{n^\delta}{n^\delta - 1})$-expanding under distribution μ for $\Phi = n^{\delta\rho}$ and $\Psi = 2^{\Omega(d/\gamma^2)}$.

To see the expansion is true, let $\Phi = n^{\delta\rho}$ and $\Psi = \frac{n^\delta}{n^\delta - 1}$. By Lemma 15, for any set $A \subset X$ with $\mu(A) \ge \Phi$, we have $\mu(N_\lambda(A)) \ge n^{-\delta} \ge 1 - \frac{1}{\Psi}$. This means λ-neighborhoods of $\mathcal{M}$ are $(n^{\delta\rho}, \frac{n^\delta}{n^\delta - 1})$-expanding.

Due to Corollary 12, either $\left(\frac{sw}{n^{1-\delta}}\right)^{O(t)} \ge n^{\delta\rho}$ or $= \Omega\left(\frac{n^{1-\delta}}{w + \log s}\right)$. The second bound is always higher with our ranges for w and t. The first bound gives $sw = n^{\Omega(\rho/t)}$. ◀

References

1. Amirali Abdullah and Suresh Venkatasubramanian. A directed isoperimetric inequality with application to bregman near neighbor lower bounds. In *STOC'15*.
2. Alexandr Andoni, Dorian Croitoru, and Mihai Pătraşcu. Hardness of nearest neighbor under L-infinity. In *FOCS'08*.
3. Alexandr Andoni, Piotr Indyk, and Mihai Pătraşcu. On the optimality of the dimensionality reduction method. In *FOCS'06*.
4. Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *STOC'15*.
5. Omer Barkol and Yuval Rabani. Tighter lower bounds for nearest neighbor search and related problems in the cell probe model. *Journal of Computer and System Sciences*, 64(4):873–896, 2002. Conference version in *STOC'00*.
6. Allan Borodin, Rafail Ostrovsky, and Yuval Rabani. Lower bounds for high dimensional nearest neighbor search and related problems. In *Discrete and Computational Geometry*, pages 253–274, 2003. Conference version in *STOC'99*.
7. Amit Chakrabarti, Bernard Chazelle, Benjamin Gum, and Alexey Lvov. A lower bound on the complexity of approximate nearest-neighbor searching on the hamming cube. In *Discrete and Computational Geometry*, pages 313–328, 2003. Conference version in *STOC'99*.
8. Amit Chakrabarti and Oded Regev. An optimal randomised cell probe lower bound for approximate nearest neighbour searching. In *SIAM Journal on Computing*, 39(5):1919–1940,2010. Conference version in *FOCS'04*.
9. L.H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1(3):385 – 393, 1966.
10. Piotr Indyk. Nearest neighbors in high-dimensional spaces. *Handbook of Discrete and Computational Geometry*, pages 877–892, 2004.
11. T.S. Jayram, Subhash Khot, Ravi Kumar, and Yuval Rabani. Cell-probe lower bounds for the partial match problem. In *Journal of Computer and System Sciences*, 69(3):435–447, 2004. Conference version in *STOC'03*.
12. Michael Kapralov and Rina Panigrahy. NNS lower bounds via metric expansion for ℓ_∞ and EMD. In *ICALP'12*.
13. Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *FOCS'12*.
14. Ding Liu. A strong lower bound for approximate nearest neighbor searching. *Information Processing Letters*, 92(1):23–29, 2004.
15. Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *Journal of Computer and System Sciences*, 57(1):37–49, 1998. Conference version in *STOC'95*.
16. Rina Panigrahy, Kunal Talwar, and Udi Wieder. A geometric approach to lower bounds for approximate near-neighbor search and partial match. In *FOCS'08*.
17. Rina Panigrahy, Kunal Talwar, and Udi Wieder. Lower bounds on near neighbor search via metric expansion. In *FOCS'10*.
18. Mihai Pătraşcu and Mikkel Thorup. Higher lower bounds for near-neighbor and further rich problems. *SIAM Journal on Computing*, 39(2):730–741, 2010. Conference version in *FOCS'06*.
19. Alan Siegel. On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications. In *FOCS'89*.
20. Yaoyu Wang and Yitong Yin. Certificates in data structures. In *ICALP'14*.
21. Yitong Yin. Simple average-case lower bounds for approximate near-neighbor from isoperimetric inequalities. *arXiv preprint* arXiv:1602.05391.

Quasimetric Embeddings and Their Applications*†

Facundo Mémoli[1], Anastasios Sidiropoulos[2], and Vijay Sridhar[3]

1 **Dept. of Computer Science and Engineering and Dept. of Mathematics, The Ohio State University, Columbus, OH, USA**
memoli@math.osu.edu

2 **Dept. of Computer Science and Engineering and Dept. of Mathematics, The Ohio State University, Columbus, OH, USA**
sidiropoulos.1@osu.edu

3 **Dept. of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA**
sridhar.38@buckeyemail.osu.edu

Abstract

We study generalizations of classical metric embedding results to the case of *quasimetric* spaces; that is, spaces that do not necessarily satisfy symmetry. Quasimetric spaces arise naturally from the shortest-path distances on directed graphs. Perhaps surprisingly, very little is known about low-distortion embeddings for quasimetric spaces.

Random embeddings into *ultrametric* spaces are arguably one of the most successful geometric tools in the context of algorithm design. We extend this to the quasimetric case as follows. We show that any n-point quasimetric space supported on a graph of treewidth t admits a random embedding into *quasiultrametric* spaces with distortion $O(t \log^2 n)$, where quasiultrametrics are a natural generalization of ultrametrics. This result allows us to obtain $t \log^{O(1)} n$-approximation algorithms for the Directed Non-Bipartite Sparsest-Cut and the Directed Multicut problems on n-vertex graphs of treewidth t, with running time polynomial in both n and t.

The above results are obtained by considering a generalization of random partitions to the quasimetric case, which we refer to as *random quasipartitions*. Using this definition and a construction of [Chuzhoy and Khanna 2009] we derive a polynomial lower bound on the distortion of random embeddings of general quasimetric spaces into quasiultrametric spaces. Finally, we establish a lower bound for embedding the shortest-path quasimetric of a graph G into graphs that exclude G as a minor. This lower bound is used to show that several embedding results from the metric case do not have natural analogues in the quasimetric setting.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems – Geometrical problems and computations

Keywords and phrases metric embeddings, quasimetrics, outliers, random embeddings, treewidth, Directed Sparsest-Cut, Directed Multicut

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.85

1 Introduction

Low-distortion embeddings of metric spaces have become an indispensable tool in the design of algorithms [22, 3, 5, 13]. We consider generalizations of some fundamental metric embedding

* A full version of the paper is available at https://arxiv.org/abs/1608.01396.

† This work was supported by NSF grants CCF-1423230, CCF-1526513, IIS-1422400, and award CAREER-1453472.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 85; pp. 85:1–85:14

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

problems to the case of *quasimetric spaces*. Formally, a quasimetric space is a pair (X, d) where X is the set of points and $d : X \times X \to \mathbb{R}_+ \cup \{+\infty\}$, satisfying the following conditions:
(C1) For all $x, y \in X$, $d(x, y) = 0$ iff $x = y$.
(C2) For all $x, y, z \in X$, $d(x, y) \leq d(x, z) + d(z, y)$.

In other words, a quasimetric space satisfies all the conditions of a metric space, except for the following symmetry condition:
(C3) For all $x, y \in X$, $d(x, y) = d(y, x)$.

Finite quasimetric spaces are precisely the shortest-path distances of finite directed graphs. Perhaps surprisingly, many basic questions regarding low-distortion embeddings of quasimetric spaces are poorly understood. As we explain below, these problems are of importance in algorithm design, and are intricately tied to the approximability of various cut problems on directed edge-capacitated graphs.

1.1 Our contributions

We now outline our main results, and contrast with what was previously known for the case of metric spaces. We consider quasimetric spaces that arise from the shortest-path distances of directed graphs. For a family $\mathcal{F}$ of undirected graphs, we consider the directed graphs arising from the graphs in $\mathcal{F}$ by replacing every undirected edge $\{u, v\}$ by two edges (u, v) and (v, u) with opposite directions, and by assigning arbitrary positive edge lengths to them. When $\mathcal{F}$ is the family of all trees (resp. graphs of treewidth-t), we refer to the resulting family of quasimetric spaces as *tree quasimetric spaces* (resp. *treewidth-t quasimetric spaces*).

Random embeddings

A very successful metric embedding tool in the context of algorithm design is random embeddings. The high-level idea is that given some "complicated" space, we can find a random embedding into some "simpler" space, preserving all distances in expectation.

Formally, let $M = (X, d)$ be a quasimetric space. A *random embedding* of M is a distribution $\mathcal{F}$ over pairs (f, M') where $M' = (X', d')$ is a quasimetric space and $f : X \to X'$, such that for any $x, y \in X$

$$\Pr[d'(f(x), f(y)) \geq d(x, y)] = 1.$$

Let $\alpha \geq 1$. We say that the random embedding $\mathcal{F}$ has *distortion* α if for all $x, y \in X$

$$\mathbb{E}\,[d'(f(x), f(y))] \leq \alpha\, d(x, y).$$

This definition is of algorithmic interest because for several optimization problems it allows us to reduce instances on general graphs to instances on simpler graphs (see [4, 23] for a more detailed exposition). It has been shown that any n-point metric space admits a random embedding into *ultrametric* spaces with distortion $O(\log n)$ [10] (see also [4, 5, 1]). Here, an ultrametric space is a metric space that satisfies the following stronger version of the triangle inequality:
(C2*) For all $x, y, z \in X$, $d(x, y) \leq \max\{d(x, z), d(z, y)\}$.

It is easy to construct examples of quasimetric spaces that do not admit random embeddings into ultrametric spaces with bounded distortion. This motivates the study of random embeddings of quasimetric spaces into *quasiultrametric* spaces; these are precisely the quasimetric spaces that satisfy conditions (C1) & (C2*). We show that any n-point

treewidth-t quasimetric space admits a random embedding into quasiultrametric spaces with distortion $O(t \log^2 n)$. In a similar fashion, we show that any treewidth-t quasiultrametric admits an embedding into a convex combination of 0-1 quasimetric spaces with distortion $O(t \log^2 n)$; here, a 0-1 quasimetric space requires that all distances are either 0 or 1. As we explain below, this result allows us to obtain new approximation algorithms for directed cut problems on treewidth-t graphs.

Random quasipartitions

A fundamental primitive underlying many metric embedding results is *random partitions* [4]. This primitive has been successfully used in many diverse prolems [16, 18, 15, 19, 7, 20, 21]. It is easy to construct examples of quasimetric spaces that do not admit good random partitions. We overcome this technical obstacle by defining a *quasipartition* to be a transitive reflexive relation. This is a generalization of a partition for the following reason: For a partition P of some set X we can define the relation R on X where for all $x, y \in X$, we set $(x, y) \in R$ iff x and y are in the same cluster in P. It is easy to check that R is indeed transitive and reflexive; however, there are transitive and reflexive relations that do not arise in this fashion.

Let $r \geq 0$. We say that a quasipartition P of M is *r-bounded* if for any $x, y \in X$, if $(x, y) \in P$, then $d(x, y) \leq r$. Let $\mathcal{D}$ be a distribution over r-bounded quasipartitions of M. We say that $\mathcal{D}$ is r-bounded. We also say that $\mathcal{D}$ is *β-Lipschitz*, for some $\beta > 0$, if for any $x, y \in X$, we have that

$$\Pr_{P \sim \mathcal{D}}[(x, y) \notin P] \leq \beta \frac{d(x, y)}{r}.$$

Given a distribution $\mathcal{D}$ over quasipartitions we ocasionally refer to any quasipartition P sampled from $\mathcal{D}$ as a random quasipartition (with distribution $\mathcal{D}$).

We show that for all $r > 0$, any tree quasimetric space admits a $O(1)$-Lipschitz, r-bounded random quasipartition. We remark that no such result is possible using random partitions. We further show that for all $r > 0$, any treewidth-t quasimetric space admits a $O(t \log n)$-Lipschitz, r-bounded random quasipartition. This random quasipartition is at the heart of the random embedding result outlined above.

Using a result of Chuzhoy and Khanna [9] we show that the polynomial dependence on the treewidth is necessary. More precisely, there exist n-point quasimetric spaces that do not admit $o(n^{1/7}/\log^{4/7} n)$-Lipschitz quasipartitions. Using this lower bound we further show that there exist n-point quasimetrics that do not admit random embeddings into quasiultrametric spaces with distortion $o(n^{1/7}/\log^{4/7})$.

Applications to cut problems on directed graphs

Using the above result for embedding treewidth-t quasimetric spaces into a convex combination of 0-1 quasimetric spaces, we show that the integrality gap of the Directed Non-Bipartite Sparsest-Cut LP on graphs of treewidth t it $O(t \log^2 n)$. This implies a $O(t\sqrt{\log t} \log^2 n)$-approximation algorithm for the Directed Sparsest-Cut problem with running time polynomial in both n and t. We remark that dynamic-programming based techniques for graphs of bounded treewidth can only yield algorithms with running time exponential in t, and are thus practical only for very small values of t. Our result provides an interesting trade-off between running time and approximation guarantee for graphs of moderately large treewidth. For example, our result implies a polynomial time $\log^{O(1)}$-approximation for Directed Non-Bipartite Sparsest-Cut on graphs of treewidth $\log^{O(1)} n$.

Similarly, we obtain a $O(t\sqrt{\log t}\log^3 n)$-approximation algorithm for the Directed Multicut problem on graphs of treewidth t, with running time polynomial in both n and t.

Lower bound for random topological simplification

It has been show that for various classes of topologically restricted graphs, there exist constant-distortion random embeddings into topologically simpler graphs. For example, graphs of bounded genus admit constant-distortion random embeddings into planar graphs [14, 6, 23], and similar results are known for more general classes of minor-free graphs [20].

We show that no such result is possible for the case of directed graphs. More precisely, we show that for any directed acyclic graph G, there exists a subdivision G' of G, such that for any embedding of the shortest-path quasimetric of G' into the shortest-path quasimetric of some graph H with bounded distortion, we have that G is a minor of H. For example, this implies that there is no bounded-distortion random embedding of toroidal (i.e. genus-1) quasimetric spaces into planar quasimetric spaces.

2 Quasipartitions of tree quasimetrics

In this section we describe a method to construct an $O(1)$-Lipschitz distribution over r-bounded quasipartitions of tree quasimetric spaces. More precisely, we prove the following result.

▶ **Theorem 1.** *Let M be a shortest path quasimetric space supported on some directed tree T in which every edge has non-negative weights in both directions. For any $r > 0$, there exists an $O(1)$-Lipschitz distribution over r-bounded quasipartitions of M.*

First we describe an algorithm to construct a distribution over r-bounded quasipartitions of M. We will then show that the distribution produced by the algorithm is $O(1)$-Lipschitz.

Algorithm 1 Random quasipartition of a tree quasimetric space

Input: A tree quasimetric space $M = (V(T), d_M)$ and $r > 0$.
Output: An r-bounded probabilistic quasipartition R.

Step 1. Set $R = \emptyset$. Add (u, v) to R for all $(u, v) \in E(M)$.
Step 2. Pick a root vertex t in M arbitrarily.
Step 3. Pick $z \in [0, r/2]$ uniformly at random.
Step 4. For all $(u, v) \in E(M)$ remove (u, v) from R if at least one of the following holds:
  (a) $d_M(u, t) > z + i\frac{r}{2}$ and $d_M(v, t) \leq z + i\frac{r}{2}$ for any integer $i > 0$.
  (b) $d_M(t, v) > z + i\frac{r}{2}$ and $d_M(t, u) \leq z + i\frac{r}{2}$ for any integer $i > 0$.
Step 5. Enforce transitivity on R: For all $u, v, w \in V(M)$, if $(u, v) \in R$ and $(v, w) \in R$, then add (u, w) to R.

We shall now prove some properties of the random quasipartition R in the following Lemmas. We will use these to prove Theorem 1.

▶ **Lemma 2.** *Let $u, v \in V(T)$. Let D be the path from u to v in T. Then either u is in the path from t to v, or v is in the path from u to t, or there exists a vertex w on D such that w lies on the path from u to t and w lies on the path from t to v.*

Proof. Let w be the nearest common ancestor of u and v in T. If $w = u$, then u is in the path from t to v; if $w = v$, then v is in the path from t to u; if $w \neq u$ and $w \neq v$, then w is in the path from t to u and in the path from t to v. ◀

▶ **Lemma 3.** *For any $u, v \in V(T)$ where u is in the path from t to v, let $P = \{a_1 = u, a_2, \ldots, a_m = v\}$ be the path from u to v. If $(a_i, a_{i+1}) \in R$ for all $i \in \{1, \ldots, m-1\}$ after Step 4 then $d_M(u, v) \leq \frac{r}{2}$.*

Proof. Let j be the largest integer such that $d_M(t, a_1) > z + j\frac{r}{2}$. By the choice of j it must be that $z + j\frac{r}{2} < d_M(t, a_1) \leq z + (j+1)\frac{r}{2}$. Since (a_i, a_{i+1}) is not removed from R in Step 4 of the algorithm it must be that $z + j\frac{r}{2} < d_M(t, a_i) \leq z + (j+1)\frac{r}{2}$ for all $i \in \{1, \ldots, m\}$. This implies that $d_M(t, v) \leq z + (j+1)\frac{r}{2} \leq d_M(t, u) + \frac{r}{2}$. Since $d_M(t, v) = d_M(t, u) + d_M(u, v)$, we have that $d_M(u, v) \leq \frac{r}{2}$, which concludes the proof. ◀

▶ **Lemma 4.** *For any $u, v \in V(M)$ where v is in the path from u to t, let $P = \{a_1 = u, a_2, \ldots, a_m = v\}$ be the path from u to v. If $(a_i, a_{i+1}) \in R$ for all $i \in \{1, \ldots, m-1\}$ after Step 4 then $d_M(u, v) \leq r/2$.*

Proof. The proof is similar to the proof of Lemma 3. ◀

▶ **Lemma 5.** *If $(u, v) \in R$ then $d_M(u, v) \leq r$.*

Proof. The fact that $(u, v) \in R$ implies that at the beginning of Step 4 there must have been a path $P = \{a_1 = u, a_2, \ldots, a_m = v\}$ from u to v such that $(a_i, a_{i+1}) \in R$ for all $i \in \{1, \ldots, m-1\}$. Since M is a tree quasimetric space, the shortest path is the single unique path from u to v for any $u, v \in V(T)$. From Lemma 2 we have that one of the following three cases is true:

Case 1: u is in the shortest path from t to v. We have $d_M(u, v) \leq r/2$ from Lemma 3.

Case 2: v is in the shortest path from u to t. We have $d_M(u, v) \leq r/2$ from Lemma 4.

Case 3: There exists a_j that lies on the shortest path from u to t and on the shortest path from t to v. From Lemmas 3 and 4 we have that $d_M(u, a_j) \leq r/2$ and $d_M(a_j, v) \leq r/2$. By the triangle inequality we get $d_M(u, v) \leq r$. ◀

▶ **Lemma 6.** *Any $(u, v) \in E(T)$ is removed with probability at most $2d_M(u, v)/r$ in Step 4 of the algorithm.*

Proof. Since M is a tree quasimetric space there are exactly two cases:

Case 1: The edge (u, v) is in the direction away from t which implies that $d_M(t, u) \leq d_M(t, v)$. Let i be the largest integer such that $i \cdot r/2 \leq d_M(t, u)$. The edge (u, v) is removed from R if z is chosen between $d_M(t, u) - i \cdot r/2$ and $d_M(t, v) - i \cdot r/2$. The probability of that event is bounded by $\int_{d_M(t,u)-i\frac{r}{2}}^{d_M(t,v)-i\frac{r}{2}} p(z)dz = \frac{2}{r}(d_M(t, v) - d_M(t, u)) \leq 2d_M(u, v)/r$ by the triangle inequality.

Case 2: The edge (u, v) is in the direction toward t which implies that $d_M(v, t) \leq d_M(u, t)$. Let i be the largest integer such that $i \cdot r/2 \leq d_M(v, t)$. (u, v) is removed from R if z is chosen between $d_M(v, t) - i \cdot r/2$ and $d_M(u, t) - i \cdot r/2$. The probability of that event is bounded by $\int_{d_M(v,t)-i\cdot r/2}^{d_M(u,t)-i\cdot r/2} p(z)dz = \frac{2}{r}(d_M(u, t) - d_M(v, t)) \leq 2d_M(u, v)/r$ by the triangle inequality. ◀

▶ **Lemma 7.** $\Pr[(u, v) \notin R] \leq 2d_M(u, v)/r$ *for all $u, v \in V(T)$.*

Proof. Let the unique path from u to v in M be $p = \{x_1 = u, x_2, \ldots, x_h = v\}$. Let X_p be the event that path p contains at least one edge (x_i, x_{i+1}) such that $(x_i, x_{i+1}) \notin R$ at the beginning of Step 5. Let $Y_{(a,b)}$ be the event that $(a, b) \notin R$ for $(a, b) \in E(G)$. We have $\Pr[X_p] = \Pr[Y_{(x_1,x_2)} \vee \ldots \vee Y_{(x_{h-1},x_h)}]$. From Lemma 6 and the union bound we have that $\Pr[Y_{(x_1,x_2)} \vee \ldots \vee Y_{(x_{h-1},x_h)}] \leq \Pr[Y_{(x_1,x_2)}] + \ldots + \Pr[Y_{(x_{h-1},x_h)}] \leq 2d_M(x_1, x_2)/r + \ldots + 2d_M(x_{h-1}, x_h)/r = 2d_M(u, v)/r$, concluding the proof. ◀

We are now ready to prove the main result of this Section.

Proof of Theorem 1. It follows by Lemmas 5 and 7 that the algorithm outputs an $O(1)$-Lipschitz distribution over r-bounded quasipartitions of M. ◀

3 Quasipartitions for graphs of small treewidth

In this section we prove the existence of a $O(t \log n)$-Lipschitz distribution over r-bounded quasipartitions for any quasimetric supported on a directed graph of treewidth t. The main result is summarized in the following.

▶ **Theorem 8.** *Let G be a n-vertex directed graph of treewidth t. Let M be the shortest-path quasimetric space induced by G. Then for any $r > 0$, there exists an $O(t \log n)$-Lipschitz distribution over r-bounded quasipartitions of M.*

In the proof of the above theorem we use the following proposition which is immediate from the definition of treewidth.

▶ **Proposition 9.** *Any graph G of treewidth t has a set of vertices $K \subseteq V(G)$ where $|K| \leq t$ such that removing K gives connected components each of which contains at most $\frac{|V(G)|}{2}$ vertices.*

First we introduce an algorithm to construct the required distribution over r-bounded quasipartitions of M. Steps 2 to 4 of the algorithm are recursive. At each recursive call the algorithm works on an associated sub-graph G^* and a global set R which is common to all recursive calls.

Algorithm 2 Random quasipartition of a bounded treewidth graph

Input: A digraph G of treewidth t, and $r > 0$.
Output: A random r-bounded quasipartition R.

Initialization: Set $G^* = G$ and $R = E(G)$. Perform the following recursive algorithm on G^*.

Step 1. Pick $z \in [0, r/2]$ uniformly at random.

Step 2. If $|V(G^*)| \leq 1$, terminate the current recursive call. Otherwise pick a set of vertices $K \subseteq V(G^*)$ such that $|K| \leq t$ and removing K from G^* gives connected components $C_1, \ldots, C_m$, each containing at most $\frac{|V(G^*)|}{2}$ vertices. This is possible by Proposition 9.

Step 3. For all $(u, v) \in E(G^*)$ remove (u, v) from R if one of the following holds:

(a) $d_G(u, x) > z$ and $d_G(v, x) \leq z$ for some vertex $x \in K$.
(b) $d_G(x, v) > z$ and $d_G(x, u) \leq z$ for some vertex $x \in K$.

Step 4. Recursively call Steps 2-4 on the vertex-induced subgraphs $G^*[C_1], \ldots, G^*[C_m]$.

Step 5. Once all branches of the recursion terminate enforce transitivity on R: For all $u, v, w \in V(G)$ if $(u, v) \in R$ and $(v, w) \in R$ add (u, w) to R.

Next we state some properties of the resulting random quasipartition R. We will use these to prove the main theorem. The proofs of all these Lemmas are given in the full version of the paper.

▶ **Lemma 10.** *If $(u, v) \in R$ then $d_G(u, v) \leq r$.*

▶ **Lemma 11.** *The depth of the recursion is $O(\log n)$.*

▶ **Lemma 12.** *Any $(u,v) \in E(G)$ is removed with probability at most $4t\frac{d(u,v)}{r}$ in Step 4 of the algorithm.*

▶ **Lemma 13.** $\Pr[(u,v) \notin R] \leq 4t \log n \frac{d_G(u,v)}{r}$ *for all $u, v \in V(G)$.*

With these Lemmas we can now prove the main result of this section.

Proof of Theorem 8. It follows by Lemmas 10 and 13 that the Algorithm outputs an $O(t \log n)$-Lipschitz distribution over r-bounded quasipartitions of M. ◀

The above algorithm can be implemented in polynomial time with an additional $O(\sqrt{\log t})$ loss on the quality of the partition. This is summarized in the following Theorem.

▶ **Theorem 14.** *Let M be the shortest-path quasimetric space induced by a n-vertex directed graph G of treewidth t. Then, there exists an algorithm with running time polynomial in n and t that computes the set of all quasipartitions in the support of an $O(t\sqrt{\log t} \log n)$-Lipschitz distribution over r-bounded quasipartitions of M, for any $r > 0$.*

Proof. The randomized algorithm described in Theorem 8 can be derandomized to yield a polynomial time algorithm. First we note that it is possible to find in polynomial time a set K, with $|K| = O(t\sqrt{\log t})$ such that removing K from G gives connected components containing at most $\frac{2|V(G)|}{3}$ vertices [11]. We can use this in Step 2 of the algorithm. The only random decision in the algorithm is in Step 1 when z is chosen. Given $r > 0$, we can instead select z exhaustively from all values in the set $S = \{d(u,v) : u, v \in V(G) \text{ and } d(u,v) \leq \frac{r}{2}\}$. It can be observed from Step 3 of the algorithm that picking any other value of z does not produce a new non-trivial r-bounded quasipartition. Since there are less than n^2 elements in S this derandomized version of the algorithm runs in polynomial time in n and the set of quasipartitions returned has less than n^2 elements. ◀

4 Embeddings into quasiultrametrics and into convex combinations of 0-1 quasimetrics

In this Section we present our results on random embeddings into quasiultrametric spaces, and deterministic embeddings into convex combinations of 0-1 quasimetric spaces (quasimetrics where all distances are either 0 or 1).

4.1 Upper bounds

We begin by establishing a relationship between quasipartitions and embeddings of quasimetric spaces into quasiultrametric spaces and 0-1 quasimetric spaces. We say that a distribution over quasipartitions $\mathcal{D}$ is ϵ-*forcing* if whenever $u, v \in X$ are such that $d(u,v) \leq \epsilon r$ then $\Pr_{P \sim \mathcal{D}}[(u,v) \notin P] = 0$. First we state a result, inspired by [4], that we use in subsequent proofs. Its proof can be found in the full version of the paper.

▶ **Lemma 15.** *Let G be a directed graph on n vertices. Let M_W denote the shortest-path quasimetric space induced by G where edge weights are specified by a function $W : E(G) \to \mathbb{R}^+$. Suppose that for all $r > 0$ there exists a β-Lipschitz distribution over r-bounded quasipartitions of M_W. Then for all $r > 0$ there exists a 2β-Lipschitz $\frac{1}{2n}$-forcing distribution over r-bounded quasipartitions of M_W.*

Now we present methods to use quasipartitions for constructing embeddings of quasimetric spaces into quasiultrametric spaces and 0-1 quasimetric spaces. The proof resembles the argument used in [4] for computing random embeddings of a metric space into a tree.

▶ **Theorem 16.** *Let $M = (X, d)$ be an n-point quasimetric space and let $\beta > 0$. Suppose that for any $r > 0$, there exists a β-Lipschitz distribution over r-bounded quasipartitions of M. Then M admits a random embedding into quasiultrametrics with distortion $O(\beta \log n)$.*

Proof. We may assume w.l.o.g. that the minimum distance in M is 1 and the diameter is Δ. By Lemma 15 it follows that for any $r > 0$ there exists a 2β-Lipschitz $\frac{1}{2n}$-forcing distribution $\mathcal{D}_r$ over r-bounded quasipartitions of M. To get the required embedding we combine a series of quasipartitions. Let $S = \{P_0, P_1, \ldots, P_{\lfloor \log \Delta \rfloor}\}$ where $P_i \in \mathcal{D}_{2^i}$ is a randomly chosen 2^i-bounded quasipartition from $\mathcal{D}_{2^i}$.

We combine the quasipartitions as follows to get a quasiultrametric M^*:

Step 1: Set $d_{M^*}(u, v) = 2^{\lfloor \log \Delta \rfloor + 1}$ for all $u, v \in V(M)$. Set $i = \lfloor \log \Delta \rfloor$.

Step 2: Set $d_{M^*}(u, v) = 2^i$ for all $(u, v) \in P_i$ if $d_{M^*}(u, v) = 2^{i+1}$. Decrease i by 1. Repeat step 2 if $i \geq 0$.

We first argue that M^* is a quasiultrametric. To that end, consider any $u, v \in X$. Let j be the maximum value of i such that $(u, v) \notin P_i$. This implies that $d^*_M(u, v) = 2^{i+1}$. Consider any $w \in X$. It must be that either $(u, w) \notin P_j$ or $(w, v) \notin P_j$ because $(u, v) \notin P_j$. This implies that either $d^*_M(u, w) = 2^{i+1}$ or $d^*_M(w, v) = 2^{i+1}$. So, for any $u, v, w \in X$ it must be that $d^*_M(u, v) \leq \max\{d^*_M(u, w), d^*_M(w, v)\}$. This establishes that M^* is a quasiultrametric.

Next we argue that M^* is non-contracting. Let us suppose that the claim is false and that M^* is contracting. This means that for some $u, v \in X$ we have $d_{M^*}(u, v) < d_M(u, v)$. This means that in some iteration of Step 2 we set $d_{M^*}(u, v) = 2^i$ for some $i < \log d_M(u, v)$. This implies that $(u, v) \in P_i$ even though $d_M(u, v) > 2^i$, which is a contradiction.

It remains to show that M^* has expansion $O(\beta \log n)$. Let $u, v \in X$. We have

$$\begin{aligned}
\mathbb{E}\left[\frac{d_{M^*}(u,v)}{d_M(u,v)}\right] &\leq \sum_{i=0}^{\lfloor \log \Delta \rfloor} \Pr[(u,v) \notin P_i] \frac{2^{i+1}}{d_M(u,v)} \\
&\leq \sum_{i=0}^{\lfloor \log d_M(u,v) \rfloor} \frac{2^{i+1}}{d_M(u,v)} + \sum_{i=\lfloor \log d_M(u,v) \rfloor + 1}^{\lfloor \log (2n d_M(u,v)) \rfloor} 2\beta \frac{d_M(u,v)}{2^i} \frac{2^{i+1}}{d_M(u,v)} \\
&\quad + \sum_{i=\lfloor \log (2n d_M(u,v)) \rfloor + 1}^{\lfloor \log \Delta \rfloor]} 0 \cdot \frac{2^{i+1}}{d_M(u,v)} \\
&\leq \frac{4 d_M(u,v)}{d_M(u,v)} + 4\beta \log n = O(\beta \log n),
\end{aligned}$$

concluding the proof. ◀

▶ **Theorem 17.** *Let $M = (X, d)$ be an n-point quasimetric space and let $\beta > 0$. Suppose that for any $r > 0$, there exists a β-Lipschitz distribution over r-bounded quasipartitions of M. Then M admits an embedding into a convex combination of 0-1 quasimetric spaces with distortion $O(\beta \log n)$.*

Proof. We may assume w.l.o.g. that the minimum distance in M is 1 and the diameter is Δ. By Lemma 15 it follows that for any $r > 0$ there exists $\mathcal{D}_r$ a β-Lipschitz $\frac{1}{2n}$-forcing distribution over r-bounded quasipartitions of M. Let $S = \{\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_{\lfloor \log \Delta \rfloor}\}$. Let $c = \sum_{i \in [0, \lfloor \log \Delta \rfloor]} 2^{i+1}$. Let H be a discrete distribution over S where the probability density function F is given by $F(\mathcal{D}_i) = \frac{2^{i+1}}{c}$.

Let us define Y to be the event that a random quasipartition is selected from the distribution $\mathcal{D}_i$ where $\mathcal{D}_i$ is randomly chosen from the distribution H. This gives a distribution over a set of quasipartitions. We can replace every quasipartition P in this set by a 0-1

quasimetric space $Q = (X, d_P)$ where for all $(u, v) \in P$ we have that $d_P(u, v) = 0$ and for all $(u, v) \notin P$ we have that $d_P(u, v) = 1$. This gives a distribution over a set of 0-1 quasimetric spaces which can be interpreted as a convex combination of 0-1 quasimetric spaces. Let the quasimetric space given by this convex combination of 0-1 quasimetric spaces be $\phi = (X, d_\phi)$. We will now show that the distortion is bounded for this embedding. First we claim that for all $u, v \in X$, $d_\phi(u, v) \geq \frac{d(u,v)}{c}$. This can be shown as follows. We have that

$$d_\phi(u,v) = \sum_{i=0}^{\lfloor \log \Delta \rfloor} \frac{2^{i+1}}{c} \Pr_{P \sim \mathcal{D}_i}[(u,v) \notin P] \geq \sum_{i=0}^{\lfloor \log d(u,v) \rfloor} \frac{2^{i+1}}{c} \geq \frac{d(u,v)}{c},$$

which proves the claim. Next we show that for all $x, y \in X$, $d_\phi(x, y) \leq O(\beta \log n)\frac{d(x,y)}{c}$. We have

$$\begin{aligned} d_\phi(u,v) &= \sum_{i=0}^{\lfloor \log \Delta \rfloor} \frac{2^{i+1}}{c} \Pr_{P \sim \mathcal{D}_i}[(u,v) \notin P] \\ &\leq \sum_{i=0}^{\lfloor \log d(u,v) \rfloor} \frac{2^{i+1}}{c} + \sum_{i=\lfloor \log d(u,v) \rfloor + 1}^{\lfloor \log (2nd(u,v)) \rfloor} \beta \frac{d(u,v)}{2^i} \frac{2^{i+1}}{c} + \sum_{i=\lfloor \log (2nd(u,v)) \rfloor + 1}^{\lfloor \log \Delta \rfloor} 0 \cdot \frac{2^{i+1}}{c} \\ &\leq 4\frac{d(x,y)}{c} + 2\beta \log n \frac{d(x,y)}{c} \leq O(\beta \log n)\frac{d(x,y)}{c}. \end{aligned}$$

From the above lower and upper bounds we get that ϕ is an embedding of M with distortion $O(\beta \log n)$. This concludes the proof of the theorem. ◀

We get the following Corollaries by combining the above Theorems with the main result of Section 3.

▶ **Corollary 18.** *Let $M = (X, d)$ be the shortest path quasimetric space induced by a directed graph on n vertices having treewidth t. Then M admits a random embedding into a quasiultrametric space with distortion $O(t \log^2 n)$. Moreover there exists an algorithm with running time polynomial in n and t, that samples a random embedding into a quasiultrametric space with distortion $O(t\sqrt{\log t} \log^2 n)$.*

Proof. The existential part follows from Lemma 15 and Theorems 8 and 16. The computational part uses Theorem 14. ◀

▶ **Corollary 19.** *Let $M = (X, d)$ be the shortest path quasimetric space induced by a directed graph on n vertices having treewidth t. Then M admits an embedding into a convex combination of 0-1 quasimetric spaces with distortion $O(t \log^2(n))$. Moreover, there exists an algorithm with running time polynomial in n and t, that computes an embedding into a convex combination of 0-1 quasimetric spaces with distortion $O(t\sqrt{\log t} \log^2 n)$.*

Proof. This follows from Lemma 15 and Theorem 8 and 17. The computational part uses Theorem 14. ◀

4.2 Lower bounds

We now obtain a lower bound on the distortion of random embeddings of general quasimetric spaces into quasiultrametric spaces. To this end we show that if a quasimetric space admits a random embedding into quasiultrametric spaces, then it is possible to construct a Lipschitz distribution over r-bounded quasipartitions of the quasimetric space for any $r > 0$.

We also show that subdivided directed acyclic graphs cannot embed into a graph with bounded distortion unless there is a minor of the original graph present in the embedding.

▶ **Theorem 20.** *Let $M = (X, d)$ be a quasimetric space and let $\beta > 0$. Suppose that M admits a random embedding into quasiultrametric spaces $\mathcal{D}^*$ with distortion β. Then for any $r > 0$, there exists an β-Lipschitz distribution over r-bounded quasipartitions of M.*

Proof. We get a β-Lipschitz distribution $\mathcal{D}$ over r-bounded quasipartitions of M by modifying $\mathcal{D}^*$. For every quasiultrametric $M^* \in \mathcal{D}^*$ we add the r-bounded quasipartition $R(M^*)$ to $\mathcal{D}$ where $(u, v) \in R(M^*)$ iff $d_{M^*}(u, v) \leq r$. The probability of selecting any $R(M^*) \in \mathcal{D}$ is set to be equal to that of selecting the corresponding $M^* \in \mathcal{D}^*$.

First we claim that $R(M^*)$ is an r-bounded quasipartition of M. Consider any $u, v, w \in V(M)$. Suppose we have that $(u, v) \in R(M^*)$ and $(v, w) \in R(M^*)$ then it must be that $d_{M^*}(u, v) \leq r$ and $d_{M^*}(v, w) \leq r$. Since M^* is a quasiultrametric this implies that $d_{M^*}(u, w) \leq \max\{d_{M^*}(u, v), d_{M^*}(v, w)\} \leq r$. This means that $(u, w) \in R(M^*)$ which implies that $R(M^*)$ is transitive and is hence a quasipartition of M. Since we have $(u, v) \in R(M^*)$ iff $d_{M^*}(u, v) \leq r$ and $d_{M^*}(u, v) \geq d_M(u, v)$ it follows that $R(M^*)$ is an r-bounded quasipartitions of M.

Next we prove that $\mathcal{D}$ is β-Lipschitz. We have

$$\beta = \mathbb{E}\left[\frac{d_{M^*}(u,v)}{d_M(u,v)}\right] \geq \int_{d_M(u,v)}^{\infty} \left(\frac{z}{d_M(u,v)}\right) \Pr[d_{M^*}(u,v) = z]dz$$
$$\geq \left(\frac{r}{d_M(u,v)}\right) \Pr[d_{M^*}(u,v) > r].$$

This implies that $\Pr[d_{M^*}(u, v) > r] \leq \beta \cdot \frac{d_M(u,v)}{r}$. Since $\Pr[d_{M^*}(u, v) > r] = \Pr[(u, v) \notin R(M^*)]$, we have that $\mathcal{D}$ is β-Lipschitz and this concludes the proof of the theorem. ◀

The following Theorem follows directly from the work of Chuzhoy and Khanna [9] and a result (Theorem 4.2) of Charikar, Makarychev and Makarychev [8].

▶ **Theorem 21.** *There exists a quasimetric space M such that any embedding of M into a convex combination of 0-1 quasimetric spaces has distortion $\Omega\left(\frac{n^{1/7}}{\log^{4/7} n}\right)$.*

A lower bound on the flow-cut gap of Directed Sparsest-Cut does not directly give the same lower bound on the quality of Lipschitz quasipartitions of quasimetric spaces. The following Theorem follows from the work of Chuzhoy and Khanna [9]. We present its proof in the full version of the paper.

▶ **Theorem 22.** *There exists a quasimetric space M and a positive r such that any distribution over r-bounded quasipartitions of M is $\Omega\left(\frac{n^{1/7}}{\log^{4/7} n}\right)$-Lipschitz.*

Combining Theorems 20 and 22 gives the following Corollary.

▶ **Corollary 23.** *There exists an n-point quasimetric space $M = (X, d)$ such that any random embedding of M into quasiultrametric spaces has distortion $\Omega\left(\frac{n^{1/7}}{\log n^{4/7}}\right)$.*

The proof of the following theorem is given in the full version of the paper.

▶ **Theorem 24.** *Let $G = (E, V)$ be a directed acyclic graph. Let $G' = (E', V')$ be the graph obtained by subdividing each directed edge of G into three edges in the same direction, i.e., for any directed edge $(u, v) \in E$, E' contains the directed edges $(u, x_{uv}), (x_{uv}, y_{uv}), (y_{uv}, v)$ obtained by subdividing (u, v). Suppose that there exists an embedding of G' into some graph H with bounded distortion. Then H must contain G as a minor.*

5 Applications to directed cut problems

In this section we will use the results from the previous section, for embedding quasimetric spaces into convex combinations of 0-1 quasimetric spaces, to get approximation algorithms for some cut problems on directed graphs.

Directed Non-Bipartite Sparsest-Cut

Let G be a directed graph with non-negative capacities $c(e)$ for all $e \in E(G)$. Let T be a set of terminal vertex pairs $\{(s_1,t_1),(s_2,t_2),\ldots,(s_k,t_k)\}$. Let $\mathsf{dem}(i)$ be a non-negative demand for the terminal pair (s_i,t_i). A *cut* of G is a set of edges $S \subset E(G)$. We define the *capacity* of the cut S to be $c(S) = \sum_{e\in S} c(e)$. Let I_S be the set of all integers $i \in \{1,\ldots,k\}$ such that all paths from s_i to t_i have at least one edge in S. We define the demand separated by the cut S to be $\mathsf{dem}(S) = \sum_{i\in I_S} \mathsf{dem}(i)$. The *sparsity* of a cut S is defined to be $c(S)/\mathsf{dem}(S)$. The goal of the Directed Non-Bipartite Sparsest-Cut problem is to find the cut with minimum sparsity.

For non-uniform demands, there exists a $O(\sqrt{n})$-approximation by [12], which has been improved to $\tilde{O}(n^{11/23})$-approximation [2]. There is also a $2^{\Omega(\log^{1-\epsilon} n)}$-hardness due to Chuzhoy and Khanna [9].

Consider the following standard LP relaxation of the Directed Non-Bipartite Sparsest-Cut Problem on G.

$$\begin{aligned}
\min \quad & \sum_{e\in E(G)} c(e)x(e) \\
& \sum_{(s_i,t_i)\in T} \mathsf{dem}(i)d(s_i,t_i) \geq 1 \\
& x(e) \geq 0 \qquad \forall e \in E(G) \\
& d(u,v) \geq 0 \qquad \forall u,v \in V(G)
\end{aligned}$$

In the LP, the $x(e)$ values can be treated as distance assignments for the edges. The $d(u,v)$ values are the shortest path distances from u to v in G for edge weights defined by the distance assignments. Since d is the shortest path distance, it follows that for all $(u,v) \in E$ we have that $d(u,v) \leq x(e)$ where $e = (u,v)$. Since replacing $x(e)$ with $d(u,v)$ only reduces the objective function while preserving feasibility, we have that the optimal edge distance assignment of the LP is given by a shortest path quasimetric space on G.

Charikar, Makarychev and Makarychev [8] showed that the integrality gap of this LP is closely related to the minimum distortion achievable for embedding a quasimetric space into a convex combination of 0-1 quasimetric spaces. Theorem A.1 from their paper implies that the integrality gap of the LP for a graph G with edge capacities $c(e)$ is equal to the minimum distortion for embedding a shortest path quasimetric space supported on G into a convex combination of 0-1 quasimetric spaces (referred to as 0-1 semimetrics in their paper). We now describe how the minimum distortion embedding of a quasimetric space into a convex combination of 0-1 quasimetric spaces can be used to upper bound the integrality gap of the LP. First we observe that a 0-1 quasimetric space on $V(G)$ corresponds to a cut of G. This is because a 0-1 quasimetric space $M = (V(G), d^*)$ can be used to describe a cut $S \subset E(G)$ where $(u,v) \in S$ iff $d^*(u,v) = 1$ for all $(u,v) \in E(G)$. Let the optimal Sparsest-Cut value for the LP be LP_{OPT}. Let the shortest path quasimetric space on G that gives the optimal

solution for the LP be $M_{\mathsf{OPT}} = (V(G), d_{\mathsf{OPT}})$. We have that,

$$\frac{\sum_{e=(u,v)\in E(G)} c(e) d_{\mathsf{OPT}}(u,v)}{\sum_{i\in[0,k]} \mathsf{dem}(i) d_{\mathsf{OPT}}(s_i,t_i)} = LP_{\mathsf{OPT}}.$$

Let $\phi = \sum_{j\in[0,m]} \alpha_j M_j$ be an $O(\log^2(n))$ embedding of M_{OPT} into a convex combination of 0-1 quasimetric spaces, where $M_j = (V(G), d_j)$ is a 0-1 quasimetric space and α_j is a non-negative real number for any $j \in [0,m]$. We have that,

$$\begin{aligned} O(\log^2(n)) LP_{\mathsf{OPT}} &\geq \frac{\sum_{e=(u,v)\in E(G)} c(e) \sum_{j\in[0,m]} \alpha_j d_j(u,v)}{\sum_{i\in[0,k]} \mathsf{dem}(i) \sum_{j\in[0,m]} \alpha_j d_j(s_i,t_i)} \\ &\geq \frac{\sum_{j\in[0,m]} \alpha_j \sum_{e=(u,v)\in E(G)} c(e) d_j(u,v)}{\sum_{j\in[0,m]} \alpha_j \sum_{i\in[0,k]} \mathsf{dem}(i) d_j(s_i,t_i)} \\ &\geq \min_{j\in[0,m]} \frac{\sum_{e=(u,v)\in E(G)} c(e) d_j(u,v)}{\sum_{i\in[0,k]} \mathsf{dem}(i) d_j(s_i,t_i)}. \end{aligned}$$

Therefore the cut corresponding to the 0-1 quasimetric space in ϕ having minimum sparsity gives an integral solution to the LP that is at most a $O(\log^2(n))$ factor larger than LP_{OPT}. Combined with Corollary 19 and Theorem 14 this implies the following corollary.

▶ **Corollary 25.** *The integrality gap (which is also the flow-cut gap) of the Directed Non-Bipartite Sparsest-Cut LP relaxation on graphs of n vertices and treewidth t is $O(t \log^2 n)$. Moreover there exists a polynomial-time $O(t\sqrt{\log t} \log^2 n)$-approximation algorithm for the Directed Non-Bipartite Sparsest-Cut problem on such graphs with running time linear in t and polynomial in n. If a tree decomposition of width t is given as part of the input, then there exists a polynomial-time $O(t \log^2 n)$-approximation algorithm.*

Directed Multicut

Let G be a directed graph with non-negative capacities $c(e)$ for all $e \in E(G)$. Let T be a set of terminal vertex pairs $\{(s_1,t_1),(s_2,t_2),\ldots,(s_k,t_k)\}$. The goal of the Directed Multicut problem is to find the minimum capacity cut that separates all terminal vertex pairs.

There is a $\tilde{O}(n^{2/3}/\mathsf{OPT}^{1/3})$-approximation by Kortsarts, Kortsarz and Nutov [17], and a $\tilde{O}(n^{11/23})$-approximation by Agarwal, Alon and Charikar [2]. Finally, there is a $2^{\Omega(\log^{1-\epsilon} n)}$-hardness due to Chuzhoy and Khanna [9]. It is known that for all $\beta > 0$, if there exists a polynomial time β-approximation algorithm for Directed Non-Bipartite Sparsest-Cut then there also exists a polynomial time $O(\beta \log n)$-approximation for Directed Multicut. The details are given in the full version of the paper.

▶ **Corollary 26.** *There exist a $O(t\sqrt{\log t} \log^3 n)$-approximation algorithm for the Directed Multicut problem on n-vertex graphs of treewidth t, with running time polynomial in both n and t. If a tree decomposition of width t is given as part of the input, then there exists a polynomial-time $O(t \log^3 n)$-approximation algorithm.*

References

1 Ittai Abraham, Yair Bartal, and Ofer Neiman. Nearly tight low stretch spanning trees. *arXiv preprint arXiv:0808.2017*, 2008.

2 Amit Agarwal, Noga Alon, and Moses S Charikar. Improved approximation for directed cut problems. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 671–680. ACM, 2007.

3 Sanjeev Arora, James Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. *Journal of the American Mathematical Society*, 21(1):1–21, 2008.

4 Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 184–193. IEEE, 1996.

5 Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 161–168. ACM, 1998.

6 Glencora Borradaile, James R Lee, and Anastasios Sidiropoulos. Randomly removing g handles at once. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 371–376. ACM, 2009.

7 Gruia Calinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing*, 34(2):358–372, 2005.

8 Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Directed metrics and directed graph partitioning problems. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 51–60. Society for Industrial and Applied Mathematics, 2006.

9 Julia Chuzhoy and Sanjeev Khanna. Polynomial flow-cut gaps and hardness of directed cut problems. *Journal of the ACM (JACM)*, 56(2):6, 2009.

10 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 448–455. ACM, 2003.

11 Uriel Feige, MohammadTaghi Hajiaghayi, and James R Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008.

12 Mohammad Taghi Hajiaghayi and Harald Räcke. An-approximation algorithm for directed sparsest cut. *Information Processing Letters*, 97(4):156–160, 2006.

13 Piotr Indyk and Jiri Matoušek. Low-distortion embeddings of finite metric spaces. *Handbook of Discrete and Computational Geometry*, page 177, 2004.

14 Piotr Indyk and Anastasios Sidiropoulos. Probabilistic embeddings of bounded genus graphs into planar graphs. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 204–209. ACM, 2007.

15 Jonathan A Kelner, James R Lee, Gregory N Price, and Shang-Hua Teng. Metric uniformization and spectral bounds for graphs. *Geometric and Functional Analysis*, 21(5):1117–1143, 2011.

16 Philip Klein, Serge A Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 682–690. ACM, 1993.

17 Yana Kortsarts, Guy Kortsarz, and Zeev Nutov. Greedy approximation algorithms for directed multicuts. *Networks*, 45(4):214–217, 2005.

18 Robert Krauthgamer, James R Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. *Geometric & Functional Analysis GAFA*, 15(4):839–858, 2005.

19 James R Lee and Assaf Naor. Extending lipschitz functions via random metric partitions. *Inventiones mathematicae*, 160(1):59–95, 2005.

20 James R Lee and Anastasios Sidiropoulos. On the geometry of graphs with a forbidden minor. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 245–254. ACM, 2009.

21 James R Lee and Anastasios Sidiropoulos. Genus and the geometry of the cut graph. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 193–201. Society for Industrial and Applied Mathematics, 2010.

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

23 Anastasios Sidiropoulos. Optimal stochastic planarization. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 163–170. IEEE, 2010.

The Landscape of Communication Complexity Classes*

Mika Göös[1], Toniann Pitassi[2], and Thomas Watson[3]

1 **University of Toronto, Computer Science Department, Toronto, ON, Canada**
mika.goos@mail.utoronto.ca
2 **University of Toronto, Computer Science Department, Toronto, ON, Canada**
toni@cs.toronto.edu
3 **University of Toronto, Computer Science Department, Toronto, ON, Canada**
thomasw@cs.toronto.edu

Abstract

We prove several results which, together with prior work, provide a nearly-complete picture of the relationships among classical communication complexity classes between P and PSPACE, short of proving lower bounds against classes for which no explicit lower bounds were already known. Our article also serves as an up-to-date survey on the state of structural communication complexity.

Among our new results we show that $\mathsf{MA} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}$, that is, Merlin–Arthur proof systems cannot be simulated by zero-sided error randomized protocols with one NP query. Here the class $\mathsf{ZPP}^{\mathsf{NP}[1]}$ has the property that generalizing it in the slightest ways would make it contain $\mathsf{AM} \cap \mathsf{coAM}$, for which it is notoriously open to prove any explicit lower bounds. We also prove that $\mathsf{US} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}$, where US is the class whose canonically complete problem is the variant of set-disjointness where yes-instances are uniquely intersecting. We also prove that $\mathsf{US} \not\subseteq \mathsf{coDP}$, where DP is the class of differences of two NP sets. Finally, we explore an intriguing open issue: are rank-1 matrices inherently more powerful than rectangles in communication complexity? We prove a new separation concerning PP that sheds light on this issue and strengthens some previously known separations.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases Landscape, communication, complexity, classes

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.86

1 Introduction

Complexity classes form the infrastructure of classical complexity theory. They are used to express the power of models of computation, characterize the complexities of important computational problems, and catalyze proofs of other results. A central project is to ascertain the full, intricate landscape of relationships among complexity classes.

Beginning with [3], there has been a lot of research on the analogues of classical (Turing machine) complexity classes in two-party communication complexity. The analogue of P (the class of decision problems solvable in polynomial time) is the class of functions $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ for which Alice and Bob, given x and y respectively, can evaluate $F(x,y)$ with a protocol that uses polylogarithmically many bits of communication. For other classical complexity classes representing other models of computation, one can generally define,

* The full version is available at http://eccc.hpi-web.de/report/2015/049/. This work was supported by NSERC funding.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 86; pp. 86:1–86:15

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

in a canonical way, associated communication complexity classes representing associated models of communication. There are many motivations for studying the relationships (inclusions and non-inclusions) between these communication complexity classes.

- A holy grail of classical complexity is to prove separations of classes between P and PSPACE. Separations relative to oracles can often be viewed as class separations in the restricted setting of *query complexity*; see [52] for an excellent survey. Communication complexity can be viewed as a restricted (but generally less restricted than query complexity) setting for which lower bounds are more difficult to obtain. Such separations in restricted settings are sometimes construed as evidence for the classical separations, or at least as barriers to refuting the classical separations. A stronger form of relativization barriers is known as *algebrization* [2], which directly employs communication complexity class separations.
- Proving lower bounds against strong communication complexity classes has applications to other areas of theoretical computer science. One of the most notorious open problems in communication complexity is to prove lower bounds against the analogue of the polynomial hierarchy (PH) for any explicit two-party function. Proving PH lower bounds is a necessary step for obtaining strong rank rigidity lower bounds [44, 36, 37, 53] (as well as margin complexity rigidity lower bounds [35]), which in turn are related to circuit complexity [50]. Lower bounds against PH are also related to graph complexity [42, 25]. It even remains open to prove communication lower bounds against the subclass of PH known as AM (Arthur–Merlin games) for any explicit function (which would be relevant to streaming delegation [8, 31, 19, 7, 9, 32]).
- Communication complexity has a menagerie of techniques for proving lower bounds (among the oldest being discrepancy and corruption). These techniques often provide lower bounds against powerful communication complexity classes, and in some cases turn out to be *equivalent* to the communication measures corresponding to those classes (e.g., discrepancy is equivalent to PP communication [29], and corruption is equivalent to SBP communication [18]). See [17] for more background on this. Thus, by studying complexity classes, as a byproduct we study the relative strength of lower bound techniques.
- The various models of communication corresponding to complexity classes are mathematically interesting because protocols in these models can be viewed as succinct representations of boolean matrices. The study of classes exposes natural questions about the combinatorial power of such succinct representations.

We contribute to the exploration of the communication complexity landscape by filling in many of the remaining gaps in the known relationships among classes, and discovering new techniques and insights along the way. In Section 2 we state our results more precisely and provide some intuition for the proofs. In the full version, we summarize the state of affairs (including our new results) by showing a map of known inclusions and non-inclusions between pairs of traditional communication classes, and we provide a comprehensive survey of these results. This updates previous surveys by Babai, Frankl, and Simon [3] and Halstenberg and Reischuk [21].

We refer to [33, 26] for background on communication complexity. In the full version we provide a catalog of communication complexity class definitions; throughout the text, we provide definitions on a "need-to-know" basis. If $\mathcal{C}$ is the name of a model (e.g., P for deterministic or NP for nondeterministic), we follow the convention of using $\mathcal{C}$ to denote both a complexity class and the corresponding complexity measure: $\mathcal{C}(F)$ denotes the minimum cost of a correct protocol for the (possibly partial) two-party function F in model $\mathcal{C}$, and $\mathcal{C}$ denotes the class of all (families of) partial functions F with $\mathcal{C}(F) \leq \mathrm{poly}(\log n)$.

2 Our Contributions

Several of our results concern two-party composed functions, so we introduce some general notation for this. A *composed function* is of the form $f \circ g^m$ where $f\colon \{0,1\}^m \to \{0,1\}$ is a (possibly partial) *outer function* and $g\colon \{0,1\}^b \times \{0,1\}^b \to \{0,1\}$ is an *inner function* also called a *gadget*. We write $F := f \circ g^m\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ where $n := m \cdot b$. We view the inputs to Alice and Bob as $x, y \in (\{0,1\}^b)^m$, which are partitioned into *blocks* $x_i, y_i \in \{0,1\}^b$ for $i \in [m]$. The goal is to compute $F(x,y) := f(g(x_1,y_1),\ldots,g(x_m,y_m))$.

2.1 MA $\not\subseteq$ ZPP$^{\mathsf{NP}[1]}$

A Merlin–Arthur (MA) communication protocol is a proof system in which a nondeterministic party called Merlin sends a proof string (depending on the input) to Alice and Bob (collectively constituting Arthur), who then execute a randomized protocol to verify the proof. Merlin–Arthur communication protocols have been studied many times [28, 43, 2, 16, 30, 19, 20], starting with the work of Klauck [28], who gave a $\Omega(\sqrt{n})$ lower bound on the MA communication complexity of set-disjointness. In contrast, for the related (and stronger) model of Arthur–Merlin (AM) communication protocols, in which Merlin's proof string may depend on Alice's and Bob's randomness, no nontrivial lower bound is known for any explicit function, and such lower bounds have become very sought-after in the recent literature [35, 40, 32, 9].

Our first result concerns the relationship between MA and another class, ZPP$^{\mathsf{NP}[1]}$, which is a slightly obscure but intriguing character with many curious properties. A ZPP-type protocol is randomized and may output the correct answer or $\perp$ (representing "don't know"), and must output the correct answer with high probability on every input; granting the protocol access to one query to an NP oracle yields ZPP$^{\mathsf{NP}[1]}$. It is not a priori clear that the model is robust with respect to the choice of threshold for the success probability, since standard amplification by repetition would increase the number of NP oracle queries. However, it was shown in [11] that ZPP$^{\mathsf{NP}[1]}$ does indeed admit efficient amplification as long as the success probability is $> 1/2$ (the proof for time-bounded complexity also works for communication complexity); hence we define the model with success probability some constant $> 1/2$, say $3/4$.

If we allowed ZPP$^{\mathsf{NP}[1]}$ to have success probability $< 1/2$, the class would change drastically: it would contain AM $\cap$ coAM (see the full version), and hence proving explicit lower bounds for the communication version would yield breakthrough AM communication lower bounds. Granting the model access to two nonadaptive NP queries (and requiring success probability $> 1/2$) would also encompass AM $\cap$ coAM. Thus, in a sense, ZPP$^{\mathsf{NP}[1]}$ represents a boundary beyond which AM lower bounds would be the next step. The class ZPP$^{\mathsf{NP}[1]}$ is also sandwiched between BPP and $\mathsf{S}_2\mathsf{P}$ [6]; $\mathsf{S}_2\mathsf{P}$ is a subclass of the polynomial hierarchy that has not been studied before in communication complexity (the definition appears in the full version), and no nontrivial lower bounds against it are known for any explicit function. This is another sense in which ZPP$^{\mathsf{NP}[1]}$ constitutes a new frontier toward the elusive goal of proving explicit PH communication lower bounds. We also mention that ZPP$^{\mathsf{NP}[1]}$ shows up frequently in the literature on the "two queries problem" (e.g., if $\mathsf{P}_{\|}^{\mathsf{NP}[2]} \subseteq$ ZPP$^{\mathsf{NP}[1]}$ then PH $= \mathsf{S}_2\mathsf{P}$ [49]).

We prove that MA $\not\subseteq$ ZPP$^{\mathsf{NP}[1]}$ in the setting of communication complexity. This can be interpreted as saying that one-round non-interactive[1] proof systems cannot be made to have

[1] Here, the term non-interactive means that Alice and Bob cannot interact with Merlin other than receiving the proof string.

zero-sided error, even if the proof is generalized to an NP oracle query that depends on the randomness.

Before officially stating the theorem, we give the relevant formal definitions. An MA communication protocol computing $F\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ consists of a randomized two-party protocol which takes as input, in addition to the usual inputs x and y, a proof string (witness) $w \in \{0,1\}^k$ that is visible to both Alice and Bob. The completeness criterion is that for every $(x,y) \in F^{-1}(1)$ there exists a w such that the protocol accepts with probability at least $3/4$, and the soundness criterion is that for every $(x,y) \in F^{-1}(0)$ and every w, the protocol rejects with probability at least $3/4$. The cost is the witness length k plus the length of the subsequent transcript between Alice and Bob.

A $\mathsf{ZPP}^{\mathsf{NP}[1]}$ protocol Π computing F is a distribution over $\mathsf{P}^{\mathsf{NP}[1]}$-type protocols, each of which is of the following form: There is a deterministic protocol where for each leaf v having associated rectangle R_v, there is also an associated collection of "witness rectangles" $\left\{S_{v,w} \subseteq R_v \ : \ w \in \{0,1\}^k\right\}$ and an associated "output function" $o_v\colon \{0,1\} \to \{0,1,\perp\}$. The output of the $\mathsf{P}^{\mathsf{NP}[1]}$-type protocol on input (x,y) is obtained by running the deterministic part to reach a leaf v, then applying o_v to the indicator of whether $(x,y) \in \bigcup_w S_{v,w}$. The correctness criterion is that for every $(x,y) \in F^{-1}$, $\mathbb{P}\big[\Pi(x,y) \in \{F(x,y),\perp\}\big] = 1$ and $\mathbb{P}\big[\Pi(x,y) = F(x,y)\big] \geq 3/4$. The cost is the witness length k plus the maximum communication cost of the deterministic part of any of the constituent $\mathsf{P}^{\mathsf{NP}[1]}$-type protocols. The result of [11] shows that changing the success probability from $3/4$ to any other constant strictly between $1/2$ and 1 would only change the measure $\mathsf{ZPP}^{\mathsf{NP}[1]}(F)$ by a constant factor.

We prove a lower bound for the block-equality function Block-Eq, defined as follows:[2] Given $\sqrt{n}$ instances of the equality function Eq of length $\sqrt{n}$, is at least one of them a yes-instance? More formally, we have Block-Eq $\coloneqq$ Or $\circ$ Eq^m where the input to Or is $m \coloneqq \sqrt{n}$ bits, and each input to Eq is $b \coloneqq \sqrt{n}$ bits. In other words, writing $x \coloneqq x_1 \cdots x_{\sqrt{n}} \in (\{0,1\}^{\sqrt{n}})^{\sqrt{n}}$ and $y \coloneqq y_1 \cdots y_{\sqrt{n}} \in (\{0,1\}^{\sqrt{n}})^{\sqrt{n}}$, we have Block-Eq$(x,y) = 1$ iff $x_i = y_i$ for some i. Note that Block-Eq $\in \mathsf{MA}$ since i can be nondeterministically guessed by Merlin, and then $x_i = y_i$ can be verified using a randomized protocol for Eq. (It was first noticed in [34] that Block-Eq $\in \Sigma_2\mathsf{P} \cap \Pi_2\mathsf{P}$, which is a superset of MA.)

▶ **Theorem 1.** $\mathsf{ZPP}^{\mathsf{NP}[1]}(\text{Block-Eq}) = \Theta(\sqrt{n})$, *and hence* $\mathsf{MA} \not\subseteq \mathsf{ZPP}^{\mathsf{NP}[1]}$.

To prove Theorem 1 (Section 3), we apply a new lower bound technique that combines the corruption bound with the 1-monochromatic rectangle size bound and asserts that they hold *simultaneously* (under the same distribution over inputs). We prove that, perhaps surprisingly, this combined technique gives a lower bound for $\mathsf{ZPP}^{\mathsf{NP}[1]}$ (though neither of the individual bounds suffices).

To apply our technique to Block-Eq, we first note that it is straightforward to achieve the two bounds separately: the 1-monochromatic rectangle size bound follows by simple counting, and the corruption bound follows by using Razborov's corruption lemma for the set-intersection function Inter [45] together with a simple reduction from Inter to Block-Eq. However, the latter does *not* result in a distribution satisfying the 1-monochromatic rectangle size bound for Block-Eq. To fix this problem, we argue that if we average Razborov's distribution over all ways of implementing the reduction (of which there are many), then the corruption bound is still satisfied, and now the 1-monochromatic rectangle size bound is also satisfied.

[2] The complement of block-equality is often known as list-non-equality.

2.2 US $\not\subseteq$ ZPP$^{\mathsf{NP}[1]}$

For the set-intersection function Inter, Alice and Bob are each given a subset of $[n]$ (and we identify the subset with its characteristic vector, a length-n bit string), and the goal is to output 1 when the sets are intersecting and 0 when they are disjoint.[3] Phrased as a composed function, Inter $:=$ Or $\circ$ And^n (for single-bit And). This is the canonical NP-complete problem in communication complexity, holding a comparable status to satisfiability, the canonical NP-complete problem in time-bounded complexity.

In the literature, "unique-set-intersection" commonly refers to the partial function version of Inter where the intersection is promised to have size 0 or 1. We propose a change in terminology, in order to be consistent with the following corresponding terminology from time-bounded complexity (see, e.g., [4, 51, 10]): Unique-satisfiability is the problem of determining whether the number of satisfying assignments of a formula is exactly 1, and is complete for the complexity class called US. Unambiguous-satisfiability is the problem of determining whether the number of satisfying assignments of a formula is 0 or 1 under the promise that one of these cases holds, and is complete for the complexity class called UP.

Therefore, we make the following declarations: Unique-set-intersection is the total function Unique-Inter: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that maps (x,y) to 1 iff $|x \cap y| = 1$, i.e., Unique-Inter $:=$ Unique-Or $\circ$ And^n where Unique-Or$(z) = 1$ iff the Hamming weight of z is 1. Unambiguous-set-intersection is the partial function Unambig-Inter: $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that maps (x,y) to $|x \cap y|$ if the latter is in $\{0,1\}$, i.e., Unambig-Inter $:=$ Unambig-Or $\circ$ And^n where Unambig-Or(z) equals the Hamming weight of z if the latter is in $\{0,1\}$.

Note that Unique-Inter is US-complete, where a cost-k US communication protocol is defined as a collection of rectangles $\left\{R_w \subseteq \{0,1\}^n \times \{0,1\}^n \ : \ w \in \{0,1\}^k\right\}$, where on input (x,y) the output of the protocol is 1 iff (x,y) is in R_w for exactly one w.

▶ **Theorem 2.** ZPP$^{\mathsf{NP}[1]}$(Unique-Inter) $= \Theta(n)$, *and hence* US $\not\subseteq$ ZPP$^{\mathsf{NP}[1]}$.

We give two proofs of Theorem 2. Both proofs show that Theorem 2 holds even under the promise that the input sets intersect in at most two coordinates. Also, in both proofs, handling ZPP$^{\mathsf{NP}[1]}$ instead of P$^{\mathsf{NP}[1]}$ incurs almost no extra complication.

The first proof (Section 3) employs the same lower bound technique as in Theorem 1, but where we use Razborov's corruption lemma [45] directly (and we must do a little analysis to verify the 1-monochromatic rectangle size bound). The optional second proof (relegated to the full version) uses information complexity tools (including an adaptation of the "partial information cost" approach from [24]) and, although longer to write, has some minor advantages over the first proof: It is more self-contained, as it does not rely on the corruption lemma (only on some basic facts that are standard in information complexity). Also, it directly handles success probability $1/2 + \epsilon$ (for any constant $\epsilon > 0$) without relying on the amplification result of [11] (whereas the first proof assumes success probability 0.999).

2.3 US $\not\subseteq$ coDP

The class DP was introduced in [39] to capture the complexity of certain exact versions of optimization problems. A set (of all 1-inputs of a function) is in DP iff it is the difference between two NP sets. The classes P, NP, and DP are the 0^{th}, 1^{st}, and 2^{nd} (respectively) levels of the so-called boolean hierarchy.

[3] We let "set-disjointness" refer to the complementary function where 1-inputs are disjoint.

We have $\mathsf{US} \subseteq \mathsf{DP}$ since to check that there is exactly one witness, we can use an NP computation to check that there is at least one witness, and another to check that there are at least two witnesses, and require that the first computation returns 1 and the second returns 0. However, it is unlikely that $\mathsf{US} \subseteq \mathsf{coDP}$: [10] showed that this inclusion cannot hold in the classical time-bounded setting unless the polynomial hierarchy collapses. This result does not yield a communication separation, since it is unknown whether the polynomial hierarchy collapses in the communication setting. Nevertheless, we show that indeed $\mathsf{US} \not\subseteq \mathsf{coDP}$ in communication complexity.

Formally, a cost-k coDP communication protocol is defined as a pair of collections of rectangles, $\{S_w \subseteq \{0,1\}^n \times \{0,1\}^n \ : \ w \in \{0,1\}^k\}$ and $\{T_w \subseteq \{0,1\}^n \times \{0,1\}^n \ : \ w \in \{0,1\}^k\}$, where on input (x,y) the output is 0 iff $(x,y) \in \bigcup_w S_w \setminus \bigcup_w T_w$.

▶ **Theorem 3.** $\mathsf{coDP}(\textsc{Unique-Inter}) = \Theta(n)$*, and hence* $\mathsf{US} \not\subseteq \mathsf{coDP}$*.*

To prove Theorem 3 (Section 3), we show that the same lower bound technique we introduced for $\mathsf{ZPP}^{\mathsf{NP}[1]}$ (the combination of the corruption bound and the 1-monochromatic rectangle size bound) *also* lower bounds coDP complexity. Thus we can simply reuse the application of the technique to Unique-Inter from Theorem 2. (Reusing the application to Block-Eq from Theorem 1 would show that Block-Eq $\notin \mathsf{coDP}$, but in fact Block-Eq $\notin \mathsf{P}^{\mathsf{NP}} \supseteq \mathsf{coDP}$ was already known [23].)

2.4 $\mathsf{ZPP}^{\mathsf{NP}[1]} \subseteq \mathsf{PostBPP}$

Consider bounded-error randomized computations (like in BPP) but with *postselection*: the output may come from $\{0,1,\perp\}$ and must be correct with high probability *conditioned* on not outputting $\perp$ (and the probability of this conditioning event must be positive). The complexity class corresponding to this model was originally called $\mathsf{BPP}_{\mathsf{path}}$ [22], but the name $\mathsf{PostBPP}$ (inspired by [1]) has gained popularity in the recent literature ([17] is one example) and seems more appropriate, so we use it instead.

According to modern conventions, the standard way to define the cost of a $\mathsf{PostBPP}$ communication protocol for F would be as the communication cost plus $\log(1/\alpha)$, where α is the minimum over all $(x,y) \in F^{-1}$ of the probability of not outputting $\perp$. (Allowing public randomness and not charging for α would enable $\mathsf{PostBPP}$ protocols to compute every function with constant cost.) Similarly, the cost of a PP (i.e., unbounded-error randomized) protocol would be the communication cost plus $\log(1/\epsilon)$ where $1/2+\epsilon$ is the minimum over all $(x,y) \in F^{-1}$ of the probability of outputting the correct answer.

However, for reasons that will become clear in Section 2.5, we choose to revert to the original convention of [3] and define $\mathsf{PostBPP}$ and PP in a slightly different but equivalent way: we do not charge for α or ϵ but we require the public randomness to be uniformly distributed over $\{0,1\}^k$ and we charge for k. For both $\mathsf{PostBPP}$ and PP, this cost measure is equivalent to the above "modern" definition within a constant factor and additive $O(\log n)$ term, by standard sparsification of the public randomness [38].

Formally, we define a $\mathsf{PostBPP}$ communication protocol Π for F in the following succinct way: For each outcome of the public randomness (which is uniformly distributed over $\{0,1\}^k$) there is a deterministic protocol outputting values in $\{0,1,\perp\}$. For each $(x,y) \in F^{-1}$ we must have $\mathbb{P}\big[\Pi(x,y) = F(x,y)\big] > 2 \cdot \mathbb{P}\big[\Pi(x,y) = 1 - F(x,y)\big]$. The cost is the randomness length k plus the maximum communication cost of any of the constituent deterministic protocols.

A priori it is not clear that any explicit lower bounds for $\mathsf{ZPP}^{\mathsf{NP}[1]}$ follow from prior work. The following result shows that in fact they do, since many explicit lower bounds for $\mathsf{PostBPP}$ were known.

▶ **Theorem 4.** $\mathsf{PostBPP}(F) \leq O\big(\mathsf{ZPP}^{\mathsf{NP}[1]}(F) + \log n\big)$ *for all* F*, and hence* $\mathsf{ZPP}^{\mathsf{NP}[1]} \subseteq \mathsf{PostBPP}$*.*

It turns out that Theorem 4 can be derived from the lower bound technique we develop for $\mathsf{ZPP}^{\mathsf{NP}[1]}$ in Section 3; however, that approach is more complicated than necessary and, more importantly, is specific to communication complexity. We give a proof of Theorem 4 (in the full version) using a black-box simulation that also works for time-bounded complexity, without exploiting any special properties of communication.

Intuitively, the worst case for simulating a $\mathsf{ZPP}^{\mathsf{NP}[1]}$ protocol is the following situation: Whenever the NP oracle responds "0" the protocol outputs the right answer, and whenever the NP oracle responds "1" the protocol outputs $\perp$ but would have output the wrong answer if the response were "0". In this situation, pretending the oracle always responds "0" would yield a BPP protocol (this is where we crucially need the success probability to be $> 1/2$). To handle more general situations, we must also randomly guess and verify a witness for the NP query, outputting $\perp$ if the witness is invalid.

2.5 Open issue: Rank-1 vs. rectangles

The classes $\mathsf{PostBPP}$ and PP can be further generalized by allowing the use of private randomness, which does not count toward the cost. This gives rise to the so-called "unrestricted probabilities" classes $\mathsf{UPostBPP}$ (which was defined, but not extensively studied, in [17]) and UPP (which is well-studied [41, 13, 48, 46]). In $\mathsf{UPostBPP}$ and UPP we can dispense with public randomness altogether as the public coins could be tossed privately by Alice and then sent to Bob.

Combinatorially, $\mathsf{PostBPP}$ and PP protocols of cost c induce a distribution over 2^c labeled rectangles (rank-1 matrices with 0-1 entries) each occurring with a "restricted" probability of at least 2^{-c} (see the full version). In the case of $\mathsf{UPostBPP}$ and UPP there is a similar characterization with rectangles replaced by nonnegative rank-1 matrices (see the full version). A natural question arises:

> **Informal question:** *Are rank-1 matrices inherently more powerful than rectangles in communication complexity?*

While it has been shown that, e.g., $\mathsf{PP} \neq \mathsf{UPP}$ [5, 47], the known examples of functions $F \in \mathsf{UPP} \setminus \mathsf{PP}$ can actually be computed without exploiting the full power of private randomness (their rank-1 property): we can use a UPP protocol whose associated rank-1 matrices are still rectangles, but occurring with *unrestricted*, possibly tiny, probability. We conclude that "PP vs. UPP" is not the right way to formalize our informal question (and the existing proofs for $\mathsf{PP} \neq \mathsf{UPP}$ do not incidentally answer our question), since UPP protocols can be more powerful than PP protocols for reasons unrelated to their rank-1 property.[4]

A better formalization is as follows. We define new communication classes, $\mathsf{UPostBPP}_{\square} \subseteq \mathsf{UPostBPP}$ and $\mathsf{UPP}_{\square} \subseteq \mathsf{UPP}$, in the same way as $\mathsf{PostBPP}$ and PP, except allowing the public randomness to be arbitrarily distributed over $\{0,1\}^k$ (still charging for k and not for α or ϵ). Combinatorially, we have a distribution over 2^k labeled rectangles, but with no restrictions on their probabilities. Our informal question can now be formalized as follows:

[4] The Log Rank Conjecture (and its variants) also do not adequately formalize our question, since the definition of a protocol imposes constraints on how its rectangles interrelate, whereas there are no analogous constraints on the rank-1 matrices making up a low-rank decomposition. A fairer formalization along these lines would be to compare the power (in representing boolean matrices) of sums of rank-1 matrices vs. linear combinations of rectangles; nothing seems to be known about this question.

Formal question: *Do we have* $\mathsf{UPostBPP} = \mathsf{UPostBPP}_\square$*? How about* $\mathsf{UPP} = \mathsf{UPP}_\square$*?*

The seemingly minor syntactic generalization introduced in the definitions of the $\square$-classes makes a huge difference: We observe (in the full version) that $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{UPostBPP}_\square$,[5] whereas it is known that $\mathsf{PostBPP}$ and P^{NP} are incomparable. Hence $\mathsf{UPostBPP}_\square$ is a strict superset of both $\mathsf{PostBPP}$ and P^{NP}. This leaves us with no known examples of functions to witness a separation for our "rank-1 vs. rectangle" question; currently the best gap is $\mathsf{UPostBPP}(F) \leq O(1)$ vs. $\mathsf{UPostBPP}_\square(F) \geq \Omega(\log n)$ where F is the usual Greater-Than function defined by $F(x, y) = 1$ iff $x > y$ when $x, y \in [2^n]$ are viewed as numbers. There is also no clear analogue of the "rank-1 vs. rectangle" distinction in query complexity, so a separation of the two notions in communication complexity might require interesting techniques. In fact, in the context of SBP (subclass of $\mathsf{PostBPP}$), it can be shown that rank-1 matrices do not add any power over mere rectangles [17].

2.6 $\mathsf{PP} \not\subseteq \mathsf{UPostBPP}_\square$

Our final result is to develop and apply a useful lower bound method for the class $\mathsf{UPostBPP}_\square$ introduced above. $\mathsf{PostBPP}$ already has a tight rectangle-based lower bound technique, which was dubbed "extended discrepancy" in [15] but was used earlier in [28] to show that $\mathsf{PP} \not\subseteq \mathsf{PostBPP}$. We strengthen the latter result to show that $\mathsf{PP} \not\subseteq \mathsf{UPostBPP}_\square$. (Showing $\mathsf{PP} \not\subseteq \mathsf{UPostBPP}$ remains open.) In our proof, we make use of the main theorem from [17], which applies to composed functions where the gadget is as follows.

▶ **Definition 5.** The *confounding* gadget g is defined by $g(x_i, y_i) \coloneqq \langle x_i, y_i \rangle \bmod 2$, where $x_i, y_i \in \{0,1\}^b$ and the block length b is $b(m) \coloneqq 100 \log m$.

We introduce the confounded-majority function, defined as Conf-Maj $\coloneqq f \circ g^m$ where f is the majority function and g is the confounding gadget. Note that Conf-Maj has input length $n \coloneqq m \cdot b = m \cdot 100 \log m$ and is in PP since Alice and Bob can pick $i \in [m]$ uniformly at random and then exchange $b + 1 \leq O(\log n)$ bits to evaluate $g(x_i, y_i)$.

▶ **Theorem 6.** $\mathsf{UPostBPP}_\square(\text{Conf-Maj}) = \Theta(n)$*, and hence* $\mathsf{PP} \not\subseteq \mathsf{UPostBPP}_\square$*.*

To prove Theorem 6 (in the full version) we introduce a lower bound technique for $\mathsf{UPostBPP}_\square$ that strengthens the extended discrepancy bound (for $\mathsf{PostBPP}$) by requiring it to hold under a product distribution over inputs (analogously to how [40] showed that the "monochromatic rectangle size bound under product distributions" gives a lower bound for P^{NP}). However, only a $\Omega(\sqrt{n \log n})$ lower bound for Conf-Maj follows using this technique, so to get the $\Omega(n)$ lower bound in Theorem 6, we generalize the technique further by allowing a rectangle's *size* to be measured with respect to some product distribution while its *error* is measured with respect to some other (arbitrary) distribution. (This is very analogous to the idea of relative discrepancy [14, 12].) To apply our general lower bound technique to Conf-Maj, we employ the communication-to-query machinery from [17] in a new, somewhat indirect way.

Finally, we mention another intriguing property of $\mathsf{UPostBPP}_\square$: By our lower bound technique and the results of [15] it follows immediately that to prove the Log Rank Conjecture,

[5] This inclusion also holds for time-bounded complexity. In defining the time-bounded version of $\mathsf{UPostBPP}_\square$, we would allow the distribution of the random string to depend nonuniformly on the input length n, though for the inclusion of P^{NP}, the distribution is computable in exponential time given the string 1^n.

i.e., that $\mathsf{P}(F) \leq \mathrm{poly}(\log \mathrm{rank}(F))$ for all total boolean matrices F, it suffices to prove the same with $\mathsf{UPostBPP}_{\square}$ instead of P. See the full version for more details.

3 Lower Bounds for Block-Equality and Unique-Set-Intersection

We now describe a technique for lower bounding both $\mathsf{ZPP}^{\mathsf{NP}[1]}$ and coDP communication.

▶ **Lemma 7.** *Suppose μ_0 is a distribution over $F^{-1}(0)$, μ_1 is a distribution over $F^{-1}(1)$, and C is a constant such that for every rectangle $R \subseteq \{0,1\}^n \times \{0,1\}^n$, $\mu_0(R) \leq C \cdot \mu_1(R) + \delta$, and if R is 1-monochromatic (i.e., contains no 0-inputs) then $\mu_1(R) \leq \delta$. Then*

(i) $\mathsf{ZPP}^{\mathsf{NP}[1]}(F) \geq \Omega(\log(1/\delta))$,

(ii) $\mathsf{coDP}(F) \geq \Omega(\log(1/\delta))$.

The first half of the technique ($\mu_0(R) \leq C \cdot \mu_1(R) + \delta$) is the corruption bound (which is a tight lower bound technique for so-called coSBP [18]), and the other half is the 1-monochromatic rectangle size bound (which is a tight lower bound technique for NP [33, §2.4]). The combined technique gives a lower bound for both $\mathsf{ZPP}^{\mathsf{NP}[1]}$ and coDP, even though neither of these classes appears to be a "combination" of coSBP and NP.

We prove parts (i) and (ii) of Lemma 7 in Section 3.1 and Section 3.2. Then we apply the technique to Block-Eq in Section 3.3 (thus proving Theorem 1), and finally we apply the technique to Unique-Inter in Section 3.4 (thus proving Theorem 2 and Theorem 3).

3.1 Proof of Lemma 7(i)

Suppose for contradiction there is a cost-$o(\log(1/\delta))$ $\mathsf{ZPP}^{\mathsf{NP}[1]}$ protocol Π computing F. Then in particular we have $\delta \leq o(1)$. By the amplification result of [11], we may assume $\mathbb{P}\big[\Pi(x,y) = \perp\big] \leq 1/10C$ for all $(x,y) \in F^{-1}$. By Markov's inequality and a union bound, we may fix a $\mathsf{P}^{\mathsf{NP}[1]}$-type protocol Π^* in the support of Π such that $\mathbb{P}_{(x,y)\sim\mu_0}\big[\Pi^*(x,y) = \perp\big] \leq 1/5C$ and $\mathbb{P}_{(x,y)\sim\mu_1}\big[\Pi^*(x,y) = \perp\big] \leq 1/5C$. Let the notation $k, R_v, S_{v,w}, o_v$ be with respect to Π^* (see the definition of $\mathsf{ZPP}^{\mathsf{NP}[1]}$ in Section 2.1), and note that without loss of generality, each o_v is non-constant (otherwise we could redefine $S_{v,w} = \emptyset$ for all w and redefine $o_v(1)$ arbitrarily).

For $b \in \{0,1,\perp\}$, define $W_b := \bigcup_{v,w\,:\,o_v(1)=b} S_{v,w}$ as the set of "witnessed" inputs (the NP oracle responds "1") on which Π^* outputs b, and define $N_b := \bigcup_{v\,:\,o_v(0)=b} \big(R_v \setminus \bigcup_w S_{v,w}\big)$ as the set of "non-witnessed" inputs (the NP oracle responds "0") on which Π^* outputs b. Note that $\{W_0, N_0, W_1, N_1, W_\perp, N_\perp\}$ partitions $\{0,1\}^n \times \{0,1\}^n$. By assumption, $\mu_0(W_\perp \cup N_\perp) \leq 1/5C$ and $\mu_1(W_\perp \cup N_\perp) \leq 1/5C$. By the correctness of Π, for $b \in \{0,1\}$ we have $(W_b \cup N_b) \cap F^{-1}(1-b) = \emptyset$.

▶ **Claim 8.** $\mu_0(W_0) \leq 1/4$.

▶ **Claim 9.** $\mu_0(N_0) \leq 1/4$.

This provides the contradiction since then $\mu_0\big(\{0,1\}^n \times \{0,1\}^n\big) = \mu_0(W_0) + \mu_0(N_0) + \mu_0(W_1 \cup N_1) + \mu_0(W_\perp \cup N_\perp) \leq 1/4 + 1/4 + 0 + 1/5C < 1$.

Proof of Claim 8. For each v, w such that $o_v(1) = 0$, we have $\mu_1(S_{v,w}) = 0$ and hence $\mu_0(S_{v,w}) \leq \delta$. Thus by a union bound, $\mu_0(W_0) \leq \sum_{v,w\,:\,o_v(1)=0} \mu_0(S_{v,w}) \leq 2^{o(\log(1/\delta))} \cdot \delta \leq \delta^{1-o(1)} \leq 1/4$. ◀

Proof of Claim 9. If v is such that $o_v(0) = 0$, then we have

$$\mu_0\big(R_v \setminus \bigcup_w S_{v,w}\big) \;\le\; \mu_0(R_v) \;\le\; C \cdot \mu_1(R_v) + \delta \;=\; C \cdot \mu_1\big(\bigcup_w S_{v,w}\big) + \delta$$

by the fact that $\big(R_v \setminus \bigcup_w S_{v,w}\big) \cap F^{-1}(1) = \emptyset$. Also, since each o_v is non-constant, we have

$$\begin{aligned}
\textstyle\sum_{v\,:\,o_v(0)=0} \mu_1\big(\bigcup_w S_{v,w}\big) \;&=\; \textstyle\sum_{v\,:\,o_v(0)=0,\,o_v(1)=\perp} \mu_1\big(\bigcup_w S_{v,w}\big) \\
&\quad + \textstyle\sum_{v\,:\,o_v(0)=0,\,o_v(1)=1} \mu_1\big(\bigcup_w S_{v,w}\big) \\
&\le\; \mu_1(W_\perp) + \textstyle\sum_{v,w\,:\,o_v(1)=1} \mu_1(S_{v,w}) \\
&\le\; \mu_1(W_\perp \cup N_\perp) + 2^{o(\log(1/\delta))} \cdot \delta \\
&\le\; 1/5C + \delta^{1-o(1)}
\end{aligned}$$

where the third line follows since $S_{v,w}$ is 1-monochromatic if $o_v(1) = 1$. Combining these, we have

$$\begin{aligned}
\mu_0(N_0) \;&=\; \textstyle\sum_{v\,:\,o_v(0)=0} \mu_0\big(R_v \setminus \bigcup_w S_{v,w}\big) \\
&\le\; \textstyle\sum_{v\,:\,o_v(0)=0} \Big(C \cdot \mu_1\big(\bigcup_w S_{v,w}\big) + \delta\Big) \\
&\le\; C \cdot \Big(\textstyle\sum_{v\,:\,o_v(0)=0} \mu_1\big(\bigcup_w S_{v,w}\big)\Big) + 2^{o(\log(1/\delta))} \cdot \delta \\
&\le\; C \cdot \big(1/5C + \delta^{1-o(1)}\big) + \delta^{1-o(1)} \\
&\le\; 1/4.
\end{aligned}$$

◀

3.2 Proof of Lemma 7(ii)

Suppose for contradiction there is a cost-k coDP protocol Π computing F where $k \le o(\log(1/\delta))$. Then in particular we have $\delta \le o(1)$. We have a pair of collections of rectangles, $\{S_w \;:\; w \in \{0,1\}^k\}$ and $\{T_w \;:\; w \in \{0,1\}^k\}$, such that if $F(x,y) = 0$ then $(x,y) \in \bigcup_w S_w$ and $(x,y) \notin \bigcup_w T_w$, and if $F(x,y) = 1$ then $(x,y) \notin \bigcup_w S_w$ or $(x,y) \in \bigcup_w T_w$. Since $\mu_0\big(\bigcup_w S_w\big) = 1$, there exists a w^* such that $\mu_0(S_{w^*}) \ge 2^{-k} \ge \delta^{1/3}$ and hence $\mu_1(S_{w^*}) \ge \frac{1}{C} \cdot (\delta^{1/3} - \delta) \ge \delta^{1/2}$. Since $S_{w^*} \cap F^{-1}(1) \subseteq \bigcup_w T_w$, there exists a w' such that $\mu_1(T_{w'}) \ge \mu_1\big(S_{w^*} \cap F^{-1}(1)\big) \cdot 2^{-k} > \delta^{1/2} \cdot \delta^{1/2} = \delta$. But $T_{w'}$ is 1-monochromatic since $F^{-1}(0) \cap \bigcup_w T_w = \emptyset$, so this is a contradiction.

3.3 Proof of Theorem 1

Let μ_0 be the uniform distribution over $\textsc{Block-Eq}^{-1}(0)$, and let μ_1 be the uniform distribution over the subset of $\textsc{Block-Eq}^{-1}(1)$ consisting of all (x,y) for which $x_i = y_i$ for a unique i.

▶ **Lemma 10.** *$\mu_0(R) \le 45 \cdot \mu_1(R) + 2^{-\Omega(\sqrt{n})}$ holds for every rectangle $R \subseteq \{0,1\}^n \times \{0,1\}^n$.*

▶ **Lemma 11.** *$\mu_1(R) \le 2^{-\Omega(\sqrt{n})}$ holds for every 1-monochromatic rectangle R of* $\textsc{Block-Eq}$.

Together, Lemma 10 and Lemma 11 show that the hypothesis of Lemma 7 holds with $F \coloneqq \textsc{Block-Eq}$, $C \coloneqq 45$, and $\delta \coloneqq 2^{-\Omega(\sqrt{n})}$. The lower bound in Theorem 1 follows. For the upper bound, in fact $\mathsf{ZPP}(\textsc{Block-Eq}) \le O(\sqrt{n})$ holds [33, §4.1.1] (though it is slightly quicker to see that $\mathsf{NP}(\textsc{Block-Eq}) \le O(\sqrt{n})$ holds by guessing i and deterministically verifying that $x_i = y_i$).

For the proofs of the lemmas, we define $m \coloneqq \sqrt{n}$ and $b \coloneqq \sqrt{n}$ (as in the notation for the decomposition $\textsc{Block-Eq} \coloneqq \textsc{Or} \circ \textsc{Eq}^m$ where Eq takes b-bit inputs).

Proof of Lemma 10. For $x^0, x^1, y^0, y^1 \in \{0,1\}^b$, we say the tuple (x^0, x^1, y^0, y^1) is *valid* iff $x^0 \neq y^0$, $x^0 \neq y^1$, $x^1 \neq y^0$, and $x^1 = y^1$. We say $\Xi := \big((x_1^0, x_1^1, y_1^0, y_1^1), \ldots, (x_m^0, x_m^1, y_m^0, y_m^1)\big)$ is valid iff it is a tuple of valid tuples. If Ξ is valid then the injection $\Phi_\Xi \colon \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^n \times \{0,1\}^n$ defined by $\Phi_\Xi(u,v) := \big(x_1^{u_1} \cdots x_m^{u_m}, y_1^{v_1} \cdots y_m^{v_m}\big)$ is a reduction from $\textsc{Inter} := \textsc{Or} \circ \textsc{And}^m$ (for single-bit And) to Block-Eq:

$$\textsc{Inter}(u,v) \;=\; \textsc{Block-Eq}\big(\Phi_\Xi(u,v)\big).$$

(In other words, the image of Φ_Ξ, as a submatrix of the Block-Eq matrix, is a copy of the Inter matrix.)

Define $\textsc{Unambig-Inter} := \textsc{Unambig-Or} \circ \textsc{And}^m$ where the partial function Unambig-Or is Or restricted to the domain of strings of Hamming weight 0 or 1; i.e., $\textsc{Unambig-Inter}^{-1}(0)$ consists of all pairs of disjoint sets, and $\textsc{Unambig-Inter}^{-1}(1)$ consists of all pairs of uniquely intersecting sets.

▶ **Lemma 12** ([45]). *There exists a distribution ν_0 over $\textsc{Unambig-Inter}^{-1}(0)$ and a distribution ν_1 over $\textsc{Unambig-Inter}^{-1}(1)$ such that $\nu_0(R) \leq 45 \cdot \nu_1(R) + 2^{-\Omega(m)}$ holds for every rectangle $R \subseteq \{0,1\}^m \times \{0,1\}^m$. Moreover, the uniquely intersecting coordinate in ν_1 is uniformly distributed.*

We claim that for $a \in \{0,1\}$ we have $\mu_a = \mathbb{E}_\Xi\, \Phi_\Xi(\nu_a)$ where a valid Ξ is chosen uniformly at random independently of ν_a. In other words, μ_a equals the distribution obtained by choosing Ξ, then independently taking a sample from ν_a, then applying Φ_Ξ to the sample (i.e., the uniform mixture of the distributions $\Phi_\Xi(\nu_a)$). We only argue that $\mu_1 = \mathbb{E}_\Xi\, \Phi_\Xi(\nu_1)$ (the argument for $\mu_0 = \mathbb{E}_\Xi\, \Phi_\Xi(\nu_0)$ is essentially the same). In fact, we make the stronger claim that for every $(u,v) \in \textsc{Unambig-Inter}^{-1}(1)$, say with $u_i = v_i = 1$, the distribution $\mathbb{E}_\Xi\, \Phi_\Xi(u,v)$ is uniform over the subset of $\textsc{Block-Eq}^{-1}(1)$ consisting of all (x,y) for which $x_i = y_i$ and $x_j \neq y_j$ for all $j \neq i$. The original claim follows from this since the uniquely intersecting coordinate i is uniformly distributed. The stronger claim follows immediately from the facts that the coordinates of Ξ are independent, that (x_i^1, y_i^1) is uniformly distributed over $\textsc{Eq}^{-1}(1)$, and that for $j \neq i$, (x_j^0, y_j^0), (x_j^0, y_j^1), and (x_j^1, y_j^0) are all marginally uniformly distributed over $\textsc{Eq}^{-1}(0)$. The claim is established.

Now for every rectangle $R \subseteq \{0,1\}^n \times \{0,1\}^n$, if we let $\Phi_\Xi^{-1}(R)$ denote the rectangle of all points in $\{0,1\}^m \times \{0,1\}^m$ that map into R under Φ_Ξ, then we have

$$\begin{aligned}
\mu_0(R) &= \mathbb{E}_\Xi \big(\Phi_\Xi(\nu_0)(R)\big) \\
&= \mathbb{E}_\Xi\, \nu_0\big(\Phi_\Xi^{-1}(R)\big) \\
&\leq \mathbb{E}_\Xi \Big(45 \cdot \nu_1\big(\Phi_\Xi^{-1}(R)\big) + 2^{-\Omega(m)}\Big) \\
&= 45 \cdot \mathbb{E}_\Xi\, \nu_1\big(\Phi_\Xi^{-1}(R)\big) + 2^{-\Omega(m)} \\
&= 45 \cdot \mathbb{E}_\Xi \big(\Phi_\Xi(\nu_1)(R)\big) + 2^{-\Omega(m)} \\
&= 45 \cdot \mu_1(R) + 2^{-\Omega(\sqrt{n})}.
\end{aligned}$$

◀

Proof of Lemma 11. Note that μ_1 is uniform over a set of size

$$m \cdot 2^b \cdot (2^{2b} - 2^b)^{m-1} \;=\; m \cdot 2^b \cdot 2^{2b(m-1)} \cdot (1 - 2^{-b})^{m-1} \;\geq\; \Omega(m \cdot 2^b \cdot 2^{2b(m-1)}).$$

If $R := A \times B$ is 1-monochromatic then $|A| \leq m \cdot 2^{b(m-1)}$ (since for any $y \in B$ there are at most $m \cdot (2^b)^{m-1}$ many x's for which $\textsc{Block-Eq}(x,y) = 1$), and similarly $|B| \leq m \cdot 2^{b(m-1)}$, and hence $|R| \leq m^2 \cdot 2^{2b(m-1)}$. It follows that

$$\mu_1(R) \;\leq\; \frac{m^2 \cdot 2^{2b(m-1)}}{\Omega(m \cdot 2^b \cdot 2^{2b(m-1)})} \;\leq\; O(m \cdot 2^{-b}) \;\leq\; 2^{-\Omega(\sqrt{n})}.$$

◀

3.4 Proof of Theorem 2 and Theorem 3

We again use the corruption lemma from [45], but now we need to take a closer look at the distribution over 1-inputs. Let $n = 4\ell - 1$. Let μ_0 be the distribution over $\textsc{Unique-Inter}^{-1}(0)$ that samples uniformly random disjoint sets of size ℓ, and let μ_1 be the distribution over $\textsc{Unique-Inter}^{-1}(1)$ that samples uniformly random uniquely intersecting sets of size ℓ.

▶ **Lemma 13** ([45]). *$\mu_0(R) \leq 45 \cdot \mu_1(R) + 2^{-\Omega(n)}$ holds for every rectangle $R \subseteq \{0,1\}^n \times \{0,1\}^n$.*

▶ **Lemma 14.** *$\mu_1(R) \leq 2^{-\Omega(n)}$ holds for every 1-monochromatic rectangle R of* Unique-Inter.

Together, Lemma 13 and Lemma 14 show that the hypothesis of Lemma 7 holds with $F \coloneqq \textsc{Unique-Inter}$, $C \coloneqq 45$, and $\delta \coloneqq 2^{-\Omega(n)}$. Theorem 2 and Theorem 3 follow.

Proof of Lemma 14. For each $i \in [n]$ let us define the rectangle $R_i \coloneqq \big\{(x,y) \in R \,:\, x_i = y_i = 1\big\}$, and note that the R_i's partition R. For each i we have $|R_i| \leq 2^{n-1}$ since every $(x,y) \in R_i$ is disjoint on the coordinates $[n] \setminus \{i\}$.[6] Hence $|R| \leq n2^{n-1} \leq 2^{(1+o(1))n}$.

Note that μ_1 can be sampled by the following process.

1. Pick a uniformly random $i \in [n]$.
2. Pick a uniformly random $H \subseteq [n] \setminus \{i\}$ of size $2\ell - 2$. There are $\binom{n-1}{2\ell-2} = \Theta(2^n/\sqrt{n})$ choices.
3. Pick a uniformly random partition $H = H_1 \cup H_2$ into sets of size $\ell - 1$. There are $\binom{2\ell-2}{\ell-1} = \Theta(2^{0.5n}/\sqrt{n})$ choices.
4. Let $x \coloneqq \{i\} \cup H_1$ and $y \coloneqq \{i\} \cup H_2$.

Hence μ_1 is uniform over its support of size $n \cdot \Theta(2^n/\sqrt{n}) \cdot \Theta(2^{0.5n}/\sqrt{n}) = \Theta(2^{1.5n}) \geq 2^{(1.5-o(1))n}$. It follows that $\mu_1(R) \leq 2^{(1+o(1))n}/2^{(1.5-o(1))n} \leq 2^{-\Omega(n)}$. ◀

Acknowledgements. We thank anonymous referees for helpful comments.

References

1 Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Proceedings of the Royal Society A*, 461(2063):3473–3482, 2005. doi:10.1098/rspa.2005.1546.

2 Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory*, 1(1), 2009. doi:10.1145/1490270.1490272.

3 László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *Proceedings of the 27th Symposium on Foundations of Computer Science (FOCS)*, pages 337–347. IEEE, 1986. doi:10.1109/SFCS.1986.15.

4 Andreas Blass and Yuri Gurevich. On the unique satisfiability problem. *Information and Control*, 55(1–3):80–88, 1982. doi:10.1016/S0019-9958(82)90439-9.

5 Harry Buhrman, Nikolai Vereshchagin, and Ronald de Wolf. On computation and communication with small bias. In *Proceedings of the 22nd Conference on Computational Complexity (CCC)*, pages 24–32. IEEE, 2007. doi:10.1109/CCC.2007.18.

[6] By [27], the bound $|R_i| \leq 2^{n-1}$ also holds assuming every input in R has intersection size either 1 or ≥ 3. Using this, it follows that Theorem 2 and Theorem 3 hold under the promise that at most two coordinates intersect.

6 Jin-Yi Cai and Venkatesan Chakaravarthy. On zero error algorithms having oracle access to one query. *Journal of Combinatorial Optimization*, 11(2):189–202, 2006. doi:10.1007/s10878-006-7130-0.

7 Amit Chakrabarti, Graham Cormode, Navin Goyal, and Justin Thaler. Annotations for sparse data streams. In *Proceedings of the 25th Symposium on Discrete Algorithms (SODA)*, pages 687–706. ACM-SIAM, 2014. doi:10.1137/1.9781611973402.52.

8 Amit Chakrabarti, Graham Cormode, Andrew McGregor, and Justin Thaler. Annotations in data streams. *ACM Transactions on Algorithms*, 11(1):7, 2014. doi:10.1145/2636924.

9 Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and Arthur–Merlin communication. In *Proceedings of the 30th Computational Complexity Conference (CCC)*. Schloss Dagstuhl, 2015. To appear.

10 Richard Chang, Jim Kadin, and Pankaj Rohatgi. On unique satisfiability and the threshold behavior of randomized reductions. *Journal of Computer and System Sciences*, 50(3):359–373, 1995. doi:10.1006/jcss.1995.1028.

11 Richard Chang and Suresh Purini. Amplifying $\mathrm{ZPP}^{\mathrm{SAT}[1]}$ and the two queries problem. In *Proceedings of the 23rd Conference on Computational Complexity (CCC)*, pages 41–52. IEEE, 2008. doi:10.1109/CCC.2008.32.

12 Lila Fontes, Rahul Jain, Iordanis Kerenidis, Sophie Laplante, Mathieu Lauriere, and Jérémie Roland. Relative discrepancy does not separate information and communication complexity. Technical Report TR15-028, Electronic Colloquium on Computational Complexity (ECCC), 2015.

13 Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *Journal of Computer and System Sciences*, 65(4):612–625, 2002. doi:10.1016/S0022-0000(02)00019-3.

14 Anat Ganor, Gillat Kol, and Ran Raz. Exponential separation of information and communication for boolean functions. In *Proceedings of the 47th Symposium on Theory of Computing (STOC)*. ACM, 2015. To appear.

15 Dmitry Gavinsky and Shachar Lovett. En route to the log-rank conjecture: New reductions and equivalent formulations. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 514–524. Springer, 2014. doi:10.1007/978-3-662-43948-7_43.

16 Dmitry Gavinsky and Alexander Sherstov. A separation of NP and coNP in multiparty communication complexity. *Theory of Computing*, 6(1):227–245, 2010. doi:10.4086/toc.2010.v006a010.

17 Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. In *Proceedings of the 47th Symposium on Theory of Computing (STOC)*. ACM, 2015. To appear.

18 Mika Göös and Thomas Watson. Communication complexity of set-disjointness for all probabilities. In *Proceedings of the 18th International Workshop on Randomization and Computation (RANDOM)*, pages 721–736. Schloss Dagstuhl, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.721.

19 Tom Gur and Ran Raz. Arthur–Merlin streaming complexity. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 528–539. Springer, 2013. doi:10.1007/978-3-642-39206-1_45.

20 Tom Gur and Ron Rothblum. Non-interactive proofs of proximity. In *Proceedings of the 6th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 133–142. ACM, 2015. doi:10.1145/2688073.2688079.

21 Bernd Halstenberg and Rüdiger Reischuk. Relations between communication complexity classes. *Journal of Computer and System Sciences*, 41(3):402–429, 1990. doi:10.1016/0022-0000(90)90027-I.

22 Yenjo Han, Lane Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. *SIAM Journal on Computing*, 26(1):59–78, 1997. doi:10.1137/S0097539792240467.

23 Russell Impagliazzo and Ryan Williams. Communication complexity with synchronized clocks. In *Proceedings of the 25th Conference on Computational Complexity (CCC)*, pages 259–269. IEEE, 2010. doi:10.1109/CCC.2010.32.

24 T.S. Jayram, Ravi Kumar, and D. Sivakumar. Two applications of information complexity. In *Proceedings of the 35th Symposium on Theory of Computing (STOC)*, pages 673–682. ACM, 2003. doi:10.1145/780542.780640.

25 Stasys Jukna. On graph complexity. *Combinatorics, Probability, & Computing*, 15(6):855–876, 2006. doi:10.1017/S0963548306007620.

26 Stasys Jukna. *Boolean Function Complexity: Advances and Frontiers*, volume 27 of *Algorithms and Combinatorics*. Springer, 2012.

27 Volker Kaibel and Stefan Weltge. A short proof that the extension complexity of the correlation polytope grows exponentially. *Discrete & Computational Geometry*, 53(2):397–401, 2015. doi:10.1007/s00454-014-9655-9.

28 Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Proceedings of the 18th Conference on Computational Complexity (CCC)*, pages 118–134. IEEE, 2003. doi:10.1109/CCC.2003.1214415.

29 Hartmut Klauck. Lower bounds for quantum communication complexity. *SIAM Journal on Computing*, 37(1):20–46, 2007. doi:10.1137/S0097539702405620.

30 Hartmut Klauck. On Arthur Merlin games in communication complexity. In *Proceedings of the 26th Conference on Computational Complexity (CCC)*, pages 189–199. IEEE, 2011. doi:10.1109/CCC.2011.33.

31 Hartmut Klauck and Ved Prakash. Streaming computations with a loquacious prover. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 305–320. ACM, 2013. doi:10.1145/2422436.2422471.

32 Hartmut Klauck and Ved Prakash. An improved interactive streaming algorithm for the distinct elements problem. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 919–930. Springer, 2014. doi:10.1007/978-3-662-43948-7_76.

33 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

34 Tak Wah Lam and Walter Ruzzo. Results on communication complexity classes. *Journal of Computer and System Sciences*, 44(2):324–342, 1992. doi:10.1016/0022-0000(92)90025-E.

35 Nathan Linial and Adi Shraibman. Learning complexity vs communication complexity. *Combinatorics, Probability, & Computing*, 18(1–2):227–245, 2009. doi:10.1017/S0963548308009656.

36 Satyanarayana Lokam. Spectral methods for matrix rigidity with applications to size-depth trade-offs and communication complexity. *Journal of Computer and System Sciences*, 63(3):449–473, 2001. doi:10.1006/jcss.2001.1786.

37 Satyanarayana Lokam. Complexity lower bounds using linear algebra. *Foundations and Trends in Theoretical Computer Science*, 4(1–2):1–155, 2009. doi:10.1561/0400000011.

38 Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 1991. doi:10.1016/0020-0190(91)90157-D.

39 Christos Papadimitriou and Mihalis Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244–259, 1984. doi:10.1016/0022-0000(84)90068-0.

40 Periklis Papakonstantinou, Dominik Scheder, and Hao Song. Overlays and limited memory communication. In *Proceedings of the 29th Conference on Computational Complexity (CCC)*, pages 298–308. IEEE, 2014. doi:10.1109/CCC.2014.37.

41 Ramamohan Paturi and Janos Simon. Probabilistic communication complexity. *Journal of Computer and System Sciences*, 33(1):106–123, 1986. doi:10.1016/0022-0000(86)90046-2.

42 Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Informatica*, 25(5):515–535, 1988. doi:10.1007/BF00279952.

43 Ran Raz and Amir Shpilka. On the power of quantum proofs. In *Proceedings of the 19th Conference on Computational Complexity (CCC)*, pages 260–274. IEEE, 2004. doi:10.1109/CCC.2004.1313849.

44 Alexander Razborov. On rigid matrices. Technical report, Steklov Mathematical Institute, 1989. In Russian.

45 Alexander Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992. doi:10.1016/0304-3975(92)90260-M.

46 Alexander Razborov and Alexander Sherstov. The sign-rank of AC^0. *SIAM Journal on Computing*, 39(5):1833–1855, 2010. doi:10.1137/080744037.

47 Alexander Sherstov. Halfspace matrices. *Computational Complexity*, 17(2):149–178, 2008. doi:10.1007/s00037-008-0242-4.

48 Alexander Sherstov. The unbounded-error communication complexity of symmetric functions. *Combinatorica*, 31(5):583–614, 2011. doi:10.1007/s00493-011-2580-0.

49 Rahul Tripathi. The 1-versus-2 queries problem revisited. *Theory of Computing Systems*, 46(2):193–221, 2010. doi:10.1007/s00224-008-9126-x.

50 Leslie Valiant. Graph-theoretic arguments in low-level complexity. In *Proceedings of the 6th Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 162–176. Springer, 1977. doi:10.1007/3-540-08353-7_135.

51 Leslie Valiant and Vijay Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47(3):85–93, 1986. doi:10.1016/0304-3975(86)90135-0.

52 Nikolai Vereshchagin. Relativizability in complexity theory. In *Provability, Complexity, Grammars*, volume 192 of *AMS Translations, Series 2*, pages 87–172. American Mathematical Society, 1999.

53 Henning Wunderlich. On a theorem of Razborov. *Computational Complexity*, 21(3):431–477, 2012. doi:10.1007/s00037-011-0021-5.

Information Complexity Is Computable*

Mark Braverman[†1] **and Jon Schneider**[2]

1 **Department of Computer Science, Princeton University, Princeton, NJ, USA**
mbraverm@cs.princeton.edu

2 **Department of Computer Science, Princeton University, Princeton, NJ, USA**
js44@cs.princeton.edu

Abstract

The information complexity of a function f is the minimum amount of information Alice and Bob need to exchange to compute the function f. In this paper we provide an algorithm for approximating the information complexity of an arbitrary function f to within any additive error $\epsilon > 0$, thus resolving an open question as to whether information complexity is computable.

In the process, we give the first explicit upper bound on the rate of convergence of the information complexity of f when restricted to b-bit protocols to the (unrestricted) information complexity of f.

1998 ACM Subject Classification E.4 Coding and Information Theory

Keywords and phrases Communication complexity, convergence rate, information complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.87

1 Introduction

In 1948, Shannon introduced the field of information theory as a set of tools for understanding the limits of one-way communication [15]. One of these tools, the information entropy function $H(X)$, measures the amount of information contained in a random source X.

The analogue of information entropy in communication complexity is *information complexity*. The information complexity of a function f is the least amount of information Alice and Bob need to exchange about their inputs to compute a function f. Just as the information entropy of a random source X provides a lower bound on the amount of communication required to transmit X, the information complexity of a function f provides a lower bound on the communication complexity of f [3]. Moreover, just as Shannon's source coding theorem establishes $H(X)$ as the asymptotic communication-per-message required to send multiple independent copies of X, the information complexity of f is the asymptotic communication-per-copy required to compute multiple copies of f in parallel on independently distributed inputs [7, 5].

These properties make information complexity a valuable tool for proving results in communication complexity. Communication complexity lower bounds themselves have a wide variety of applications to other areas of computer science; for example, results in circuit complexity such as Karchmer-Wigderson games and ACC lower bounds rely on communication complexity lower bounds [12, 4]. In addition, techniques from information complexity have been applied to prove various direct sum results in communication complexity

* A full version of this paper is available at http://arxiv.org/abs/1502.02971.

† Research supported in part by an NSF CAREER award (CCF-1149888), NSF CCF-1525342, a Packard Fellowship in Science and Engineering, and the Simons Collaboration on Algorithms and Geometry.

[8, 2, 11], including the only known direct sum results for general randomized communication complexity [3]. Information complexity has also been applied to prove a tight asymptotic bound on the communication complexity of the set disjointness function [6].

Despite this, many fundamental properties of information complexity remain unknown [6]. It is unknown how the information complexity of a function changes asymptotically as we allow the protocol to fail with probability ϵ. It is unknown how the information complexity of a function grows if we restrict our attention to protocols of bounded depth. Perhaps most surprisingly, it is even unknown if, given the truth table of a function f, whether it is possible to even compute (to within some additive factor of ϵ) the information complexity of f, $IC_\mu(f)$. (Contrast this with the case of the information entropy $H(X)$, which is easily computed given the distribution of X).

In this paper, we resolve the last of these questions; we prove that the information complexity of f is indeed computable. Our main technical result is an explicit bound on the convergence rate of b-bit information complexity (information complexity when restricted to protocols that have total communication at most b bits) to unrestricted information complexity. More specifically, we show how to convert an arbitrary protocol π into a protocol π' that leaks at most ϵ more information than π, but has communication cost at most $(N\epsilon^{-1})^{O(N)}$ bits, where N is the size of the truth table of f (Theorem 12). Equivalently, we show that the b-bit information complexity of f is at most $b^{-O(N^{-1})}$ larger than the information complexity of f. By then enumerating over all protocols with this communication cost, we obtain an algorithm that computes the information complexity of f to within an additive factor of ϵ in time $2^{\exp\left((N\epsilon^{-1})^{O(N)}\right)}$ (here N is the size of the truth table of f).

1.1 Prior Work

In [13, 14], Ma and Ishwar present a method to compute tight bounds on the information complexity of functions for protocols restricted to r rounds of computation. By examining the limit as r tends to infinity, this method allows them to numerically compute the information complexity of several functions (such as the 2-bit AND function). To make these computations provably correct, one would need effective (computable) estimates on the rate of convergence of r-round information complexity to the true information complexity. Such estimates were unknown prior to the present paper.

Plenty of unsolved problems of this flavor – where the computability of some limiting value is unknown despite it being straightforward to compute individual terms of this limit – occur in information theoretic contexts. One famous problem is the problem of computing the Shannon capacity of a graph, the amortized independence number of the kth power of a graph (this limiting quantity also has an interpretation as the zero-error channel capacity of a certain channel defined by this graph). While computing the independence number of any given graph is possible (albeit NP-hard), the rate at which this limit converges is unknown. Indeed, Alon and Lubetzky have shown that the limiting behavior of this quantity can be quite complex; no fixed number of terms of this limit is guaranteed to give a subpolynomial approximation to the Shannon capacity [1]. Another example, from the realm of quantum information theory, occurs in computing the quantum value of games [9]. Here it is straightforward to compute the quantum value of a game when limited to n bits of entanglement, but no explicit bounds are known for how many bits of entanglement are required to achieve within ϵ of optimal performance.

1.2 Outline of Proof

The main result of our paper is that zero-error information complexity is computable. Formally, we prove the following theorem.

▶ **Theorem 1.** *There exists an algorithm which, given a function $f : \mathcal{A} \times \mathcal{B} \to \{0,1\}$, initial distribution $\mu \in \Delta(\mathcal{A} \times \mathcal{B})$, and a real number $\epsilon > 0$, returns a value C between $IC_\mu(f) - \epsilon$ and $IC_\mu(f) + \epsilon$. This can be performed in time $2^{\exp\left((N\epsilon^{-1})^{O(N)}\right)}$, where $N = |\mathcal{A} \times \mathcal{B}|$.*

Throughout this paper, we will take the perspective of an outside observer watching in as Alice and Bob execute some protocol. This observer starts with some probabilistic *belief* about the inputs of Alice and Bob (initially this is just μ, the distribution of inputs to Alice and Bob). As Alice and Bob execute the protocol, they send each other *signals* – Bernoulli random variables that contain information about their inputs – which cause the observer to update his belief. The total amount of information leaked by the protocol to the participants can then be represented directly in terms of the final belief and initial belief. These notions are defined in more detail in Section 2.

The strategy of the proof is as follows. We start with a general protocol π for solving f, and whose information cost is very close to the information complexity of f. Unfortunately, we do not know anything about π besides the fact that it's a finite, discrete protocol that computes f without error. Note that if we could restrict π to a finite family of protocols (e.g. protocols that sent at most b bits, for an explicit bound $b = b(\epsilon, N)$), then we could just brute force over all such π's and compute the approximate information complexity of f. The proof shows that, indeed, there is always a protocol π' that can be derived from π, and which belongs to such an explicit family. The proof proceeds in several steps. In each step, more structure is added to π (structure that is then exploited by the following steps). The difficulty is, of course, ensuring at each step that π can be replaced with a more structured protocol π' while increasing its information cost by only, say, $\epsilon/10$. Ultimately, we manage to turn π into a protocol with r back-and-forth rounds, where r is an explicit function of N and ϵ. Finally, it is shown that an r-rounds of interaction protocol can be replaced with a b-bit protocol where $b = b(\epsilon, N, r) = b(\epsilon, N)$ is an explicit function, while only increasing its information cost by a controlled amount, completing the proof.

The full proof of Theorem 1 is roughly structured into three parts. In the first part, we begin by showing that we can 'discretize' any protocol π; that is, we can simulate any protocol π with a protocol π' that only uses a bounded number of different types of signals, but that only reveals a marginal amount of additional information. We accomplish this by building a 'mesh' of signals and rounding each signal in π to one of the signals in this mesh. This is described in Section 3.1.

In the second part, we show in Section 3.2 that we can transform any suitably discrete protocol π (i.e. one that uses an explicitly bounded number of distinct signals) into a protocol that uses few rounds. We achieve this via a bundling scheme; the main idea is that, where Alice would ordinarily send Bob one instance of a signal, she instead sends Bob several instances of this signal. Then, the next several times Alice would send that signal to Bob, Bob simply refers to the next unused copy sent by Alice, thus decreasing the number of rounds in the protocol.

In the third part, we show in Section 3.3 that it is never advantageous to send more than $\log N$ bits in any round of a protocol, thus providing an explicit bound on the communication complexity of the protocol. We accomplish this by showing that if it is ever the case that Alice sends one of $M > N$ different messages in a round, Alice can use public randomness

to sample a subset of N of these M messages and use $\log N$ bits to send one of these N messages instead.

Combining the above steps allows us to prove the following bound on the convergence rate of b-bit information complexity.

▶ **Theorem 12** (restated). *Let π be a communication protocol with information cost C that successfully computes function f over inputs drawn from distribution μ over $\mathcal{A} \times \mathcal{B}$. Then there exists a protocol π' with information cost at most $C + \epsilon$ that also successfully computes f over inputs drawn from μ, but has communication cost at most $b(f, \epsilon)$ where*

$$b(f, \epsilon) = (N\epsilon^{-1})^{O(N)} \tag{1}$$

and $N = |\mathcal{A} \times \mathcal{B}|$.

Finally, by reapplying the discretization procedure of Section 3.1, we show that it suffices to consider protocols whose signals all belong to an explicit finite set. By enumerating over all protocols of this depth that use signals from this set and computing the minimum information cost of any such protocol, we can therefore approximate $IC_\mu(f)$ to within ϵ, thus completing the proof.

The proof we provide below shows that zero-error internal and external information complexity are computable. We believe similar techniques can be used to show that ϵ-error information complexity is computable, but do not include such a proof in this paper.

1.3 Open Problems

Naturally, the most immediate open problem arising from our work is understanding whether (and how much) the rate of convergence in Theorem 12 can be improved:

▶ **Open Problem 2.** *What is the (worst case) rate of convergence of the b-bit (or r-round) information complexity of f to $IC_\mu(f)$? In other words, for a given $\epsilon > 0$ and truth table size $N = |\mathcal{A} \times \mathcal{B}|$, how large does $b(N, \epsilon)$ need to be to ensure that the b-bit information complexity $IC_{b,\mu}(f)$ satisfies*

$$IC_{b,\mu}(f) > IC_\mu(f) - \epsilon?$$

In this paper we prove that $b(N, \epsilon) \leq (N\epsilon^{-1})^{O(N)}$. On the other hand, [6] shows that when f is the two-bit AND (and thus $N = 4$ is a constant), the tight estimate for b is $b = \Theta(\epsilon^{-1/2})$. Therefore, the polynomial dependence on ϵ, even when N is a constant, is necessary. On the other hand, we do not have any interesting lower bounds on b in terms of N. In particular, it is not known whether the exponential dependence on N is necessary here.

The second open problem is in a similar vein, asking whether Theorem 1 can be improved.

▶ **Open Problem 3.** *What is the computational complexity of computing the (zero-error internal) information complexity of a function f within error ϵ given its truth table? By how much can the bound of $2^{\exp\left((N\epsilon^{-1})^{O(N)}\right)}$ be improved?*

By the analysis in Section 3.4, any progress on Problem 2 will translate into progress on Problem 3. For comparison, it is not hard to see that the trivial algorithm for computing the *average-case communication complexity* of a function $f : [n] \times [n] \to \{0, 1\}$ (so that $N = n^2$) within an additive error ϵ runs in time $2^{n \cdot N^{N/\epsilon}} = 2^{\exp((N\epsilon^{-1})^{O(1)})}$ (it suffices to look at all protocols of depth at most $\frac{N \log N}{\epsilon}$, of which there are at most $2^{n \cdot N^{N/\epsilon}}$). In other words, there is an exponential gap between the trivial communication complexity upper bound and the bound we obtain in Theorem 1.

2 Preliminaries

2.1 Information Theory

For an introduction to the information theoretic notions used throughout this paper, we refer the reader to [10] (A brief introduction can also be found in the full version of this paper).

2.2 Protocols and Information Complexity

In the two-party communication setting, Alice is given an element a from a finite set $\mathcal{A}$, while Bob is given an element b from a finite set $\mathcal{B}$, where (a, b) is drawn from some distribution μ over $\mathcal{A} \times \mathcal{B}$. Their goal is to compute $f(a, b)$, where $f : \mathcal{A} \times \mathcal{B} \to \{0, 1\}$ is a function known to both parties. They would like to accomplish this while revealing as little information as possible; either to each other (in the case of information cost) or to an outside observer (in the case of external information cost). To do this, they execute a *communication protocol*, which we view as being built out of *signals*.

▶ **Definition 4.** A *signal* σ over a set S is an assignment of a probability $\sigma_s \in [0, 1]$ to each element s in S. For a given element s of S, we define $\sigma(s)$ to be the Bernoulli random variable that equals 1 with probability σ_s.

▶ **Definition 5.** A *communication protocol* π is a finite rooted binary tree, where each non-leaf node is labeled by either a signal over $\mathcal{A}$ (corresponding to Alice's move) or a signal over $\mathcal{B}$ (corresponding to Bob's move), and each edge is labeled either 0 or 1. Alice and Bob can execute this protocol by starting at the root and repeatedly performing the following procedure; if the signal σ at the current node is a signal over $\mathcal{A}$, Alice sends Bob an instance of $\sigma(a)$, and they both move down the corresponding edge; likewise, if the signal is a signal over $\mathcal{B}$, Bob performs the analogous procedure.

Each leaf node is labeled with a value 0 or 1. We say the communication protocol *successfully computes* f with zero error if the value of the leaf node Alice and Bob finish the protocol on is always equal to $f(a, b)$ for all $(a, b) \in \mathcal{A} \times \mathcal{B}$ (in particular, even (a, b) where $\mu(a, b) = 0$). The communication cost $CC(\pi)$ of protocol π is equal to the depth of the deepest leaf in π.

This agrees with the usual definition of a private coins protocol (indeed, any bit Alice can ever send in any protocol must be a signal over $\mathcal{A}$, and likewise for Bob). A public coins protocol is simply a distribution over private coins protocols. For our purposes, it suffices to solely examine private coins protocols, since the information cost of a public coins protocol is simply the expected information cost of the corresponding private coins protocols.

As is standard, we will let A and B be random variables representing Alice's input and Bob's input respectively, and let Π be the random variable representing the protocol's transcript. We can then define the *information cost* of a protocol and the *information complexity* of a function as follows.

▶ **Definition 6.** The *information cost* of a protocol π is given by

$$IC_\mu(\pi) = I(A; \Pi | B) + I(B; \Pi | A)\,.$$

The *external information cost* of a protocol π is given by

$$IC^{ext}_\mu(\pi) = I(AB; \Pi)\,.$$

▶ **Definition 7.** The *information complexity* of a function f is given by

$$IC_\mu(f) = \inf_\pi IC_\mu(\pi)$$

where the infimum is over all protocols π that successfully compute f. Likewise, the *external information complexity* of a function f is given by

$$IC^{ext}_\mu(f) = \inf_\pi IC^{ext}_\mu(\pi)\,.$$

where again, the infimum is over all protocols π that successfully compute f.

Throughout the remainder of this paper, it will be useful to think of signals as operating on the space $\Delta(\mathcal{A}\times\mathcal{B})$ of probability distributions over $\mathcal{A}\times\mathcal{B}$, which we term *beliefs*. At the beginning of a protocol, an outside observer's belief is simply given by μ, the distribution (a,b) was drawn from. As this observer observes new signals, his belief evolves according to Bayes' rule; for example, if he observes the signal $\sigma(a)$ sent by Alice, his belief changes from the prior belief p to the posterior belief

$$p_0(a,b) = \frac{(1-\sigma_a)p(a,b)}{\sum_{i,j}(1-\sigma_i)p(i,j)} \tag{2}$$

if $\sigma(a) = 0$ (which occurs with probability $P_0 = \sum_{i,j}(1-\sigma_i)p(i,j)$) and to the posterior belief

$$p_1(a,b) = \frac{\sigma_a p(a,b)}{\sum_{i,j}\sigma_i p(i,j)} \tag{3}$$

if $\sigma(a) = 1$ (which occurs with probability $P_1 = \sum_{i,j}\sigma_i p(i,j)$). As shorthand, we will say that σ *shifts* belief p to (p_0,p_1). Note that the probabilities P_0 and P_1 are uniquely recoverable given p_0 and p_1 (in particular, treating beliefs as vectors in $\mathbb{R}^{|\mathcal{A}\times\mathcal{B}|}$, it must be the case that $P_0p_0 + P_1p_1 = p$ and that $P_0 + P_1 = 1$).

Throughout the remainder of the paper, we will let $N = |\mathcal{A}\times\mathcal{B}| = |\mathcal{A}|\cdot|\mathcal{B}|$. Note that N is the size of the truth table of f and is thus (in some sense) the size of the input to the problem of computing the information complexity of f. All logarithms are to base 2 unless otherwise specified.

3 Computability of Information Complexity

3.1 Discretizing signals

In the first part of the proof, we show that we can convert any protocol for f into a protocol that uses a bounded number of types of signals. In particular, we prove the following theorem.

▶ **Theorem 8.** *Let π be a communication protocol with information cost C. Then, for any $\epsilon > 0$, there exists a communication protocol π' that computes the same function as π with information cost at most $C+\epsilon$ but that only uses $Q = (N\epsilon^{-1})^{O(N)}$ different types of signals.*

Proof Sketch. Recall that signals are simply vectors in $\mathbb{R}^N$. Our general approach will therefore be to build a 'mesh' of signals in $\mathbb{R}^N$ and round each signal in our protocol to one of the nearby signals in the mesh. We can show this rounding procedure preserves the correctness of the protocol but possibly leaks some additional information.

Via the concavity of mutual information, it happens that if we take the width of this mesh to be $poly(\epsilon/N)$, then this rounding procedure leaks at most ϵ additional information. Such a mesh in N dimensions contains at most $(1/poly(\epsilon/N))^N = (N\epsilon^{-1})^{O(N)}$ different signals, as desired. ◀

The above sketch suppresses a number of technical difficulties in proving the above theorem. In particular, in the full paper, we demonstrate how to deal with:

1. Initial distributions μ that lack full support (Section 3.1 in full paper).
2. Signals sent near the boundary of $\Delta(\mathcal{A} \times \mathcal{B})$ (Section 3.2 in full paper).
3. Signals with widely differing magnitudes (Section 3.3 in full paper).

3.2 Bounding the number of alternations

We next show that we can convert a protocol for f that uses a bounded number of distinct signals (yet arbitrarily many of them) into a protocol for f that, while leaking at most ϵ extra information, uses a bounded number of alternations (steps in the protocol where Alice stops talking and Bob starts talking, or vice versa).

We achieve this by 'bundling' signals of the same type together; that is, at a point in the protocol where Alice would send Bob a certain signal, she may instead send him a bundle of t signals. Then, the next $t-1$ times Alice would send Bob this signal, Bob instead refers to the next unused signal in the bundle. If there are unused signals in a bundle, this may increase the information cost of the protocol; however, by choosing the size of the bundle cleverly, we can bound the size of this increase.

▶ **Definition 9.** Let π be a communication protocol and let $v_1, v_2, \ldots, v_k$ be one possible computation path for π. An *alternation* in this computation path is an index i where the signals at v_i and v_{i+1} are sent by different players. The number of alternations in π is the maximum number of alternations over all computation paths of π.

▶ **Theorem 10.** *Let π be a communication protocol with information cost C that only uses Q distinct signals. Then, for any $\epsilon > 0$, there exists a communication protocol π' that computes the same function as π with information cost at most $C + 2\epsilon$ but that uses at most*

$$W = \left(\frac{2Q \log N}{\epsilon} + Q\right) \frac{\log N}{\epsilon}$$

alternations.

Proof Sketch. Label our Q different signals $\sigma^{(1)}$ through $\sigma^{(Q)}$. We will reduce the number of alternations in π by bundling signals of the same type in large groups. That is, if Alice (at a specific point in the protocol) would send Bob signal $\sigma^{(i)}$, she instead sends Bob t copies of signal i (for an appropriately chosen t). Then, the next $t-1$ times in the protocol that Alice would send Bob signal $\sigma^{(i)}$, Bob instead refers to one of the unused t copies Alice originally sent. Once these t copies are depleted and protocol calls for a $(t+1)$st copy, the process repeats and Alice sends a new bundle to Bob (possibly with a different value for t).

We choose t as follows. Without loss of generality, assume Alice is sending a bundle of signals σ to Bob. Let Π_{pre} be the transcript of the protocol thus far. Let $X^t = (X_1, X_2, \ldots, X_t)$ be a random variable corresponding to t independently generated outputs of σ. We consider three cases:

Case 1: It is the case that $I(A; X^1 | \Pi_{pre} B) \geq \frac{\epsilon}{Q}$. In this case we set $t = 1$ (note that this is equivalent to simply following the original protocol).

Case 2: There exists a positive t_0 such that $\frac{\epsilon}{2Q} \leq I(A; X^{t_0} | \Pi_{pre} B) \leq \frac{\epsilon}{Q}$. In this case, we set $t = t_0$.

Case 3: For all positive t, $I(A; X^t | \Pi_{pre} B) \leq \frac{\epsilon}{2Q}$. In this case, we set t to be the maximum number of times signal σ is ever sent in protocol π.

The remainder of this proof is divided into three parts. In the first part, we argue that the three cases above are comprehensive. In the second part, we argue that the information cost of this new protocol is at most $C + \epsilon$. Finally, in the third part we argue that this bundling process decreases the total number of alternations to at most W. We briefly sketch these arguments here (see the full paper for detailed proofs).

Cases are comprehensive

It is not immediately clear that one of the three above cases must occur; it could be the case that $I(A; X^t|\Pi_{pre}B)$ 'jumps' from below $\epsilon/2Q$ to above ϵ/Q as t increases by one step. To show this cannot happen, we prove a 'diminishing returns' theorem for information revealed by additional copies of X (in particular, we show $I(A; X^{t+1}|\Pi_{pre}B) - I(A; X^t|\Pi_{pre}B)$ is decreasing in t).

Information leakage is small

When not all the signals in a bundle are used, this new protocol leaks some additional information over our original protocol. However, by the selection of t, each bundle is either a Case 1 bundle (which is immediately used up) or leaks at most ϵ/Q information. Since there are at most Q unused bundles (one for each signal type), we leak at most ϵ additional information.

Number of alternations is small

The number of alternations is at most the number of bundles sent. With the exception of Case 3 bundles (of which we send at most one of each type, for a total of Q), each bundle increases the expected information revealed by at least $\epsilon/2Q$. Since the amount of information revealed by the protocol is bounded above by $\log N$, in expectation we send at most $Q + \frac{2Q \log N}{\epsilon}$ bundles. To translate this to a worst case result, we simply terminate the protocol after sending at most W bundles; it then follows from Markov's inequality that we leak at most ϵ additional information by doing this. ◀

3.3 Bounding the number of bits

We finally show that each alternation in any protocol can be executed in a way that requires the exchange of at most $\log N$ bits without any loss in information cost; it follows that a protocol with at most W alternations can be converted into an equivalent protocol with communication complexity at most $W \log N$.

▶ **Theorem 11.** *Let π be a communication protocol with information cost C that has W alternations. Then there exists a communication protocol π' with information cost C that computes the same function as π but that sends a total of at most $W \log N$ bits.*

Proof. We will show how to execute each alternation of a protocol in at most $\log N$ bits. For simplicity, we will assume Alice and Bob have access to public randomness; this can later be converted into a protocol with only private randomness via the observation that some fixed choice of public randomness minimizes the information cost of the protocol.

Assume that Alice is speaking during some alternation of π, and let there be M possible strings she may send to Bob. If p is the belief at the beginning of the alternation, then at the end of the alternation we will have shifted to one of M possible beliefs, $p_1, p_2, \ldots, p_M$. Let α_i equal the probability we end up at belief p_i. We can therefore write $p = \sum_{i=1}^{M} \alpha_i p_i$.

In particular, note that p is contained in the convex hull of the p_i. Since the beliefs p_i lie in a space of dimension $N-1$, by Caratheodory's theorem, it follows that there exists some set $T \subset [M]$ such that $|T| \leq N$ and p is a convex combination of $\{p_i | i \in T\}$. Fix such a T. We can then write $p = \sum_{i \in T} \beta_i p_i$.

Let $\gamma = \min_{i \in T} \frac{\alpha_i}{\beta_i}$. Alice and Bob now use public randomness to flip a weighted coin that comes up heads with probability γ. If this coin comes up heads, then Alice samples an element of T according to the distribution induced by the β_i and sends this element to Bob using at most $\log |T| \leq \log N$ bits (by say, specifying its location in T).

If this coin comes up tails, they construct a new probability distribution α' over $[M]$ given by setting $\alpha'_i = \alpha_i - \gamma\beta_i$ for all $i \in [M]$ (where $\beta_i = 0$ if $i \notin T$) and renormalizing. Note that by our choice of γ, it will be the case that $\alpha'_i \geq 0$ for all i; moreover, for at least one i, $\alpha'_i = 0$. They now repeat this process for this new distribution α'.

Note that throughout this modified round, Alice sends in total at most $\log N$ bits to Bob. Moreover, each time they use public randomness, the round either terminates (Alice sends a message to Bob) or the size of the support of α shrinks by one, guaranteeing that the round eventually terminates. Finally, at the end of this process, the probability Bob receives message $i \in [M]$ from Alice is equal to α_i, hence making this modified round information-theoretically equivalent to the original round.

Applying this to every alternation in a protocol π with W rounds results in a protocol π' with communication complexity of at most $W \log N$, as desired. ◀

3.4 Computing Information Complexity

Combining the results of Theorems 8, 10 and 11, we obtain the following result.

▶ **Theorem 12.** *Let π be a communication protocol with information cost C that successfully computes function f over inputs drawn from distribution μ over $\mathcal{A} \times \mathcal{B}$. Then there exists a protocol π' with information cost at most $C + \epsilon$ that also successfully computes f over inputs drawn from μ, but has communication cost at most $b(f, \epsilon)$ where*

$$b(f, \epsilon) = (N\epsilon^{-1})^{O(N)}$$

and $N = |\mathcal{A} \times \mathcal{B}|$.

By a similar rounding technique to that in Section 3.1, we can further ensure each signal in π belongs to a set $\mathbf{S}$ of size $(N\epsilon^{-1})^{O(N^2)}$ (see Section 3.7 of the full paper for details). We can now proceed to prove our main theorem.

Proof of Theorem 1. Fix an $\epsilon > 0$; we will show how to approximate the information complexity of f to within an additive factor of ϵ.

By Theorem 12, there exists some protocol with information cost at most $IC_\mu(f) + \epsilon$ with communication complexity at most $B(f, \epsilon)$ and that only uses signals in the set $\mathbf{S}$. The number of such protocols is finite; in particular each such protocol has at most $2^{B(f,\epsilon)}$ nodes, each of which is labelled by one of $|\mathbf{S}|$ signals. Since $|\mathbf{S}| = (N\epsilon^{-1})^{O(N^2)}$, it follows that the total number of protocols is at most

$$\begin{aligned} |\mathbf{S}|^{2^{B(f,\epsilon)}} &= (N\epsilon^{-1})^{O(N^2)2^{(N\epsilon^{-1})^{O(N)}}} \\ &= 2^{\exp\left((N\epsilon^{-1})^{O(N)}\right)} \end{aligned}$$

The information cost of a protocol with depth B (and thus at most 2^B nodes) can be computed in time $2^{O(B)}$. It follows that computing the minimum information cost of the

above protocols can be done in time $2^{\exp\left((N\epsilon^{-1})^{O(N)}\right)}$, and hence one can approximate $IC_\mu(f)$ to within an additive factor ϵ in time $2^{\exp\left((N\epsilon^{-1})^{O(N)}\right)}$, as desired. ◀

Acknowledgments. We would like to thank Ankit Garg and Noga Ron-Zewi for providing helpful comments on an earlier draft of this paper.

References

1. Noga Alon and Eyal Lubetzky. The shannon capacity of a graph and the independence numbers of its powers. *Information Theory, IEEE Transactions on*, 52(5):2172–2176, 2006.
2. Ziv Bar-Yossef, Thathachar S Jayram, Ravindra Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 209–218. IEEE, 2002.
3. Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM Journal on Computing*, 42(3):1327–1363, 2013.
4. Richard Beigel and Jun Tarui. On acc [circuit complexity]. In *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pages 783–792. IEEE, 1991.
5. Mark Braverman. Interactive information complexity. *SIAM Journal on Computing*, 44(6):1698–1739, 2015.
6. Mark Braverman, Ankit Garg, Denis Pankratov, and Omri Weinstein. From information to exact communication. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 151–160. ACM, 2013.
7. Mark Braverman and Akhila Rao. Information equals amortized communication. *Information Theory, IEEE Transactions on*, 60(10):6058–6069, 2014.
8. Amit Chakrabart, Yaoyun Shi, Anthony Wirth, and Andrew Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 270–278. IEEE, 2001.
9. Richard Cleve, Peter Høyer, Benjamin Toner, and John Watrous. Consequences and limits of nonlocal strategies. In *Computational Complexity, 2004. Proceedings. 19th IEEE Annual Conference on*, pages 236–249. IEEE, 2004.
10. Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley series in telecommunications. J. Wiley and Sons, New York, 1991.
11. Rahul Jain. New strong direct product results in communication complexity. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 18, page 2, 2011.
12. Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require superlogarithmic depth. *SIAM Journal on Discrete Mathematics*, 3(2):255–265, 1990.
13. Nan Ma and Prakash Ishwar. Two-terminal distributed source coding with alternating messages for function computation. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 51–55. IEEE, 2008.
14. Nan Ma and Prakash Ishwar. Some results on distributed source coding for interactive function computation. *Information Theory, IEEE Transactions on*, 57(9):6180–6195, 2011.
15. Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

Rényi Information Complexity and an Information Theoretic Characterization of the Partition Bound*

Manoj M. Prabhakaran[1] and Vinod M. Prabhakaran[2]

1 **Department of Computer Science, University of Illinois, Urbana-Champaign, USA**
mmp@illinois.edu

2 **School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, India.**
vinodmp@tifr.res.in

Abstract

In this work we introduce a new information-theoretic complexity measure for 2-party functions, called Rényi information complexity. It is a lower-bound on communication complexity, and has the two leading lower-bounds on communication complexity as its natural relaxations: (external) information complexity and logarithm of partition complexity. These two lower-bounds had so far appeared conceptually quite different from each other, but we show that they are both obtained from Rényi information complexity using two different, but natural relaxations:

1. The relaxation of Rényi information complexity that yields information complexity is to change the order of Rényi mutual information used in its definition from infinity to 1.
2. The relaxation that connects Rényi information complexity with partition complexity is to replace protocol transcripts used in the definition of Rényi information complexity with what we term "pseudotranscripts," which omits the interactive nature of a protocol, but only requires that the probability of any transcript given inputs x and y to the two parties, factorizes into two terms which depend on x and y separately. While this relaxation yields an apparently different definition than (log of) partition function, we show that the two are in fact identical. This gives us a surprising characterization of the partition bound in terms of an information-theoretic quantity.

We also show that if both the above relaxations are simultaneously applied to Rényi information complexity, we obtain a complexity measure that is lower-bounded by the (log of) relaxed partition complexity, a complexity measure introduced by Kerenidis et al. (FOCS 2012). We obtain a sharper connection between (external) information complexity and relaxed partition complexity than Kerenidis et al., using an arguably more direct proof.

Further understanding Rényi information complexity (of various orders) might have consequences for important direct-sum problems in communication complexity, as it lies between communication complexity and information complexity.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases Information Complexity, Communication Complexity, Rényi Mutual Information

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.88

* Full version available at http://arxiv.org/abs/1511.07949.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 88; pp. 88:1–88:14

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

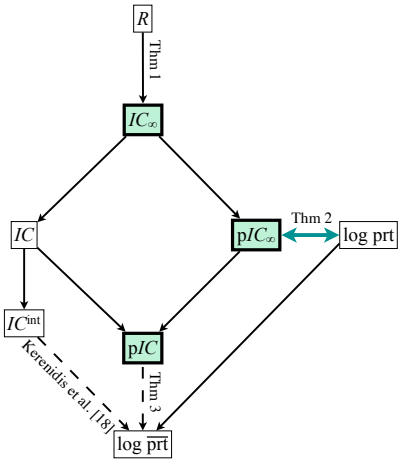


Figure 1 New complexity measures (shaded) and their relation to existing ones. Existing ones shown include the (public-coin) worst-case communication complexity (R), external and internal information complexity (IC and IC^{int}), partition complexity (prt) and relaxed partition complexity ($\overline{\text{prt}}$). An arrow from one measure to another shows that the latter is a lower-bound for the former. (The dashed lines indicate that the lower bound holds up to constant factors and shifts in error bounds.) $\text{p}IC_\infty$ is exactly equal to $\log \text{prt}$.

1 Introduction

Communication complexity, since the seminal work of Yao [26], has been a central question in theoretical computer science. Many of the recent advances in this area have centered around the notion of information complexity, which measures the *amount of information* about the inputs – rather than the *number of bits* – that should be present in a protocol's transcript, if it should compute a function (somewhat) correctly. The more traditional approach for lower bounding communication complexity relied on *combinatorial complexity measures* of functions. The goal of this work is to relate these two lines of studying communication complexity with each other.

Currently, the two leading lower bounds for communication complexity in the literature come from these two lines: (external) information complexity IC [8, 2] and partition complexity prt [14]. Either of these two lower bounds upper-bounds (and hence gives an equally good or better lower bound than) all the other bounds used in the literature. An intriguing problem in this area has been to understand if one of these two bounds is a better lower-bound than the other. An important motivation behind this problem is the possibility of separating IC from communication complexity via an intermediate combinatorial lower bound, which will have consequences for direct-sum results in communication complexity (since IC is known to be equal to amortized communication complexity [5, 4]).

Kerenidis et al. [18] showed that information complexity "subsumes" (the logarithm of) a relaxed variant of partition complexity, $\overline{\text{prt}}$, in the sense that any lower bound on $\log \overline{\text{prt}}$ in fact yields a lower bound on information complexity. Thus bounding $\log \overline{\text{prt}}$ cannot yield stronger lower bounds than bounding information complexity. In turn, all the combinatorial bounds in the literature – other than $\log \text{prt}$ – are subsumed by $\log \overline{\text{prt}}$. On the other hand, in recent breakthrough results, Ganor, Kol and Raz [10, 11] showed that for a certain range of parameters, combinatorial lower bounds can be significantly stronger than information

complexity lower bounds.[1] It remains open if such separations are possible for a less restrictive range of parameters (e.g., with communication complexity that is say, super-logarithmic in the input size). In the absence of a result analogous to that of [18] for prt itself, prt remains a candidate for showing such separations.

In this work, we do not pursue the question of whether $\log \mathrm{prt}$ could be larger than IC or vice versa. Instead, we develop a new information-theoretic complexity measure, IC_∞ which is as large or larger than both IC and $\log \mathrm{prt}$ (see Figure 1), and has *natural* relaxations that yield IC_∞ and $\log \mathrm{prt}$ respectively. *IC_∞ is thus a candidate for separating IC and communication complexity for a larger range of parameters than currently known to be possible.* Further, the relaxation of IC_∞ to $\log \mathrm{prt}$ reveals a surprising information-theoretic definition for prt. Since this new definition of (log of) prt has a markedly different form, we give it a different name, $\mathrm{p}IC_\infty$.

We also consider applying *both the relaxations* mentioned above simultaneously to IC_∞. This yields a new complexity measure $\mathrm{p}IC$. We then show that $\mathrm{p}IC$ is essentially lower bounded by $\log \overline{\mathrm{prt}}$, the relaxed partition complexity. *This recovers a result similar to that of [18], but with sharper parameters and an arguably more direct proof.*[2]

The relation between the new and old complexity measures are shown in Figure 1. (Also see Figure 3 for further extensions.) The new complexity measures are informally described below.

Rényi Information Complexity. (External) Information complexity of a function is defined as the mutual information between the transcript and the inputs, and is a lower bound on the communication complexity of the function. The notion of mutual information in this definition is due to Shannon. Rényi mutual information $I_\alpha(A;B)$, parametrized by $\alpha \geq 0$, is a generalization of Shannon's mutual information (see [25] for a recent treatment), with the latter corresponding to $\alpha \to 1$. We observe that information complexity continues to be a lower bound on communication complexity for all values of α. In particular, we may consider I_∞ instead of I_1 to define information complexity. The resulting notion of information complexity will be called IC_∞.

Pseudotranscript Complexity. Communication complexity, as well as information complexity, is defined in terms of a protocol. In contrast, the more traditional combinatorial lower bounds on communication complexity are defined in terms of simpler combinatorial properties of the function's truth table. We propose complexity measures based on one such property (which has been widely used in the analysis of protocols, but to the best of our knowledge, has never been isolated to define a complexity measure of functions).

Consider a function (generalized later to relations) $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. We define a random variable Q over a space $\mathcal{Q}$ to be a *pseudotranscript* for f if there exist two functions $\alpha : \mathcal{Q} \times \mathcal{X} \to \mathbb{R}^+$ and $\beta : \mathcal{Q} \times \mathcal{Y} \to \mathbb{R}^+$, such that $\Pr[Q = q | X = x, Y = y] = \alpha(q,x)\beta(q,y)$, for all $q \in \mathcal{Q}, x \in \mathcal{X}, y \in \mathcal{Y}$. This definition is motivated by the fact that the transcripts in a protocol do satisfy it (see Footnote 7). However, a pseudotranscript need not correspond to a protocol (indeed, any "tiling" of a function's table yields a pseudotranscript, but it need

[1] These results use combinatorial lower bounds to establish that communication complexity could be exponentially larger than information complexity. The communication complexity in these examples is (sub-)logarithmic in the size of the input itself.

[2] Our result does not subsume the result of Kerenidis et al. [18], as they deal with internal information complexity, while it is more natural for us to work with external information complexity. Conversely, the result of [18] does not yield our result for external information complexity (due to the parameters), nor the relation with the intermediate complexity measure $\mathrm{p}IC$.

not correspond to a valid protocol). We also associate a value z_q with a pseudotranscript q; the error $\mathrm{err}_{f,Q}$ is defined in terms of the probability of this value matching the function's output. We do not include any other properties of a protocol in defining a pseudotranscript.

We can define complexity measures $\mathrm{p}IC$ and $\mathrm{p}IC_\infty$ as relaxations of IC and IC_∞, simply by replacing protocols in their definitions with pseudotranscripts.

Relations Among the Complexities. The main results in this work, apart from introducing the new complexity measures, are connections between $\mathrm{p}IC_\infty$ and prt and between $\mathrm{p}IC$ and $\overline{\mathrm{prt}}$.

- Firstly, we show that $\mathrm{p}IC_\infty = \log \mathrm{prt}$. $\mathrm{p}IC_\infty$ and prt are defined very differently. prt is concerned with *tiling* the function table with weighted tiles: a tile t is a rectangle in the input domain along with an output value z_t. prt is the minimum total weight of tiles needed such that for each input (x, y), the weight of the tiles covering it adds up to 1, and the weight of the tiles with $z_t \neq f(x, y)$ is below the error threshold $\mathcal{E}(x, y)$.[3] On the other hand, $\mathrm{p}IC_\infty$ relates to pseudotranscripts q, which are similar to tiles in that they define a value z_q and a rectangle of all (x, y) such that $p(q|x, y) > 0$, but are more general in that there is no single "weight" on such a rectangle. Given our definitions, it is not hard to see that $\log \mathrm{prt}$ is as large or larger than $\mathrm{p}IC_\infty$, as any tiling can be naturally interpreted as a pseudotranscript Q with the same error, and in that case, the log of the value of the tiling indeed equals $I_\infty(X, Y; Q)$. What is more surprising is that any pseudotranscript Q can be converted to a tiling of the appropriate value (and same error). This conversion "slices" an uneven weight function $p(q|x, y)$ over a rectangle into weights $\omega_{q,t}$ over tiles t inside the rectangle; the weight of a tile t is the sum of the contributions to its weight from all the different values of q: $w(t) = \sum_q \omega_{q,t}$. Then it turns out that the value of the tiling so obtained will be equal to $I_\infty(X, Y; Q)$.
  This equivalence gives a new perspective on the partition complexity. Firstly, it shows that partition complexity exploits *exactly* the properties of a pseudotranscript, which is not apparent from its original definition. Secondly, it gives an information theoretic interpretation of a complexity measure defined in a traditional combinatorial manner. This is the first instance of the two lines of lower-bounding techniques for communication complexity – information theoretic and combinatorial – converging.
- Our second main result is that lower bounds on $\log \overline{\mathrm{prt}}$ are in fact lower bounds on $\mathrm{p}IC$. More precisely, we show that $\mathrm{p}IC(f, \varepsilon) \geq \delta \log \overline{\mathrm{prt}}(f, \varepsilon + \delta) - (\delta \log \log |\mathcal{X}||\mathcal{Y}| + 3)$. This is along the same lines as the result of [18], with improved parameters (in [18], the multiplicative overhead in the leading term is δ^2 instead of δ).
  The proof of this result is technically more involved, but is closely based on the simple slicing construction from the above result. The high-level idea is to first slice $p(q|x, y)$ into weights $\omega_{q,t}$ for each tile t, and then discard the contributions to $w(t)$ from those $\omega_{q,t}$ which are too large. One needs to ensure that the weight of the tiles discarded in this fashion is small (as it contributes to the error), while the weight of the remaining tiles is also small (as it contributes to the value of the tiling). For the first part, we show how (Shannon's) mutual information $I(X, Y; Q)$ can be approximated by a convex combination of non-negative values, and then apply Markov's inequality. For the second part, we rely on a geometric argument to derive a bound on the weight of the remaining tiles.

[3] For prt, as well as $\mathrm{p}IC_\infty$ and IC_∞, we use a very general notion of error, in which the error is specified as a function $\mathcal{E} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$.

1.1 Related Work

Many of the recent advances in the field of communication complexity [26] have followed from using various notions of information complexity. Earlier notions of information complexity appeared implicitly in several works [1, 20, 23], and was first explicitly defined in [8] and further developed in [2]. Information complexity has been extensively used or studied in the recent communication complexity literature (e.g., [5, 4, 6, 7, 18, 3, 10, 9, 11]). The notion was also adapted to specialized models or tasks [17, 15, 16, 12].

The partition bound was developed in [14], and has subsumed a long line of combinatorial bounds [19] (see e.g., [14, 9]). The relaxed partition bound put forth in [18], similarly subsumes several combinatorial bounds, with the exception of the partition bound itself.

In 1960, generalizing Shannon's entropy, Rényi proposed new measures of entropy and divergence [22], now known after him. Subsequently, several authors developed different notions of mutual information based on these measures. One such definition attributed to Sibson [24] has recently come to be regarded as the most standard choice [25], and this is the basis for our definition of $I_\infty(A:B)$. Properties of I_α for various parameters α have been studied in [13, 25]. In information theory literature, the use of generalized notions of mutual information to obtain strong lower bounds for "one-shot" versions of communication problems (rather than amortized/direct-sum versions where Shannon's mutual information is often appropriate) has a long history starting with the work of Ziv and Zakai [28, 27]. In the communication complexity literature, Rényi divergence was used as a technical tool in deriving one of the results in [2].

Recently, the authors of this work proposed a distributional complexity measure, *Wyner tension* (or more generally, tension gap) which is a lower bound for information complexity [21]. We leave it for future work to explore the exact connections between these bounds and the ones in the current work. We mention that for the case when the inputs are independent, Wyner tension is identical to $\mathrm{p}IC^{\mathrm{int}}$ (defined in Section 6), and a result in [21] is subsumed by the results in this work.

2 Preliminaries

Let $f : \mathcal{X} \times \mathcal{Y} \to 2^{\mathcal{Z}}$ be a relation. Alice who has input $x \in \mathcal{X}$ and Bob who has input $y \in \mathcal{Y}$ want to output any $z \in f(x,y)$. We consider public-coin protocols, in which Alice and Bob have access to a common random string independent of the inputs; they may also use private local randomness. For such a protocol π, we say that the probability of error, which we view as a **function** of $(x,y) \in \mathcal{X} \times \mathcal{Y}$, is

$$\mathrm{err}_{f,\pi}(x,y) = \Pr[\pi(x,y) \notin f(x,y)],$$

where $\pi(x,y)$ is the *output* of the protocol and the probability is over the randomness in the protocol execution.[4] An error function $\mathcal{E}$ that is of particular interest is the constant (or worst-case) error function: $\mathcal{E}(x,y) = \varepsilon$ for some constant ε, for all $(x,y) \in \mathcal{X} \times \mathcal{Y}$.

For a protocol π, let $\#\mathrm{bits}(\pi,x,y)$ denote the maximum number of bits exchanged in an execution of π with inputs (x,y), in the worst case (i.e., over all choices of randomness). Note that this measure excludes the number of bits in the public randomness. The (worst

[4] For a protocol to be considered valid, we will insist that the two parties output the same value with probability 1; hence the output of a protocol is well-defined.

case) *communication complexity* $R(f,\mathcal{E})$ of f, for an error function $\mathcal{E}$, is defined as

$$R(f,\mathcal{E}) = \inf_{\substack{\text{protocol } \pi: \\ \text{err}_{f,\pi} \leq \mathcal{E}}} \max_{x,y} \quad \#\text{bits}(\pi, x, y).$$

To define information complexities, we will need to consider the distribution $\mathbf{p}_{X,Y}$ on the inputs X, Y. Let Π be the random variable that denotes the communication transcript *and* the public-coins of the protocol π. Then, the external information cost of the protocol π under the input distribution $\mathbf{p}_{X,Y}$ is $I(X,Y;\Pi)$, i.e., the amount of information about the inputs X, Y contained in Π. The (non-distributional) *external information complexity* $IC(f,\mathcal{E})$ is defined as

$$IC(f,\mathcal{E}) = \inf_{\substack{\text{protocol } \pi: \\ \text{err}_{f,\pi} \leq \mathcal{E}}} \max_{\mathbf{p}_{X,Y}} \quad I(X,Y;\Pi).$$

Similarly, internal information complexity is defined as

$$IC^{\text{int}}(f,\mathcal{E}) = \inf_{\substack{\text{protocol } \pi: \\ \text{err}_{f,\pi} \leq \mathcal{E}}} \max_{\mathbf{p}_{X,Y}} \quad I(X;\Pi|Y) + I(Y;\Pi|X).$$

Here the internal information cost, $I(X;\Pi|Y) + I(Y;\Pi|X)$, of the protocol π under input distribution $\mathbf{p}_{X,Y}$ is the sum of the information learned by the parties about each other's input from Π. The following relationship between these quantities is well-known.

$$IC^{\text{int}}(f,\mathcal{E}) \leq IC(f,\mathcal{E}) \leq R(f,\mathcal{E}).$$

A *tile* for $(\mathcal{X},\mathcal{Y},\mathcal{Z})$ is a pair $(r_X \times r_Y, z)$, where $r_X \subseteq \mathcal{X}$, $r_Y \subseteq \mathcal{Y}$ and $z \in \mathcal{Z}$. If $t = (r_X \times r_Y, z)$, then we let $\mathcal{X}_t, \mathcal{Y}_t$, and z_t denote r_X, r_Y and z respectively. We say $(x,y) \in t$ if and only if $x \in \mathcal{X}_t$ and $y \in \mathcal{Y}_t$. The set of all tiles for $(\mathcal{X},\mathcal{Y},\mathcal{Z})$ is denoted by $\mathcal{T}(\mathcal{X},\mathcal{Y},\mathcal{Z})$ or simply $\mathcal{T}$ (if $\mathcal{X},\mathcal{Y},\mathcal{Z}$ are clear from the context).

For a relation $f : \mathcal{X} \times \mathcal{Y} \to 2^{\mathcal{Z}}$ and probability of error $\mathcal{E} : \mathcal{X} \times \mathcal{Y} \to [0,1]$, the partition complexity [14] is defined as follows:[5]

$$\text{prt}(f,\mathcal{E}) = \min_{w:\mathcal{T}\to[0,1]} \sum_{t\in\mathcal{T}} w(t) \qquad \text{subject to}$$

$$\sum_{t\in\mathcal{T}:(x,y)\in t} w(t) = 1, \qquad \forall (x,y) \in \mathcal{X}\times\mathcal{Y} \tag{1}$$

$$\sum_{\substack{t\in\mathcal{T}:(x,y)\in t, \\ z_t \in f(x,y)}} w(t) \geq 1 - \mathcal{E}(x,y), \qquad \forall (x,y) \in \mathcal{X}\times\mathcal{Y}. \tag{2}$$

For a weight function w as above, we define the error function as

$$\text{err}_{f,w}(x,y) = \sum_{\substack{t\in\mathcal{T}:(x,y)\in t, \\ z_t \notin f(x,y)}} w(t);$$

then the condition (2) can be written as a condition on this error function: $\text{err}_{f,w} \leq \mathcal{E}$.

[5] The definition presented in [14] is slightly more restrictive in the kind of relations and error functions considered.

The relaxed partition complexity [18] relaxes the equality constraint in (1) to an inequality. Further, the error function is restricted to be a constant function given by $\mathcal{E}(x,y) = \varepsilon$. Specifically, for a relation f and a constant $0 \le \varepsilon \le 1$,

$$\overline{\mathrm{prt}}(f,\varepsilon) = \min_{w:\mathcal{T}\to[0,1]} \sum_{t\in\mathcal{T}} w(t) \quad \text{subject to}$$

$$\sum_{t\in\mathcal{T}:(x,y)\in t} w(t) \le 1, \qquad \forall (x,y)\in\mathcal{X}\times\mathcal{Y} \tag{3}$$

$$\sum_{\substack{t\in\mathcal{T}:(x,y)\in t,\\ z_t\in f(x,y)}} w(t) \ge 1-\varepsilon, \qquad \forall (x,y)\in\mathcal{X}\times\mathcal{Y}. \tag{4}$$

The distributional form of relaxed partition complexity is defined for a distribution μ and $\varepsilon \in [0,1]$ as follows:

$$\overline{\mathrm{prt}}^{\mu}(f,\varepsilon) = \min_{w:\mathcal{T}\to[0,1]} \sum_{t\in\mathcal{T}} w(t) \quad \text{subject to} \quad \begin{array}{l} \forall (x,y)\in\mathcal{X}\times\mathcal{Y} \sum\limits_{t\in\mathcal{T}:(x,y)\in t} w(t) \le 1, \\ \sum\limits_{x,y}\mu(x,y) \sum\limits_{\substack{t\in\mathcal{T}:(x,y)\in t,\\ z_t\in f(x,y)}} w(t) \ge 1-\varepsilon. \end{array}$$

For a weight function w as above and a distribution μ over $\mathcal{X}\times\mathcal{Y}$, we write $\overline{\mathrm{err}}^{\mu}_{f,w}$ for $1 - \sum_{x,y}\mu(x,y)\sum_{\substack{t\in\mathcal{T}:(x,y)\in t,\\ z_t\in f(x,y)}} w(t)$; so the second condition can be written as $\overline{\mathrm{err}}^{\mu}_{f,w} \le \varepsilon$. As shown in [18], $\overline{\mathrm{prt}}(f,\varepsilon) = \max_{\mu}\overline{\mathrm{prt}}^{\mu}(f,\varepsilon)$.

3 Rényi Information Complexity and Pseudotranscripts

In this section we define our new complexity measures.

Rényi information complexity. For a pair of random variables (A,B) over $\mathcal{A}\times\mathcal{B}$, Rényi mutual information of order ∞ is defined as (see, e.g., [25])

$$I_\infty(A;B) = \log\left(\sum_{b\in\mathcal{B}} \max_{a\in\mathcal{A}:\mathbf{p}_A(a)>0} \mathbf{p}_{B|A}(b|a)\right).$$

For a protocol π and an input distribution $\mathbf{p}_{X,Y}$, we will call $I_\infty(X,Y;\Pi)$ the Rényi information cost. *Rényi information complexity* $IC_\infty(f,\mathcal{E})$ is defined as the smallest worst-case (over input distributions) Rényi information cost of any protocol which has a probability of error at most $\mathcal{E}(x,y), x\in\mathcal{X}, y\in\mathcal{Y}$.

$$IC_\infty(f,\mathcal{E}) = \inf_{\substack{\text{protocol } \pi:\\ \mathrm{err}_{f,\pi}\le\mathcal{E}}} \max_{\mathbf{p}_{X,Y}} I_\infty(X,Y;\Pi).$$

Note the above definition is identical to the definition of $IC(f,\mathcal{E})$ except that I_∞ is used in place of mutual information I. It is easy to see that the inner maximization above is obtained by any input distribution $\mathbf{p}_{X,Y}$ with full support. Hence, we may equivalently write

$$IC_\infty(f,\mathcal{E}) = \inf_{\substack{\text{protocol } \pi:\\ \mathrm{err}_{f,\pi}\le\mathcal{E}}} I_\infty(X,Y:\Pi),$$

where we define $I_\infty(A:B)$ which is a function only of $\mathbf{p}_{B|A}$ as

$$I_\infty(A:B) = \log\left(\sum_{b\in\mathcal{B}} \max_{a\in\mathcal{A}} \mathbf{p}_{B|A}(b|a)\right).$$

▶ **Theorem 1.** *$IC(f,\mathcal{E}) \le IC_\infty(f,\mathcal{E}) \le R(f,\mathcal{E})$.*

Proof. The inequality $IC(f,\mathcal{E}) \le IC_\infty(f,\mathcal{E})$ follows from $I(X,Y;\Pi) \le I_\infty(X,Y;\Pi)$, which in turn follows from the monotonicity of α-mutual information [13, Theorem 4(b)] (see the full version for a self-contained proof).

The proof of $IC_\infty(f,\mathcal{E}) \le R(f,\mathcal{E})$ is simple. Consider any public-coin protocol π. Let $\Pi = (\Phi,\Psi)$ where Φ represents the public-coins and Ψ the transcript of π. W.l.o.g., Ψ can be considered to be a deterministic function of Φ and the inputs X,Y.[6] We write $\Psi(x,y;\phi)$ to denote the transcript of π on inputs (x,y) and public coins ϕ. Note that $\#\text{bits}(\pi,x,y) = \max_\phi |\Psi(x,y;\phi)|$ (where $|\cdot|$ denotes the length of a bit string). We shall show that $I_\infty(X,Y:\Pi) \le \max_{x,y,\phi} |\Psi(x,y;\phi)|$. This suffices since

$$IC_\infty(f,\mathcal{E}) = \inf_{\substack{\text{protocol } \pi:\\ \text{err}_{f,\pi}\le\mathcal{E}}} I_\infty(X,Y:\Pi). \qquad R(f,\mathcal{E}) = \inf_{\substack{\text{protocol } \pi:\\ \text{err}_{f,\pi}\le\mathcal{E}}} \max_{x,y,\phi} |\Psi(x,y;\phi)|.$$

Note that $\mathbf{p}_{\Phi\Psi|XY}(\phi,\psi|x,y) = \mathbf{p}_\Phi(\phi)\mathbf{p}_{\Psi|\Phi XY}(\psi|\phi,x,y)$. Then,

$$\begin{aligned} I_\infty(X,Y:\Phi,\Psi) &= \log \sum_\phi \mathbf{p}_\Phi(\phi) \sum_\psi \max_{x,y} \mathbf{p}_{\Psi|\Phi XY}(\psi|\phi,x,y) \\ &\le \log \max_\phi \sum_\psi \max_{x,y} \mathbf{p}_{\Psi|\Phi XY}(\psi|\phi,x,y) \\ &= \max_\phi \log|\{\psi : \exists(x,y) \text{ s.t. } \psi = \Psi(x,y;\phi)\}| \le \max_{x,y,\phi} |\Psi(x,y;\phi)|. \end{aligned}$$ ◀

Pseudotranscript and pseudo-information complexities. A random variable Q defined on an alphabet $\mathcal{Q}$ and jointly distributed with the inputs X,Y is said to be a *pseudotranscript* if $\mathbf{p}_{Q|X,Y}$ satisfies the following *factorization condition:*

$$\mathbf{p}_{Q|X,Y}(q|x,y) = \alpha(q,x)\beta(q,y), \quad \forall q\in\mathcal{Q}, x\in\mathcal{X}, y\in\mathcal{Y},$$

for some pair of functions $\alpha : \mathcal{Q}\times\mathcal{X} \to \mathbb{R}^+$ and $\beta : \mathcal{Q}\times\mathcal{Y} \to \mathbb{R}^+$. In addition, we will require that Q defines an output, i.e., for each q there is an associated $z_q \in \mathcal{Z}$.

For any protocol π, clearly, Π, which is composed of the public-coins and the transcript, is a pseudotranscript.[7] For a pseudotranscript Q, the probability of error is defined analogously to that for a protocol as

$$\text{err}_{f,Q}(x,y) = \Pr[z_Q \notin f(x,y) | (X,Y) = (x,y)].$$

---

[6] Any protocol using private randomness can be transformed to one with only public randomness, by including the private coins as part of the public-coins, without changing the number of bits communicated. Further, this can only increase the quantity $I_\infty(X,Y;\Pi)$. Hence, it is enough to prove the inequality after carrying out this transformation.

[7] $Q = \Pi$ satisfies the factorization condition, as in that case, for $q = (\phi, m_1, \cdots, m_t)$, $\Pr[q|x,y] = \alpha(q,x)\cdot\beta(q,y)$, where say, $\alpha(q,x) = \Pr[\phi]\cdot\Pi_{\text{odd}\,i}\Pr[m_i \mid \phi, m_1, \cdots, m_{i-1}, x]$, and $\beta(q,y) = \Pi_{\text{even}\,i}\Pr[m_i \mid \phi, m_1, \cdots, m_{i-1}, y]$. Also, we can associate the output of the protocol, which we insisted must be the same for both parties for a valid protocol, as the corresponding output z_Q. Though the output of the parties could in principle depend on the local input and local randomness, the factorization condition and the requirement that the outputs agree together imply that the output can be unambiguously determined from the transcript together with the public-coins.

We define the following "pseudo-quantities" corresponding to IC_∞ and IC where Π is replaced by pseudotranscripts:

$$\mathrm{p}IC_\infty(f,\mathcal{E}) = \inf_{\substack{\text{pseudotranscript } Q:\\ \mathrm{err}_{f,Q}\le\mathcal{E}}} \max_{\mathbf{p}_{X,Y}} I_\infty(X,Y;Q) = \inf_{\substack{\text{pseudotranscript } Q:\\ \mathrm{err}_{f,Q}\le\mathcal{E}}} I_\infty(X,Y:Q)$$

$$\mathrm{p}IC(f,\mathcal{E}) = \inf_{\substack{\text{pseudotranscript } Q:\\ \mathrm{err}_{f,Q}\le\mathcal{E}}} \max_{\mathbf{p}_{X,Y}} I(X,Y;Q).$$

▶ **Observation 2.** *Since, for any protocol, its transcript is a pseudotranscript as well, we have* $\mathrm{p}IC_\infty(f,\mathcal{E}) \le IC_\infty(f,\mathcal{E})$ *and* $\mathrm{p}IC(f,\mathcal{E}) \le IC(f,\mathcal{E})$. *Furthermore, since* $I(A;B) \le I_\infty(A;B)$, *we also have* $\mathrm{p}IC(f,\mathcal{E}) \le \mathrm{p}IC_\infty(f,\mathcal{E})$.

4 $\mathrm{p}IC_\infty$ Equals the Partition Bound

▶ **Theorem 3.** *For any relation* $f : \mathcal{X}\times\mathcal{Y} \to 2^{\mathcal{Z}}$ *and error function* $\mathcal{E}$, $\mathrm{p}IC_\infty(f,\mathcal{E}) = \log \mathrm{prt}(f,\mathcal{E})$.

We prove $\mathrm{p}IC_\infty(f,\mathcal{E}) \le \log\mathrm{prt}(f,\mathcal{E})$ and $\mathrm{p}IC_\infty(f,\mathcal{E}) \ge \log\mathrm{prt}(f,\mathcal{E})$ separately. The first direction is easy and, as shown in the full version, follows by considering the tiles in a given partition as the pseudo transcripts. Below, we turn to the other direction.

▶ **Lemma 4.** $\mathrm{p}IC_\infty(f,\mathcal{E}) \ge \log\mathrm{prt}(f,\mathcal{E})$.

Proof sketch. A detailed proof appears in the full version. Below we sketch the analysis, including details which will be useful as a starting point in proving the result in Section 5.

Suppose $\mathbf{p}_{Q|X,Y}$ satisfies the factorization and output consistency conditions, $\mathrm{err}_{f,Q} \le \mathcal{E}$ and $\mathrm{p}IC_\infty(f,\mathcal{E}) = I_\infty(X,Y:Q)$. Let $\mathcal{T}$ be the set of all tiles. To define the partition $w : \mathcal{T} \to [0,1]$, we shall (in (8)) define quantities $\omega_{q,t}$ (for $(q,t) \in \mathcal{Q}\times\mathcal{T}$) and probability distribution $\mathbf{p}_{T|Q,X,Y}$, where T is a random variable over $\mathcal{T}$, such that the following conditions hold.

$$\omega_{q,t} = 0 \qquad \forall (q,t) \in \mathcal{Q}\times\mathcal{T} \text{ s.t. } z_t \ne z_q \tag{5}$$

$$p(q,t|x,y) = \begin{cases} \omega_{q,t} & \text{if } (x,y)\in t \\ 0 & \text{otherwise} \end{cases} \qquad \forall (q,t)\in\mathcal{Q}\times\mathcal{T}, (x,y)\in\mathcal{X}\times\mathcal{Y} \tag{6}$$

$$\log \sum_{q\in\mathcal{Q},t\in\mathcal{T}} \omega_{q,t} = I_\infty(X,Y:Q) \tag{7}$$

Now, if we let $w : \mathcal{T} \to [0,1]$ be defined by $w(t) = \sum_{q\in\mathcal{Q}} \omega_{q,t}$, then it is easy to verify that (1) and (2) hold, and further $\log\mathrm{prt}(f,\mathcal{E}) \le \log\sum_{t\in\mathcal{T}} w(t) = I_\infty(X,Y:Q) = \mathrm{p}IC_\infty(f,\mathcal{E})$.

Thus, to complete the proof, it suffices to define $\mathbf{p}_{T|Q,X,Y}$ and $\omega_{q,t}$ so that the above conditions (5)-(7) are satisfied. Recall that, since Q is a pseudotranscript, $\mathbf{p}_{Q|X,Y}$ satisfies the factorization condition; i.e., we can write

$$\mathbf{p}_{Q|X,Y}(q|x,y) = \alpha(q,x)\beta(q,y), \quad \forall q\in\mathcal{Q}, x\in\mathcal{X}, y\in\mathcal{Y},$$

for some pair of functions $\alpha : \mathcal{Q}\times\mathcal{X} \to \mathbb{R}^+$ and $\beta : \mathcal{Q}\times\mathcal{Y} \to \mathbb{R}^+$. For $q\in\mathcal{Q}$ and $t\in\mathcal{T}$, let

$$\sigma_{q,t} = \min_{x\in\mathcal{X}_t} \alpha(q,x) - \max_{x'\notin\mathcal{X}_t} \alpha(q,x') \quad \text{and} \quad \tau_{q,t} = \min_{y\in\mathcal{Y}_t} \beta(q,y) - \max_{y'\notin\mathcal{Y}_t} \beta(q,y').$$

Above, in defining $\max_{x' \notin \mathcal{X}_t}$, if no such x' exists – i.e., $\mathcal{X}_t = \mathcal{X}$ – we take the maximum to be 0 (and similarly for $\max_{y' \notin \mathcal{Y}_t}$). Now, let

$$\begin{aligned} \mathcal{T}_q &= \{t \in \mathcal{T} \mid \sigma_{q,t} > 0, \tau_{q,t} > 0, \text{ and } z_q = z_t\} \\ \omega_{q,t} &= \begin{cases} \sigma_{q,t} \cdot \tau_{q,t} & \text{if } t \in \mathcal{T}_q \\ 0 & \text{if } t \notin \mathcal{T}_q. \end{cases} \\ p(t|x,y,q) &= \begin{cases} \sigma_{q,t} \cdot \tau_{q,t} \cdot \frac{1}{p(q|x,y)} & \text{if } (x,y) \in t, t \in \mathcal{T}_q \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{8}$$

The fact that $\mathbf{p}_{T|X,Y,Q}$ is a valid probability distribution and that conditions (5)-(7) are satisfied are proven in the full version, using Claim 5 below. This completes the proof of Lemma 4. ◀

▶ **Claim 5.** *For any $q \in \mathcal{Q}$ and $(x,y) \in \mathcal{X} \times \mathcal{Y}$, $\sum_{t \in \mathcal{T}_q : (x,y) \in t} \sigma_{q,t} \cdot \tau_{q,t} = p(q|x,y)$.*

Proof. Fix $q \in \mathcal{Q}$. Let $\mathcal{X} = \{x_1, \cdots, x_M\}$, such that $\alpha(q, x_i) \geq \alpha(q, x_{i-1})$ for all $i \in [1, M]$; for notational convenience, we also define a dummy x_0 with $\alpha(q, x_0) = 0$. Define $y_0, y_1, \cdots, y_N$ similarly for β, where $N = |\mathcal{Y}|$. Let $t_{ij} = (\mathcal{X}_i \times \mathcal{Y}_j, z_q)$ for $(i,j) \in [M] \times [N]$, where $\mathcal{X}_i = \{x_i, \cdots, x_M\}$, $\mathcal{Y}_j = \{y_j, \cdots, y_N\}$. Then,

$$\mathcal{T}_q = \{t_{ij} \mid (i,j) \in [M] \times [N], \alpha(q, x_i) > \alpha(q, x_{i-1}), \beta(q, y_j) > \beta(q, y_{j-1})\}.$$

Consider an arbitrary $(x,y) \in \mathcal{X} \times \mathcal{Y}$. Let (i^*, j^*) be indices such that $(x,y) = (x_{i^*}, y_{j^*})$ in the above ordering. Note that $(x_{i^*}, y_{j^*}) \in t_{ij}$ if and only if $1 \leq i \leq i^*$ and $1 \leq j \leq j^*$. Also notice that for all $(i,j) \in [M] \times [N]$, if $t_{ij} \notin \mathcal{T}_q$, then $\sigma_{q,t_{ij}}, \tau_{q,t_{ij}} = 0$.

$$\begin{aligned} \sum_{t \in \mathcal{T}_q : (x_{i^*}, y_{i^*}) \in t} \sigma_{q,t} \cdot \tau_{q,t} &= \sum_{i=1}^{i^*} \sum_{j=1}^{j^*} \sigma_{q,t_{ij}} \cdot \tau_{q,t_{ij}} \\ &= \sum_{i=1}^{i^*} (\alpha(q, x_i) - \alpha(q, x_{i-1})) \cdot \sum_{j=1}^{j^*} (\beta(q, y_j) - \beta(q, y_{j-1})) \\ &= \alpha(q, x_{i^*}) \cdot \beta(q, y_{j^*}) = p(q|x_{i^*}, y_{j^*}) \end{aligned}$$

as was required to prove. ◀

5 pIC Subsumes Relaxed Partition Bound

▶ **Theorem 6.** *For any relation $f : \mathcal{X} \times \mathcal{Y} \to 2^{\mathcal{Z}}$ and constants $\varepsilon, \delta \in [0,1]$,*

$$\mathrm{p}IC(f, \varepsilon) \geq \delta \log \overline{\mathrm{prt}}(f, \varepsilon + \delta) - (\delta \log \log(|\mathcal{X}||\mathcal{Y}|) + 3).$$

We prove this theorem in the full version. Below we summarize the main ideas.

Proof sketch. It is enough to show that, given a distribution $\mathbf{p}_{XY} = \mu$ over $\mathcal{X} \times \mathcal{Y}$, and pseudotranscript Q such that $\mathrm{err}_{f,Q} \leq \varepsilon$, there is a partition which demonstrates that $\log \overline{\mathrm{prt}}^{\mu}(f, \varepsilon + \delta) \lesssim I(X,Y;Q)/\delta$.

The proof uses the construction from the proof of Lemma 4, and modifies it carefully. Specifically, we define $\mathbf{p}_{T|Q,X,Y}$ and $\omega_{q,t}$ as in Equation 8. Recall that we originally defined w as $w(t) = \sum_{q \in \mathcal{Q}} \omega_{q,t}$. Our plan now is to remove some of the weight on the tiles so that the log of the sum can be bounded by (roughly) $I(X,Y;Q)/\delta$ as opposed to $I_\infty(X,Y:Q)$.

Towards this, we shall define a set $\mathcal{B}$ of "bad" pairs $(q,t) \in \mathcal{Q} \times \mathcal{T}$ whose weights $\omega_{q,t}$ will not be counted towards our new weight function $w'(t)$:

$$w'(t) = \sum_{(q,t)\in(\mathcal{Q}\times\mathcal{T})\setminus\mathcal{B}} \omega_{q,t}, \qquad \forall t \in \mathcal{T}.$$

The crux of the proof is to define the set $\mathcal{B}$ such that the weight removed $\sum_{(q,t)\in\mathcal{B}} p(q,t)$ is below δ (it manifests as the increase in error), while keeping $\sum_{(q,t)\notin\mathcal{B}} \omega_{q,t}$ (approximately) below $I(X,Y;Q)/\delta$. We show that the following choice of $\mathcal{B}$ has both these properties:

$$\mathcal{B} = \{(q,t) \in \mathcal{Q} \times \mathcal{T} \mid \hat{\alpha}(q,t) \cdot \hat{\beta}(q,t) \geq \theta_q, \}$$

where $\hat{\alpha}(q,t) = \min_{(x,y)\in t} \alpha(q,x)$ and $\hat{\beta}(q,t) = \min_{(x,y)\in t} \beta(q,y)$ and θ_q is an appropriately defined threshold for each $q \in \mathcal{Q}$ (specifically, $\theta_q = p(q)2^{\Delta}$, where $\Delta \approx I(XY;Q)/\delta$).

To upper bound the mass removed, we first write $I(XY;Q) = \sum_{q\in\mathcal{Q},t\in\mathcal{T}} p(q,t)\varphi(q,t)$, where $\varphi(q,t)$ is a quantity that is lower bounded by Δ for all $(q,t) \in \mathcal{B}$. This suggests the possibility of using the Markov inequality to upper bound $\sum_{(q,t)\in\mathcal{B}} p(q,t)$. However, $\varphi(q,t)$ could be negative, and we cannot directly use the above expression for $I(X,Y;Q)$ in a Markov inequality. However, we show that removing the negative terms from $\sum_{q,t} p(q,t)\varphi(q,t)$ does not increase the sum significantly, which will let us still apply the Markov inequality.

To upper bound $\sum_{(q,t)\notin\mathcal{B}} \omega_{q,t}$, we use a geometric interpretation of $\omega_{q,t}$ and the set $\mathcal{B}$. Fix a $q \in \mathcal{Q}$. Then, using the notation in the proof of Claim 5, for each $(i,j) \in [M] \times [N]$, the tile t_{ij} will be represented by an axis-parallel rectangle on the real plane, R_{ij}, as follows. R_{ij} is defined by its diagonally opposite vertices $(\alpha(q,x_{i-1}), \beta(q,y_{j-1}))$ and $(\alpha(q,x_i), \beta(q,y_j))$. (See Figure 2.) R_{ij} could have zero area. These rectangles tile a rectangular region, without overlapping with each other. Further the area of the rectangle R_{ij} is the same as $\omega_{q,t_{ij}}$. Thus $\sum_{t:(q,t)\notin\mathcal{B}} \omega_{q,t}$ is given by the sum of the areas of the rectangles R_{ij} for which $(q,t_{ij}) \notin \mathcal{B}$. The rectangles R_{ij} that correspond to $(q,t_{ij}) \notin \mathcal{B}$ are those which have their top-right vertex (i.e., $(\alpha(q,x_i), \beta(q,y_j))$) fall "below" the hyperbola defined by the equation $xy = \theta_q$. Thus if $(q,t_{ij}) \notin \mathcal{B}$, then the entire rectangle R_{ij} is below the hyperbola $xy = \theta_q$. Hence the sum of their areas is upper-bounded by the area within R that is under this hyperbola, where R is the rectangle with diagonally opposite vertices $(0,0)$ and $(\max_{x\in\mathcal{X}} \alpha(q,x), \max_{y\in\mathcal{Y}} \beta(q,y))$. A calculation yields the required bound. ◀

6 Extensions

We may define internal information complexity associated with pseudotranscripts as

$$\mathrm{p}IC^{\mathrm{int}}(f,\mathcal{E}) = \inf_{\substack{\text{pseudotranscript } Q\colon \mathbf{p}_{X,Y} \\ \mathrm{err}_{f,Q}\leq\mathcal{E}}} \max I(X;Q|Y) + I(Y;Q|X).$$

It is easy to show that for the usual notion of information complexity (defined with respect to protocols), $IC^{\mathrm{int}}(f,\mathcal{E}) \leq IC(f,\mathcal{E})$. The proof hinges on the fact that for any protocol π and distribution $\mathbf{p}_{X,Y}$ on the inputs, the resulting Π satisfies the condition $I(X;Y) \geq I(X;Y|\Pi)$. However, it is unclear whether $\mathrm{p}IC^{\mathrm{int}}(f,\mathcal{E})$ is necessarily upper bounded by $\mathrm{p}IC(f,\mathcal{E})$. Below we define a slightly refined notion of pseudotranscripts so that information complexities defined with respect to that maintain the above inequality.

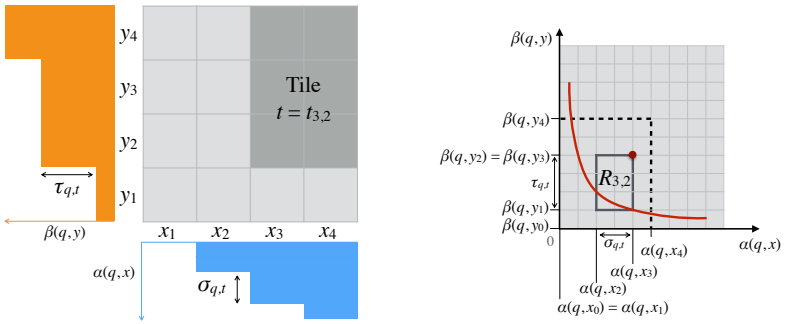


Figure 2 Illustration of the geometric interpretation of $\mathcal{B}$ used in the proof of Theorem 6. The left figure shows the domain $\mathcal{X} \times \mathcal{Y}$ and plots $\alpha(q,x)$ and $\beta(q,y)$ against x and y, which are sorted in the order of increasing $\alpha(q,x)$ and $\beta(q,y)$, respectively (for some fixed q). It also shows a tile $t = t_{3,2}$ in $\mathcal{T}_q$, and indicates the values $\sigma_{q,t}$ and $\tau_{q,t}$. The right figure shows the alternate representation of the tile $t_{3,2}$ using the rectangular region $R_{3,2}$. The area of $R_{3,2}$ equals $\omega_{q,t_{3,2}} = \sigma_{q,t_{3,2}} \cdot \tau_{q,t_{3,2}}$. A hyperbola corresponding to a threshold θ_q is also shown. Since the upper-right vertex of $R_{3,2}$, namely the point $(\alpha(q,x_3), \beta(q,y_2))$ is above the hyperbola, $(q,t_{3,2}) \in \mathcal{B}$. The area within the dotted rectangle that is under the hyperbola gives an upper-bound on the sum of areas of all rectangles under the hyperbola.

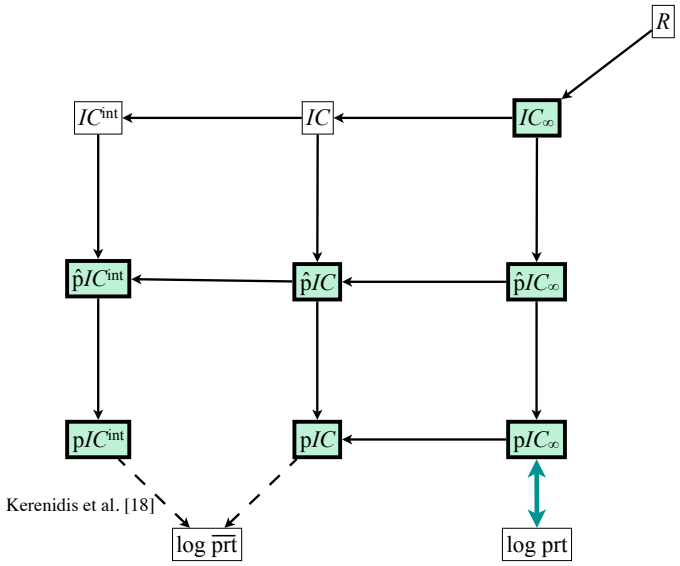


Figure 3 An extended version of the map in Figure 1, including the complexity measures in Section 6.

Refined pseudotranscripts and corresponding information complexities. A pseudotranscript Q given by $\mathbf{p}_{Q|X,Y}$ is called a *refined pseudotranscript* if, for any distribution $\mathbf{p}_{X,Y}$ on the inputs, it holds that $I(X;Y) \geq I(X;Y|Q)$. It is easy to show that for any protocol π and distribution $\mathbf{p}_{X,Y}$ on the inputs, the resulting Π satisfies the above condition and, hence, Π is a refined pseudotranscript.

Analogous to our definition of pseudo-information complexities, we define information complexities with respect to refined pseudotranscripts

$$\hat{\mathrm{p}}IC_\infty(f,\mathcal{E}) = \inf_{\substack{\text{refined pseudotranscript } Q: \\ \mathrm{err}_{f,Q} \leq \mathcal{E}}} I_\infty(X,Y:Q)$$

$$\hat{\mathrm{p}}IC(f,\mathcal{E}) = \inf_{\substack{\text{refined pseudotranscript } Q: \\ \mathrm{err}_{f,Q} \leq \mathcal{E}}} \max_{\mathbf{p}_{X,Y}} I(X,Y;Q)$$

$$\hat{\mathrm{p}}IC^{\mathrm{int}}(f,\mathcal{E}) = \inf_{\substack{\text{refined pseudotranscript } Q: \\ \mathrm{err}_{f,Q} \leq \mathcal{E}}} \max_{\mathbf{p}_{X,Y}} I(X;Q|Y) + I(Y;Q|X).$$

Figure 3 shows the relationship between the different complexities we consider. Since, for any protocol, its transcript is a refined pseudotranscript and refined pseudotranscripts are also pseudotranscripts, we have $\mathrm{p}\mathtt{X}(f,\mathcal{E}) \leq \hat{\mathrm{p}}\mathtt{X}(f,\mathcal{E}) \leq \mathtt{X}(f,\mathcal{E})$, where $\mathtt{X}$ can be IC_∞, IC or IC^{int}. Furthermore, analogous to $IC^{\mathrm{int}}(f,\mathcal{E}) \leq IC(f,\mathcal{E}) \leq IC_\infty(f,\mathcal{E})$, we have $\hat{\mathrm{p}}IC^{\mathrm{int}}(f,\mathcal{E}) \leq \hat{\mathrm{p}}IC(f,\mathcal{E}) \leq \hat{\mathrm{p}}IC_\infty(f,\mathcal{E})$. Finally, in deriving a lower bound for $IC^{\mathrm{int}}(f,\varepsilon)$ in terms of $\overline{\mathrm{prt}}(f,\varepsilon)$ [18] only relies on the fact that the transcript (along with the public-coins) Π satisfies the factorization condition. Hence, the lower bound of [18] holds with IC^{int} replaced by $\mathrm{p}IC^{\mathrm{int}}$.

Acknowledgments. We gratefully acknowledge Mark Braverman, Prahladh Harsha, Rahul Jain, Anup Rao and the anonymous referees for helpful suggestions and pointers. The research was supported in part by NSF grant 1228856 for the first author, by ITRA, Media Lab Asia, India and by a Ramanujan fellowship from DST, India for the second author.

References

1 Farid M. Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theor. Comput. Sci.*, 157(2):139–159, 1996. doi:10.1016/0304-3975(95)00157-3.

2 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.*, 68(4):702–732, 2004. doi:10.1016/j.jcss.2003.11.006.

3 Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013. doi:10.1137/100811969.

4 Mark Braverman. Interactive information complexity. In *STOC*, pages 505–524, 2012. doi:10.1145/2213977.2214025.

5 Mark Braverman and Anup Rao. Information equals amortized communication. In *FOCS*, pages 748–757, 2011. doi:10.1109/FOCS.2011.86.

6 Mark Braverman and Omri Weinstein. A discrepancy lower bound for information complexity. In *APPROX-RANDOM*, pages 459–470, 2012. doi:10.1007/978-3-642-32512-0_39.

7 Amit Chakrabarti, Ranganath Kondapally, and Zhenghui Wang. Information complexity versus corruption and applications to orthogonality and gap-hamming. In *APPROX-RANDOM*, pages 483–494, 2012. doi:10.1007/978-3-642-32512-0_41.

8 Amit Chakrabarti, Yaoyun Shi, Anthony Wirth, and Andrew Chi-Chih Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *FOCS*, pages 270–278, 2001. doi:10.1109/SFCS.2001.959901.

9 Lila Fontes, Rahul Jain, Iordanis Kerenidis, Sophie Laplante, Mathieu Laurière, and Jérémie Roland. Relative discrepancy does not separate information and communication complexity. In *ICALP*, pages 506–516, 2015.

10 Anat Ganor, Gillat Kol, and Ran Raz. Exponential separation of information and communication. In *FOCS*, pages 176–185, 2014.

11 Anat Ganor, Gillat Kol, and Ran Raz. Exponential separation of communication and external information. In *Electronic Colloquium on Computational Complexity (ECCC)*, 2015.

12 Prahladh Harsha, Rahul Jain, David McAllester, and Jaikumar Radhakrishnan. The communication complexity of correlation. *IEEE Transactions on Information Theory*, 56(1):438–449, 2010. doi:10.1109/TIT.2009.2034824.

13 Siu-Wai Ho and Sergio Verdú. Convexity/concavity of Rényi entropy and α-mutual information. In *Information Theory (ISIT), 2015 IEEE International Symposium on*, pages 745–749, 2015.

14 Rahul Jain and Hartmut Klauck. The partition bound for classical communication complexity and query complexity. In *IEEE Conference on Computational Complexity*, pages 247–258, 2010.

15 Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. A direct sum theorem in communication complexity via message compression. In *ICALP*, pages 300–315, 2003. doi:10.1007/3-540-45061-0_26.

16 Rahul Jain, Jaikumar Radhakrishnan, and Pranab Sen. Prior entanglement, message compression and privacy in quantum communication. In *IEEE Conference on Computational Complexity*, pages 285–296, 2005. doi:10.1109/CCC.2005.24.

17 T. S. Jayram, Ravi Kumar, and D. Sivakumar. Two applications of information complexity. In *STOC*, pages 673–682, 2003. doi:10.1145/780542.780640.

18 Iordanis Kerenidis, Sophie Laplante, Virginie Lerays, Jérémie Roland, and David Xiao. Lower bounds on information complexity via zero-communication protocols and applications. In *FOCS*, pages 500–509, 2012.

19 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, New York, 1997.

20 Stephen J Ponzio, Jaikumar Radhakrishnan, and Srinivasan Venkatesh. The communication complexity of pointer chasing. *Journal of Computer and System Sciences*, 62(2):323–355, 2001.

21 Manoj Prabhakaran and Vinod M. Prabhakaran. Tension bounds for information complexity. *CoRR*, abs/1408.6285, 2014. URL: http://arxiv.org/abs/1408.6285.

22 Alfred Rényi. On measures of information and entropy. In *Proceedings of the 4th Berkeley Symposium on Mathematics, Statistics and Probability*, pages 547–561, 1960.

23 Michael E. Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *STOC*, pages 360–369, 2002. doi:10.1145/509907.509963.

24 R. Sibson. Information radius. *Z. Wahrscheinlichkeitstheorie und Verw. Geb.*, 14:149–161, 1969.

25 Sergio Verdú. α-mutual information. In *Information Theory and Applications Workshop (ITA)*, 2015.

26 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *STOC*, pages 209–213, 1979. doi:10.1145/800135.804414.

27 Moshe Zakai and Jacob Ziv. A generalization of the rate-distortion theory and applications. In *Information Theory New Trends and Open Problems*, pages 87–123. Springer, 1975.

28 Jacob Ziv and Moshe Zakai. On functionals satisfying a data-processing theorem. *Information Theory, IEEE Transactions on*, 19(3):275–283, 1973.

On Isoperimetric Profiles and Computational Complexity

Pavel Hrubeš[*1] **and Amir Yehudayoff**[†2]

1 **Institute of Mathematics of CAS, Prague, Czech Republic**
hrubes@math.cas.cz
2 **Department of Mathematics, Technion-IIT, Haifa, Israel**
amir.yehudayoff@gmail.com

Abstract

The isoperimetric profile of a graph is a function that measures, for an integer k, the size of the smallest edge boundary over all sets of vertices of size k. We observe a connection between isoperimetric profiles and computational complexity. We illustrate this connection by an example from communication complexity, but our main result is in algebraic complexity.

We prove a sharp super-polynomial separation between monotone arithmetic circuits and monotone arithmetic branching programs. This shows that the classical simulation of arithmetic circuits by arithmetic branching programs by Valiant, Skyum, Berkowitz, and Rackoff (1983) cannot be improved, as long as it preserves monotonicity.

A key ingredient in the proof is an accurate analysis of the isoperimetric profile of finite full binary trees. We show that the isoperimetric profile of a full binary tree constantly fluctuates between one and almost the depth of the tree.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Monotone computation, separations, communication complexity, isoperimetry

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.89

1 Introduction

Computational complexity theory is about understanding the amount of resources required to compute a given function. One general framework for analyzing the limitations of a given computational device is to partition it to two or more parts and study the interactions between them (see [12, 2] and references trerein). This framework is directly related to communication complexity [24, 15], and appears in the study of branching programs (e.g. [4]), in algebraic complexity theory (e.g. [17, 10]), and more.

In this work, we observe the following phenomenon: the *exact* sizes of the parts in the partition matter. Some functions are "easy" to compute when the sizes can be chosen flexibly, but are "difficult" to compute when the sizes are chosen adversarially. The simplest illustration of this phenomenon comes from communication complexity, and is discussed below. Our most convincing application, however, comes from arithmetic circuit complexity, and forms the bulk of this paper.

* For the first author, the research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Program (FP7/2007-2013) / ERC grant agreement no. 339691. The Institute of Mathematics is supported by RVO:67985840.
† For the second author, the research is supported by the ISF and BSF.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 89; pp. 89:1–89:12

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

1.1 Isoperimetric profile of graphs

The aforementioned phenomenon is related to expansion properties of graphs. For a graph $G = (V(G), E(G))$ and $A \subseteq V(G)$, the size of the edge boundary of A is

$$e_G(A) = \big|\{\{a, b\} \in E(G) : a \in A, b \in B\}\big|,$$

where $B = V(G) \setminus A$. The edge isoperimetric profile of G is the function

$$\mathrm{eip}_G(k) = \min\{e_G(A) : A \subseteq V(G), |A| = k\}.$$

For example, G is connected if and only if $\mathrm{eip}(k) \geq 1$ for every non trivial k. Analyzing the isoperimetric profile of a graph is not a simple task. In fact, even understanding simple properties of the isoperimetric profile in manifolds is difficult (see [16] and references therein).

The high-level connection is as follows. Consider functions defined over some graph G. The vertices are labelled by variables, and the edges represent interactions between them. A partitioning of the variables into two parts yields a partition of the vertices of G. Intuitively, the size of the edge boundary represents the amount of interaction between the parts, and so a large boundary implies high complexity.

1.2 Communication complexity

Communication complexity studies the amount of communication two or more parties need to exchange in order to achieve a common goal. It was initiated by Yao in [24] in the context of distributed computing. For simplicity of the presentation, here we focus on the best-case deterministic two player model that was introduced by Papadimitriou and Sipser [18], and applied to study of very-large-scale integration (for more background and details, see the textbook [15]).

The deterministic communication complexity $D(f)$ of a boolean function $f : \{0,1\}^A \times \{0,1\}^B \to \{0,1\}$ is the minimum number of bits two players need to exchange in order to determine the value of $f(x, y)$, where one player sees $x \in \{0,1\}^A$ and the other sees $y \in \{0,1\}^B$. Let $F : \{0,1\}^V \to \{0,1\}$. Given a partition of V to two sets A, B, we can define a function $F_{A,B} : \{0,1\}^A \times \{0,1\}^B \to \{0,1\}$ by $F_{A,B}(x, y) = F(z)$ where $z|_A = x$ and $z|_B = y$. The best-case deterministic communication complexity with partition-size k of F, denoted $D_k^{\mathrm{best}}(F)$, is the minimum of $D(F_{A,B})$ over all partitions of V to two sets A, B with $|A| = k$.

We observe the following connection between the best-case communication complexity and the edge isoperimetric profile for certain functions. Let $G = (V, E)$ be an undirected graph with n vertices and maximum degree Δ. Consider, e.g., the function $F : \{0,1\}^V \to \{0,1\}$ defined via G as[1]

$$F(z) = \bigoplus_{\{u,v\} \in E} z_u \wedge z_v,$$

where $\oplus$ is addition modulo two. Then, for every $1 \leq k \leq n$, it holds that[2]

$$\Omega(\mathrm{eip}_G(k)/\Delta^2) \leq D_k^{\mathrm{best}}(F) \leq O(\mathrm{eip}_G(k)).$$

[1] The definition of F is inspired by a private communication with Avi Wigderson following [19].
[2] In this text, big O and Ω notation means "up to a constant multiplicative factor".

The upper bound is obtained by partitioning the variables according to the minimizer of the edge boundary, noting that the players need only to exchange inputs that appear on edges of the boundary. The lower bound holds for the following reason. For every partition A, B of the inputs with $|A| = k$, the edge boundary of A contains an induced matching of size $m \geq \Omega(\mathrm{eip}_G(k)/\Delta^2)$. Setting the variables not participating in the matching to zero, the lower bound follows from standard lower bounds on the inner product function, $\bigoplus_{i\in[m]} x_i \wedge y_i$.

In other words, the best-case communication complexity of F is determined by the isoperimetric profile of G, as long as G has constant degree. So, if the isoperimetric profile of G changes dramatically with k, then so does $D^{\mathrm{best}}_k(F)$. For example, there are functions F so that $D^{\mathrm{best}}_{n/2}(F) = O(1)$ whereas $D^{\mathrm{best}}_{n/3}(F) = \Omega(n)$.

1.3 Algebraic complexity

Algebraic complexity theory studies the complexity of computing polynomials over a field. For more background and details, see [7, 5, 21] and references within. We start by outlining definitions of the models we consider.

The most general model of computation in this area is that of *arithmetic circuit*. An arithmetic circuit is a directed acyclic graph with fan-in either zero or two. Vertices of fan-in zero are labelled by a field element or a variable. Vertices of fan-in two are labelled by either $+$ or $\times$. Every vertex in an arithmetic circuit computes a polynomial in the obvious way. A weaker model is an *arithmetic formula*: it is a circuit whose underlying graph is a tree. Another model is the *algebraic branching program*, ABP. An ABP is a directed acyclic graph with two special vertices $v_{\mathrm{start}}, v_{\mathrm{end}}$. Each edge e in it is labelled by $L(e)$ which is a variable times a field element. The program computes f, which is the following sum over all directed paths γ from v_{start} to v_{end}:

$$f = \sum_{\gamma : v_{\mathrm{start}} \to v_{\mathrm{end}}} \prod_{e\in\gamma} L(e).$$

The computation performed by ABPs corresponds to the iterated multiplication of matrices. The size parameter in each of the three models is the number of edges in the underlying graph.

We now discuss reductions between these models. Formulas can be losslessly simulated by an ABP, which in turn can be losslessly simulated by a circuit. In the opposite direction, Hyafil [11] showed that every circuit of size s computing a polynomial of degree r can be simulated by a formula of size $2^{O(\log(s)\log(r))}$. Valiant, Skyum, Berkowitz and Rackoff [23] significantly strengthened this result. In their seminal work, they showed that an arithmetic circuit of size s computing a polynomial of degree r can be simulated by a circuit of size $\mathrm{poly}(r, s)$ and depth $O(\log(r)\log(s))$. This result was later used in a sequence of works – Agrawal and Vinay [1], Koiran [13], and Tavenas [22] – to prove non-trivial simulations of circuits by circuits of depth four (and unbounded fan-in). This depth reductions sprung a long sequence of works on lower bounds for constant depth circuits (see [9, 14, 6] are references within).

An intriguing question is whether the simulations discussed above are sharp. It is possible they can be significantly improved. However, since we do not have strong enough methods for proving lower bounds for these models of computation, proving that the reductions are sharp is currently beyond us.

We therefore address this question in the restricted setting of monotone computations. A monotone arithmetic computation is a computation over the real numbers that uses only

non-negative numbers. The aforementioned reductions of Hyafil and Valiant et al. transform a monotone device to a monotone device. We call such reductions monotone.

Since we know how to prove lower bounds for monotone computation, we can in fact show that in the category of monotone reductions the reductions above are indeed sharp. Indeed, the reduction between ABPs and formulas is sharp since Shamir and Snir [20] proved that in order to multiply d matrices of size $n \times n$, we need a monotone formula of size $n^{\Omega(\log d)}$, but on the other hand this can be done via a monotone ABP of size $\mathrm{poly}(n, d)$. The reduction from circuits to ABPs was not previously studied. In this paper, we prove:

▶ **Theorem 1.** *There is a multilinear n-variate polynomial f with zero-one coefficients so that the following hold:*

1. *The polynomial f can be computed by a monotone arithmetic circuit of size $\mathrm{poly}(n)$.*
2. *Every monotone ABP computing f has size at least $2^{\Omega(\log^2(n))}$.*

The simulation of [23] shows that the lower bound in the theorem is tight, up to a constant in the exponent. Stated differently, the theorem shows that any monotone reduction from circuits to ABPs must incur a super-polynomial blowup. To the best of our knowledge, this is the first separation between algebraic branching programs and circuits. Prior to this work, it was conceivable that the monotone construction of [23] or other monotone variants of it can simulate a circuit by an ABP with only a polynomial increase in size.

It is worth mentioning that Gupta, Kamath, Kayal and Saptharishi [8] showed how to simulate a circuit of size s and degree r over n variables by a depth three circuit of size $\exp(O(\sqrt{d \log(d) \log(n) \log(s)}))$, over fields of characteristic 0. Their simulation uses the reductions to depth four from [1, 13, 22] mentioned above, together with two other reductions. Most relevant to our work is that their simulation is not monotone. In fact, as they mention, a simulation to depth three achieving the same parameters can not be monotone.

We now briefly discuss the proof. All lower bounds we know of for monotone algebraic computation are based on combinatorial problems that are related to counting monomials. The lower bounds for circuits use the following structure, which is standard by now (see e.g. [20, 21, 10]). We denote by $\deg(f)$ the total degree of f.

▶ **Lemma 2.** *If f is a homogeneous polynomial of degree r that can be computed by a monotone circuit of size s, then for every integer $2 \leq k \leq r$, we can write*

$$f = \sum_{i=1}^{s} h_i g_i \tag{1}$$

where for each i, both h_i, g_i are homogeneous, monotone, and of degrees $k/3 \leq \deg(h_i) < 2k/3$ and $\deg(g_i) = r - \deg(h_i)$.

A typical monotone lower bound proceeds by showing that the number of monomials in each $h_i g_i$ is small and so s must be large. To separate monotone circuits from ABPs, we must find a polynomial that has a small circuit and use some other property of ABPs. We use the following.

▶ **Lemma 3.** *If f is a homogeneous polynomial of degree r that can be computed by a monotone ABP of size s, then for every integer $0 \leq k \leq r$, we can write*

$$f = \sum_{i=1}^{s} h_i g_i \tag{2}$$

where for each i, both h_i, g_i are homogeneous, monotone, and of degrees $\deg(h_i) = k$ and $\deg(g_i) = r - k$.

Notice the similarities between the two lemmas. The only difference between circuits and ABPs is that circuits are slightly more flexible about the degrees of h_i, g_i in (1), whereas ABPs are not. We exploit this weakness to prove the separation. Motivated by Section 1.2, the polynomial f in Theorem 1 will be defined over an underlying graph G. The setting will be such that for each k, large $\mathrm{eip}_G(k)$ implies a super-polynomial lower bound on s in (2), and hence gives a lower bound on the ABP-size. On the other hand, the graph must be chosen so that f is easy to compute via a monotone circuit. If G is too complex, such as an expander graph, then we do not expect to prove meaningful upper bounds. In fact, in order to circumvent the lower bound implied by (1), the value of $\mathrm{eip}_G(k)$ must be small for many integers k. It turns out that the right graph to choose is the full binary tree, which we discuss next.

1.4 Full binary trees

We analyze the isoperimetric profile of full finite binary trees. The case of the infinite binary tree was previously studied by Bharadwaj, Chandran and Das in [3], where it was shown to be related to meta-Fibonacci sequences and signed almost binary partitions.

For an integer $d \geq 0$, denote by T_d the full binary tree of depth d. It has precisely 2^d leaves and $2^{d+1} - 1$ vertices. It is clear that there are many values $k \leq |V(T_d)|$ for which $\mathrm{eip}_{T_d}(k) = 1$. It is more surprising that for some k, the value of $\mathrm{eip}_{T_d}(k)$ can be fairly large, almost as large as the depth of T_d. This is the content of the first theorem:

▶ **Theorem 4.** *Let $mip(T_d)$ be the maximum of $\mathrm{eip}_{T_d}(k)$ over all $0 \leq k \leq |V(T_d)|$. Then*

$$\frac{d}{2} - O(\log d) \leq mip(T_d) \leq \frac{d}{2} + O(1)\,.$$

In order to analyze the isoperimetric profile of T_d, we relate it to the binary representation of k. For an integer $k \geq 0$, denote the binary representation of k by

$$B(k) = (B_0(k), B_1(k), B_2(k), \ldots) \in \{0,1\}^{\mathbb{N}}.$$

I.e., $k = \sum_{i=0}^{\infty} B_i(k) 2^i$. Denote by $\mathrm{drops}(k)$ the number of drops in $B(k)$:

$$\mathrm{drops}(k) = |\{i \geq 0 : B_i(k) > B_{i+1}(k)\}|.$$

The number of drops is a quantity that is relatively easy to understand, and as the following theorem shows, it captures the isoperimetric profile of binary trees.

▶ **Theorem 5.** *For every $d \geq 0$ and $0 \leq k < |V(T_d)|$,*

$$\frac{\mathrm{drops}(k)}{2} \leq \mathrm{eip}_{T_d}(k) \leq 2\mathrm{drops}(k)\,.$$

Moreover, the lower bound can be improved to $\mathrm{drops}(k) - O(\log(\mathrm{drops}(k)))$.

The theorem provides an almost optimal description of the isoperimetric profile $\mathrm{eip}_{T_d}(k)$ of T_d. It implies that although $\mathrm{eip}_{T_d}(k) = 1$ for many values of k, for most values of k we have $\mathrm{eip}_{T_d}(k) \geq \Omega(d)$.

We get an explicit choice of k for which $\mathrm{eip}(k)$ is large. For even $d \geq 0$, define

$$\sigma_d = 1 + 4 + 16 + \ldots + 2^d \approx \frac{2}{3} 2^{d+1}, \tag{3}$$

and for odd d define $\sigma_d = \sigma_{d-1}$. The number of drops in $B(\sigma_d)$ is at least $d/2$. The number σ_d can be thought of as the truncation of the binary expansion of $2/3$.

▶ **Corollary 6.** *For every $d \geq 0$, we have $\mathrm{eip}_{T_d}(\sigma_d) \geq d/2 - O(\log d) \geq d/4$.*

1.5 Organization

Section 3 contains the proof of the separation between monotone circuits and ABPs. Section 2 contains the proofs concerning the isoperimetric profile of the full binary tree that are needed in Section 3 (due to space limitations, some of the proofs that are not used in Section 3 will appear in the full version of the text).

2 Binary trees

In order to prove the lower bounds in Theorems 4 and Theorem 5, we will introduce the quantity $\mathrm{sbl}(k)$ which, roughly speaking, measures how far k is from powers of 2. We then relate $\mathrm{eip}_{T_d}(k)$ and $\mathrm{sbl}(k)$,

We say that k is a signed power of two if it is of the form $\pm 2^j$ for some integer $j \geq 0$. For $k \in \mathbb{Z}$, the signed binary length of k, denoted $\mathrm{sbl}(k)$, is the minimum b so that there exist b signed powers of two $k_1, \ldots, k_b$ so that $k = \sum_{i=1}^{b} k_i$, where repetitions are a priori allowed. For example, $\mathrm{sbl}(0) = 0$ and if $d \geq 1$ then

$$\mathrm{sbl}(1 + 2 + 4 + \ldots + 2^d) = \mathrm{sbl}(2^{d+1} - 1) = 2.$$

The following lemma gives a lower bound on $\mathrm{eip}_{T_d}(k)$ in terms of $\mathrm{sbl}(k)$.

▶ **Lemma 7.** *For every $d \geq 0$ and $0 < k \leq |V(T_d)|$,*

$$\mathrm{eip}_{T_d}(k) \geq \mathrm{sbl}(k) - O(\log(\mathrm{sbl}(k)))\,.$$

In addition, if $k < |V(T_d)|$ then $\mathrm{eip}_{T_d}(k) \geq \mathrm{sbl}(k)/2$.

Proof. For an integer k, let $\mathrm{st}(k)$ denote the smallest b such that there exist non-negative integers $j_1, \ldots, j_b$ and $\epsilon_1, \ldots, \epsilon_b \in \{-1, 1\}$ with[3]

$$k = \sum_{j=1}^{b} \epsilon_j \cdot (2^j - 1)\,. \tag{4}$$

Recall that if $j > 0$, then $2^j - 1$ is the number of vertices in T_{j-1}. We will prove the following:

▶ **Claim 8.** *For every $U \subseteq V(T_d)$ not containing the root of T_d, we have $\mathrm{st}(|U|) \leq e_{T_d}(U)$.*

Claim 8 implies that for every $U \subseteq V(T_d)$,

$$\mathrm{st}(|U|) \leq e_{T_d}(U) + 1\,. \tag{5}$$

Indeed, if U contains the root of T_d, then apply Claim 8 to the complement $\bar{U} = V(T_d) \setminus U$. Then $\bar{U}$ does not contain the root of T_d and $e_{T_d}(U) = e_{T_d}(\bar{U}) \geq \mathrm{st}(|\bar{U}|)$. But $|U| = |V(T_d)| - |\bar{U}|$ gives $\mathrm{st}(|U|) \leq \mathrm{st}(|\bar{U}|) + 1$.

Proof of Claim 8. The proof is by induction on d. The claim holds for $d = 0$ as $\mathrm{st}(0) = 0$. Assume $d > 0$. Let v_d be the root of T_d and for a node v, let $T(v)$ be the full subtree of T_d with root v. Let $U \subseteq V(T_d)$ be so that $v_d \notin U$. Let M be the subset of maximal vertices in

[3] This quantity previously appeared in [3].

U. That is, a vertex v is in M if $v \in U$ and the shortest path from v to v_d does not contain a vertex from U. For $v \in M$, let $U(v) = U \cap V(T(v))$. Hence,

$$e_{T_d}(U) = |M| + \sum_{v \in M} e_{T(v)}(U(v)).$$

Since $|U| = \sum_{v \in M} |U(v)|$,

$$\mathrm{st}(|U|) \leq \sum_{v \in M} \mathrm{st}(|U(v)|).$$

For every $v \in M$, the inductive assumption and (5) show that

$$\mathrm{st}(|U(v)|) \leq e_{T(v)}(U(v)) + 1.$$

Overall,

$$\mathrm{st}(|U|) \leq \sum_{v \in M} (1 + e_{T(v)}(U(v))) = |M| + \sum_{v \in M} e_{T(v)}(U(v)) = e_{T_d}(U). \quad \blacktriangleleft$$

To prove the lemma, it remains to estimate $\mathrm{sbl}(k)$ in terms of $\mathrm{st}(k)$. If k is written as in (4) with $b = \mathrm{st}(k)$, we have

$$\mathrm{sbl}(k) \leq \mathrm{sbl}(\sum_{j=1}^{b} \epsilon_j 2^j) + \mathrm{sbl}(\sum_{j=1}^{b} \epsilon_j) \leq \mathrm{st}(k) + \log_2(\mathrm{st}(k) + 1).$$

This gives $\mathrm{st}(k) \geq \mathrm{sbl}(k) - O(\log(\mathrm{sbl}(k)))$ and so (5) gives

$$\mathrm{eip}_{T_d}(k) \geq \mathrm{sbl}(k) - O(\log(\mathrm{sbl}(k)))\,,$$

as required.

Finally, assume that $U \neq V(T_d)$. If U does not contain the root, apply Claim 8 and the estimate $\mathrm{sbl}(k) \leq 2\mathrm{st}(k)$, to obtain $e_{T_d}(U) \geq \mathrm{sbl}(|U|)/2$. If U does contain the root, apply Claim 8 to $\bar{U} = V(T_d) \setminus U$ and note that $\mathrm{sbl}(|U|) \leq 2\mathrm{st}(|\bar{U}|)$ whenever $\bar{U} \neq \emptyset$. $\blacktriangleleft$

The next lemma shows that, up to a constant factor, $\mathrm{sbl}(k)$ and $\mathrm{drops}(k)$ are the same.

▶ **Lemma 9.** *For every $k \geq 0$, we have* $\mathrm{drops}(k) \leq \mathrm{sbl}(k) \leq 2\mathrm{drops}(k)$.

Proof. We start by showing that $\mathrm{sbl}(k) \leq 2\mathrm{drops}(k)$. First, note that k can be written as

$$k = \sum_{i=1}^{t} \sum_{r_i \leq j \leq \ell_i} 2^j\,, \tag{6}$$

where $t \leq \mathrm{drops}(k)$ and $r_1 \leq \ell_1 < r_2 \leq \ell_2 < \cdots < r_t \leq \ell_t$ are non-negative integers. Second, note that each sum $\sum_{r_i \leq j \leq \ell_i} 2^j$ can be expressed in terms of at most two signed powers of 2.

In order to prove the other inequality, we first note that for every $k, j \geq 0$

$$|\mathrm{drops}(k + 2^j) - \mathrm{drops}(k)| \leq 1\,. \tag{7}$$

To see (7), assume first that $B_j(k) = 0$. Then adding 2^j can introduce one drop if $B_{j+1}(k) = 0$ and delete one drop if $j > 0$ and $B_{j-1}(k) = 1$. If $B_j(k) = 1$, let ℓ be the smallest integer with $B_\ell(k) = 0$ and $\ell > j$. The binary representations of k and $k + 2^j$ are the same, except that $B_\ell(k + 2^j) = 1$ and $B_i(k + 2^j) = 0$ for every $j \leq i < \ell$. The number of drops in $k + 2^j$

can increase only if $B_{j-1}(k) = 1$ and $j > 0$, and increases at most by one. It can decrease only if $B_{\ell+1}(k) = 1$, and again at most by one.

By induction on $\mathrm{sbl}(k)$, we now prove that for every $k \geq 0$, $\mathrm{drops}(k) \leq \mathrm{sbl}(k)$. If $\mathrm{sbl}(k) = 0$, the statement holds. Otherwise, let $k_1, \ldots, k_b$, with $b = \mathrm{sbl}(k)$, be signed powers of two that sum to k. Choose k_i so that $k - k_i$ is non-negative. Hence, (7) shows that $\mathrm{drops}(k - k_i) \geq \mathrm{drops}(k) - 1$. Furthermore, we have $\mathrm{sbl}(k - k_i) = \mathrm{sbl}(k) - 1$. By the inductive assumption, $\mathrm{sbl}(k - k_i) \geq \mathrm{drops}(k - k_i)$. Altogether, $\mathrm{sbl}(k) = \mathrm{sbl}(k - k_i) + 1 \geq (\mathrm{drops}(k) - 1) + 1 = \mathrm{drops}(k)$, completing the proof. ◀

Proof of Theorems 4 and 5. The lower bounds in both theorems follow from Lemmas 7 and 9, since $\mathrm{drops}(\sigma_d) = d/2$ for σ_d as defined in (3). The proof of the upper bounds is by induction and is not presented due to space considerations. ◀

Remarks

First, none of the bounds in Theorem 5 is tight for all k. For example, the $2\mathrm{drops}(k)$ upper bound exceeds the upper bound from Theorem 4 for $k = \sigma_d$ from (3). Furthermore, we do not know which of the bounds in Theorem 4 is more accurate.

Second, we have $\mathrm{eip}_{T_d}(\sigma_d) \leq d/2 - \Omega(\log d)$. Hence, the lower bound of $\mathrm{drops}(k) - O(\log(\mathrm{drops}(k))$ from Theorem 5 cannot be improved as a function of $\mathrm{drops}(k)$. A similar statement holds for Lemma 7 and $\mathrm{sbl}(k)$.

Third, the quantity $\mathrm{st}(k)$ from Claim 8 characterizes eip almost exactly: $\mathrm{st}(k) - 1 \leq \mathrm{eip}_{T_d}(k) \leq \mathrm{st}(k)$. This can be deduced from Claim 8 and Theorem 2.3 from [3]. It also implies $|\mathrm{eip}_{T_d}(k) - \mathrm{sbl}(k)| \leq O(\log k)$.

Fourth, we do not know of a simple procedure to compute $\mathrm{eip}_{T_d}(k)$, or to find a subset of size k that minimizes the boundary-size. The quantity $\mathrm{sbl}(k)$, however, can be computed exactly and quite easily, in time polynomial in $\log k$, which allows to efficiently approximate $\mathrm{eip}_{T_d}(k)$ more accurately.

3 Algebraic complexity

In this section, we prove the separation between monotone circuits and ABPs stated in Theorem 1, and the structure of ABPs stated in Lemma 3.

3.1 The tree function

We first describe a boolean function $\mathcal{T}_{d,m}$ which is later used to prove our separation. Recall that T_d is the full binary tree of depth d, and let V be the set of its vertices. For an integer $m > 1$, let $\mathbb{Z}_m$ be the additive group of integers modulo m, and let $\mathbb{Z}_m^V$ be the set of functions $\gamma : V \to \mathbb{Z}_m$.

A function $\gamma \in \mathbb{Z}_m^V$ will be called *legal*, if for every vertex v which is not a leaf and its two children v_1, v_2, we have

$$\gamma(v) = \gamma(v_1) + \gamma(v_2)\,.$$

The tree function $\mathcal{T}_{d,m} : \mathbb{Z}_m^V \to \{0,1\}$ takes $\gamma \in \mathbb{Z}_m^V$ and accepts if γ is legal.

For a node $v \in V$, let $T(v)$ be the full subtree of T_d rooted at v and let $\ell(T(v))$ be the set of its leaves. Then γ is legal iff

$$\gamma(v) = \sum_{u \in \ell(T(v))} \gamma(u) \tag{8}$$

holds for every vertex v in T.

The following lemma depicts the key property of $\mathcal{T}_{d,m}$ that we later use. For two functions f, g over the same domain, write $f \leq g$ if $f(x) \leq g(x)$ for all x. Denote by $\sup(f)$ the set of inputs x so that $f(x) = 1$.

▶ **Lemma 10.** *Let V_0, V_1 be a partition of V where $|V_0| = \sigma_d$, as defined in (3). If*

$$h_0 : \mathbb{Z}_m^{V_0} \to \{0,1\}, \quad h_1 : \mathbb{Z}_m^{V_1} \to \{0,1\}, \quad h_0 \wedge h_1 \leq \mathcal{T}_{d,m},$$

then $|\sup(h_0 \wedge h_1)| \leq m^{-d/16} |\sup(\mathcal{T}_{d,m})|$.

Proof. View T_d as directed from leaves to root. A directed path $v_1, \ldots, v_k$ in T_d from $v_1 \in \ell(T(v_k))$ to v_k will be called *pure* if $v_1 \neq v_k$ and there exists $i \in \{0,1\}$ so that $\{v_1, \ldots, v_k\} \cap V_i = \{v_k\}$. A node will be called *pure* if it is the last node of some pure path. Let $P \subset V$ be the set of pure nodes. Let E be the edge boundary of V_0.

To prove the lemma, we use the following two claims.

▶ **Claim 11.** $|P| \geq |E|/4$.

Proof of Claim 11. Let S be the set of nodes $v \in V$ so that the parent of v in T_d is pure. Since $|S| \leq 2|P|$, it is enough to show that $|S| \geq |E|/2$. Let T' be the minor of T_d obtained by contracting all the edges of T_d not in E. The tree T' is a (not necessarily binary) tree with $|E|$ edges. For $x \in V(T')$, let $[x] \subseteq V$ be the set of vertices that have been contracted to x. The root of T' is the vertex x so that the root of T_d is in $[x]$. View T' as directed from leaves to the root. For $x \in V(T')$, let $v(x) \in V$ be the vertex in $[x]$ that is closest to the root of T_d. This is well defined, since the set $[x]$ is connected in T_d. For every leaf $x \in V(T')$, we have $v(x) \in S$. For every $x \in V(T')$ with in-degree one which is not the root of T', the vertex $v(x)$ is also in S. Recall that T' can have at most $|E|/2$ vertices of in-degree at least two. Hence, $|S| \geq (|E|+1) - (|E|/2+1)$, as required. ◀

For every pure node v, fix a leaf $\tilde{v}$ such that the path from $\tilde{v}$ to v is pure. Let $\tilde{P} = \{\tilde{v} : v \in P\}$. The sizes of $\tilde{P}$ and P are the same.

▶ **Claim 12.** *Assume $\sup(h_0) \neq \emptyset$. Then every $\beta \in \sup(h_1)$ is uniquely determined by its values on $(\ell(T_d) \cap V_1) \setminus \tilde{P}$.*

Proof of Claim 12. Fix $\alpha \in \sup(h_0)$. For the sake of contradiction, assume that there are two distinct $\beta_1, \beta_2 \in \sup(h_1)$ which agree on $(\ell(T_d) \cap V_1) \setminus \tilde{P}$. Since $h_0 \wedge h_1 \leq \mathcal{T}_{d,m}$, both $\alpha \cup \beta_1$ and $\alpha \cup \beta_2$ are legal maps satisfying (8). Hence, β_1 and β_2 differ on some leaf in V_1. They agree on leaves outside of $\tilde{P}$ and so there exists a pure node $v \in V_0$ such that $\beta_1(\tilde{v}) \neq \beta_2(\tilde{v})$. We can assume that v is minimal in that for every pure node $u \in T(v) \cap V_0$ with $u \neq v$, we have $\beta_1(\tilde{u}) = \beta_2(\tilde{u})$. Since every pure path that starts in $\ell(T(v)) \cap V_1$ must also end in $T(v)$, we obtain that $\beta_1(u) = \beta_2(u)$ for every leaf in $T(v) \cap V_1$ with the sole exception of $\tilde{v}$. But this is impossible since (8) implies that for every $i \in \{1,2\}$,

$$\alpha(v) - \sum_{u \in \ell(T(v)) \cap V_0} \alpha(u) = \beta_i(\tilde{v}) + \sum_{u \in (\ell(T(v)) \cap V_1) \setminus \{\tilde{v}\}} \beta_i(u)\,,$$

which gives $\beta_1(\tilde{v}) = \beta_2(\tilde{v})$. ◀

To conclude the lemma, assume that $\sup(h_0 \wedge h_1) \neq \emptyset$, otherwise we are done. Claim 12 shows that

$$|\sup(h_0)| \leq m^{|(\ell(T_d) \cap V_0) \setminus \tilde{P}|}\,, \quad |\sup(h_1)| \leq m^{|(\ell(T_d) \cap V_1) \setminus \tilde{P}|}\,.$$

Since $|\sup(\mathcal{T}_{d,m})| = m^{|\ell(T_d)|}$,

$$|\sup(h_0)| \cdot |\sup(h_1)| \leq m^{|\ell(T_d)|-|\tilde{P}|} = m^{-|\tilde{P}|}|\sup(\mathcal{T}_{d,m})|\,.$$

Finally, Claim 11 and Corollary 6 give $|\tilde{P}| = |P| \geq |E|/4 \geq e_{T_d}(V_0)/4 \geq d/16$. ◀

3.2 The tree polynomial

We now describe the polynomial that separates monotone circuits from ABPs. Recall that T_d is the full binary tree of depth d and $\mathcal{T}_{d,m} : \mathbb{Z}_m^V \to \{0,1\}$ is the tree function, where $V = V(T_d)$. For every $v \in T_d$ and $z \in \mathbb{Z}_m$, introduce the variable $x_{v,z}$. For a partial function $\gamma : V \to \mathbb{Z}_m$, define the monomial $x_\gamma = \prod_{v \in \mathrm{dom}(\gamma)} x_{v,\gamma(v)}$. Define the tree polynomial as

$$\mathcal{P}_{d,m} = \sum_{\gamma \in \mathbb{Z}_m^V : \mathcal{T}_{d,m}(\gamma)=1} x_\gamma\,.$$

It is homogeneous of degree $|V| = 2^{d+1} - 1$ and it has $m|V|$ variables.

We now prove a generalization of Theorem 1.

▶ **Proposition 13.** *The polynomial $\mathcal{P}_{d,m}$ can be computed by a monotone arithmetic circuit of size $O(m^2 2^d)$. However, every monotone ABP computing $\mathcal{P}_{d,m}$ has size at least $m^{\Omega(d)}$.*

Theorem 1 follows from Proposition 13 by setting $m = 2^d$.

Proof of Proposition 13. We first prove the lower bound. Assume that $\mathcal{P}_{d,m}$ has a monotone ABP of size s. Let σ_d be as defined in (3). By Lemma 3, we can write

$$\mathcal{P}_{d,m} = \sum_{i=1}^{s} h_i g_i\,,$$

where h_i, g_i are homogeneous and monotone, h_i has degree σ_d and g_i has degree $\deg(\mathcal{P}_{d,m}) - \sigma_d$. We can assume $h_i g_i \neq 0$ for every i. For a polynomial f, let $\mathrm{mon}(f)$ be the set of γ's such that x_γ has a non-zero coefficient in f, and let $f^\star$ be the boolean function with $\sup(f^\star) = \mathrm{mon}(f)$. Monotonicity guarantees that for every $i \in [s]$, there exists $V_i \subseteq V$ with $|V_i| = \sigma_d$ such $\mathrm{mon}(h_i) \subseteq \mathbb{Z}_m^{V_i}$ and $\mathrm{mon}(g_i) \subseteq \mathbb{Z}_m^{V \setminus V_i}$. Therefore,

$$\mathcal{P}^\star_{d,m} = \bigvee_{i=1}^{s} h_i^\star \wedge g_i^\star\,.$$

Since $\mathcal{P}^\star_{d,m} = \mathcal{T}_{d,m}$, Lemma 10 gives $s \geq m^{d/16}$.

For the upper bound, fix m and proceed by induction on d. For $a \in \mathbb{Z}_m$, let $F_{a,d}$ be the polynomial defined as $\mathcal{P}_{d,m}$, except γ range over legal maps with $\gamma(v_d) = a$, where v_d is the root. Hence, $\mathcal{T}_{d,m} = \sum_{a \in \mathbb{Z}_m} F_{a,d}$.

We will show that $F_{d,a}$, for all $a \in \mathbb{Z}_m$, can be simultaneously computed by a circuit of size $s(d) = O(m^3 2^d)$. For $d = 0$, we have $s(d) = O(m)$. Assume that $d > 0$. Let T_ℓ be the left subtree of T_d of depth $d-1$, and let T_r be the right subtree of depth $d-1$. Let C_ℓ, C_r be the two circuits guaranteed by induction on T_ℓ, T_r. For $a \in \mathbb{Z}_m$, denote by $f_{\ell,a}$ the polynomial computed in C_ℓ so that the root of T_ℓ is mapped to a, and similarly define $f_{r,a}$. To define the circuit for $F_{d,a}$, use the following:

$$F_{d,a} = \sum_{a_\ell, a_r \in \mathbb{Z}_m : a = a_\ell + a_r} f_{\ell,a_\ell} f_{r,a_r} x_{v_d,a},$$

Overall, we get the recursion

$$s(d) \leq O(m^2) + 2 \cdot S(d-1),$$

showing that $s(d) = O(m^2 2^d)$. ◀

3.3 The structure of ABPs

We end this section by proving Lemma 3.

Proof of Lemma 3. Let P be an ABP computing a homogeneous polynomial f of degree r. For a vertex v in P, denote by h_v the polynomial

$$h_v = \sum_{\gamma : v_{\text{start}} \to v} \prod_{e \in \gamma} L_P(e)$$

and denote by g_v the polynomial

$$g_v = \sum_{\gamma : v \to v_{\text{end}}} \prod_{e \in \gamma} L_P(e).$$

Since the computation is monotone and f homogeneous, both h_v, g_v are homogeneous and the sum of their degrees is r. Denote by U the set of vertices v in P so that the degree of h_v is k. Every directed path from v_{start} to v_{end} in P passes through U exactly once. It follows that

$$f = \sum_{v \in U} h_v g_v,$$

as needed. ◀

Acknowledgements We thank Pavel Pudlák for helpful discussions.

References

1. M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *FOCS*, pages 67–75, 2008.
2. S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.
3. B. S. Bharadwaj, L. S. Chandran, and A. Das. Isoperimetric problem and meta-fibonacci sequences. In *Computing and Combinatorics*, pages 22–30. Springer, 2008.
4. A. Borodin, A. A. Razborov, and R. Smolensky. On lower bounds for read-k-times branching programs. *Computational Complexity*, 3(1):1–18, 1993.
5. P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic complexity theory*, volume 315. Springer Science and Business Media, 1997.
6. H. Fournier, N. Limaye, G. Malod, and S. Srinivasan. Lower bounds for depth 4 formulas computing iterated matrix multiplication. In *STOC*, pages 128–135, 2014.
7. J. von zur Gathen. Algebraic complexity theory. *Annual review of computer science*, 3:317–347, 1988.
8. A. Gupta, P. Kamath, N. Kayal, and R. Saptharishi. Arithmetic circuits: A chasm at depth three. In *FOCS*, pages 578–587, 2013.
9. A. Gupta, P. Kamath, N. Kayal, and R. Saptharishi. Approaching the chasm at depth four. *Journal of the ACM*, 61(6):33, 2014.

10 P. Hrubes and A. Yehudayoff. Monotone separations for constant degree polynomials. *Information Processing Letters*, 110(1):1–3, 2009.

11 L. Hyafil. On the parallel evaluation of multivariate polynomials. *SIAM J. Comput.*, 8(2):120–123, 1979.

12 S. Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science and Business Media, 2012.

13 P. Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theoretical Computer Science*, 448:56–65, 2012.

14 M. Kumar and S. Saraf. The limits of depth reduction for arithmetic formulas: It's all about the top fan-in. In *STOC*, pages 136–145, 2014.

15 E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, New York, NY, USA, 1997.

16 F. Morgan. Manifolds with density. *Notices of the AMS*, pages 853–858, 2005.

17 N. Nisan. Lower bounds for non-commutative computation. In *STOC*, pages 410–418, 1991.

18 C. H. Papadimitriou and M. Sipser. Communication complexity. In *STOC*, pages 196–200, 1982.

19 R. Raz and A. Yehudayoff. Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors. *J. Comput. Syst. Sci.*, 77(1):167–190, 2011.

20 E. Shamir and M. Snir. Lower bounds on the number of multiplications and the number of additions in monotone computations. Technical Report RC-6757, IBM, 1977.

21 A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Found. Trends Theor. Comput. Sci.*, 5:207–388, 2010. URL: `10.1561/0400000039`, `doi:10.1561/0400000039`.

22 S. Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Information and Computation*, 240:2–11, 2015.

23 L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. on Computing*, 12(4):641–644, 1983.

24 A. C. Yao. Some complexity questions related to distributive computing (preliminary report). In *STOC*, pages 209–213, 1979.

Tolerant Testers of Image Properties*†

Piotr Berman[1], Meiram Murzabulatov[2], and Sofya Raskhodnikova[3]

1 **Pennsylvania State University, University Park, USA**
berman@cse.psu.edu
2 **Pennsylvania State University, University Park, USA**
mzm269@psu.edu
3 **Pennsylvania State University, University Park, USA**
sofya@cse.psu.edu

Abstract

We initiate a systematic study of tolerant testers of image properties or, equivalently, algorithms that approximate the distance from a given image to the desired property (that is, the smallest fraction of pixels that need to change in the image to ensure that the image satisfies the desired property). Image processing is a particularly compelling area of applications for sublinear-time algorithms and, specifically, property testing. However, for testing algorithms to reach their full potential in image processing, they have to be tolerant, which allows them to be resilient to noise. Prior to this work, only one tolerant testing algorithm for an image property (image partitioning) has been published.

We design efficient approximation algorithms for the following fundamental questions: What fraction of pixels have to be changed in an image so that it becomes a half-plane? a representation of a convex object? a representation of a connected object? More precisely, our algorithms approximate the distance to three basic properties (being a half-plane, convexity, and connectedness) within a small additive error ϵ, after reading a number of pixels polynomial in $1/\epsilon$ and independent of the size of the image. The running time of the testers for half-plane and convexity is also polynomial in $1/\epsilon$. Tolerant testers for these three properties were not investigated previously. For convexity and connectedness, even the existence of distance approximation algorithms with query complexity independent of the input size is not implied by previous work. (It does not follow from the VC-dimension bounds, since VC dimension of convexity and connectedness, even in two dimensions, depends on the input size. It also does not follow from the existence of non-tolerant testers.)

Our algorithms require very simple access to the input: uniform random samples for the half-plane property and convexity, and samples from uniformly random blocks for connectedness. However, the analysis of the algorithms, especially for convexity, requires many geometric and combinatorial insights. For example, in the analysis of the algorithm for convexity, we define a set of reference polygons P_ϵ such that (1) every convex image has a nearby polygon in P_ϵ and (2) one can use dynamic programming to quickly compute the smallest empirical distance to a polygon in P_ϵ. This construction might be of independent interest.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Computational geometry, convexity, half-plane, connectedness, property testing, tolerant property testing

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.90

* A full version of the paper is available at https://arxiv.org/abs/1503.01363.

† M.M. and S.R. were supported by NSF CAREER award CCF-0845701 and NSF award CCF-1422975. S.R. was also supported by Boston University's Hariri Institute for Computing and Center for Reliable Information Systems and Cyber Security and, while visiting the Harvard Center for Research on Computation & Society, by a Simons Investigator grant to Salil Vadhan.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 90; pp. 90:1–90:14

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

1 Introduction

Image processing is a particularly compelling area of applications for sublinear-time algorithms and, specifically, property testing. Images are huge objects, and our visual system manages to process them very quickly without examining every part of the image. Moreover, many applications in image analysis have to process a large number of images online, looking for an image that satisfies a certain property among images that are generally very far from satisfying it. Or, alternatively, they look for a subimage satisfying a certain property in a large image (e.g., a face in an image where most regions are part of the background.) There is a growing number of proposed *rejection-based* algorithms that employ a quick test that is likely to reject a large number of unsuitable images (see, e.g., citations in [15]).

Property testing [21, 10] is a formal study of fast algorithms that accept objects with a given property and reject objects that are far. Testing image properties in this framework was first considered in [19]. Ron and Tsur [20] initiated property testing of images with a different input representation, suitable for testing properties of sparse images. Since these models were proposed, several sublinear-time algorithms for visual properties were implemented and used: namely, those by Kleiner et al. and Korman et al. [15, 16, 17].

However, for sublinear-time algorithms to reach their full potential in image processing, they have to be resilient to noise: images are often noisy, and it is undesirable to reject images that differ only on a small fraction of pixels from an image satisfying the desired property. Tolerant testing was introduced by Parnas, Ron and Rubinfeld [18] exactly with this goal in mind—to deal with noisy objects. It builds on the property testing model and calls for algorithms that accept objects that are close to having a desired property and reject objects that are far. Another related task is approximating distance of a given object to a nearest object with the property within additive error ϵ. (Distance approximation algorithms imply tolerant testers in a straightforward way.) The only image problem for which tolerant testers were studied is the image partitioning problem investigated by Kleiner et al. [15].

Our results. We design efficient approximation algorithms for the following fundamental questions: What fraction of pixels have to be changed in an image so that it becomes a half-plane? a representation of a convex object? a representation of a connected object? In other words, we design algorithms that approximate the distance to being a half-plane, convexity and connectedness within a small additive error or, equivalently, tolerant testers for these properties. These problems were not investigated previously in the tolerant testing framework. For all three properties, we give ϵ-additive distance approximation algorithms that run in constant time (i.e., dependent only on ϵ, but not the image size). We remark that even though it was known that these properties can be tested in constant time [19], this fact does not necessarily imply constant-query tolerant testers for these properties. E.g., Fischer and Fortnow [9] exhibit a property (of objects representable with strings of length n) which is testable with a constant number of queries, but for which every tolerant tester requires $n^{\Omega(1)}$ queries. For convexity and connectedness, even the existence of distance approximation algorithms with query (or time) complexity independent of the input size does not follow from previous work. It does not follow from the VC-dimension bounds, since VC dimension of convexity and connectedness, even in two dimensions, depends on the input size[1]. Implications of the VC dimension bound on convexity are further discussed below.

[1] For $n \times n$ images, the VC dimension of convexity is $\Theta(n^{2/3})$ (this is the maximum number of vertices of a convex lattice polygon in an $n \times n$ lattice [1]); for connectedness, it is $\Theta(n)$.

Table 1 Our results on distance approximation. To get complexity of (ϵ_1, ϵ_2)-tolerant testing, substitute $\epsilon = (\epsilon_2 - \epsilon_1)/2$.

Property	Sample Complexity	Run Time	Access to Input
Half-plane	$O\left(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}\right)$	$O\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)$	uniformly random pixels
Convexity	$O\left(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}\right)$	$O\left(\frac{1}{\epsilon^8}\right)$	uniformly random pixels
Connectedness	$O\left(\frac{1}{\epsilon^4}\right)$	$\exp\left(O\left(\frac{1}{\epsilon}\right)\right)$	uniformly random blocks of pixels

Our results on distance approximation are summarized in Table 1. Our algorithm for convexity is the most important and technically difficult of our results, requiring a large number of new ideas to get running time polynomial in $1/\epsilon$. To achieve this, we define a set of reference polygons P_ϵ such that (1) every convex image has a nearby polygon in P_ϵ and (2) one can use dynamic programming to quickly compute the smallest empirical distance to a polygon in P_ϵ. It turns out that the empirical error of our algorithm is proportional to the sum of the square roots of the areas of the regions it considers in the dynamic program. To guarantee (2) and keep our empirical error small, our construction ensures that the sum of the square roots of the areas of the considered regions is small. This construction might be of independent interest.

Our algorithms do not need sophisticated access to the input image: uniformly randomly sampled pixels suffice for our algorithms for the half-plane property and convexity. For connectedness, we allow our algorithms to query pixels from a uniformly random block. (See the end of Section 2 for a formal specification of the input access.)

Our algorithms for convexity and half-plane work by first implicitly learning the object[2]. PAC learning was defined by Valiant [23], and agnostic learning, by Kearns et al. [14] and Haussler [12]. As a corollary of our analysis, we obtain fast proper agnostic PAC learners of half-planes and of convex sets in two dimensions that work under the uniform distribution. The sample and time complexity[3] of the PAC learners is as indicated in Table 1 for distance approximation algorithms for corresponding properties.

While the sample complexity of our agnostic half-plane learner (and hence our distance approximation algorithm for half-planes) follows from the VC dimension bounds, its running time does not. Agnostically learning half-spaces under the uniform distribution has been studied by [13], but only for the hypercube $\{-1, 1\}^d$ domains, not the plane. Our PAC learner of convex sets, in contrast to our half-plane learner, dimension lower bounds on sample complexity. (The sample complexity of a PAC learner for a class is at least proportional to the VC dimension of that class [8].) Since VC dimension of convexity of $n \times n$ images is $\Theta(n^{2/3})$, proper PAC learners of convex sets in two dimensions (that work under arbitrary

[2] There is a known implication from learning to testing. As proved in [10], a proper PAC learning algorithm for property $\mathcal{P}$ with sampling complexity $q(\epsilon)$ implies a 2-sided error (uniform) property tester for $\mathcal{P}$ that takes $q(\epsilon/2) + O(1/\epsilon)$ samples. There is an analogous implication from proper agnostic PAC learning to distance approximation with an overhead of $O(1/\epsilon^2)$ instead of $O(1/\epsilon)$. We choose to present our testers first and get learners as corollary because our focus is on testing and because we want additional features for our testers, such as 1-sided error, that do not automatically follow from the generic relationship.

[3] All our results are stated for error probability $\delta = 1/3$. To get results for general δ, by standard arguments, it is enough to multiply the complexity of an algorithm by $\log 1/\delta$.

distributions) must have sample complexity $\Omega(n^{2/3})$. However, one can do much better with respect to the uniform distribution. Schmeltz [22] showed that a non-agnostic learner for that task needs $\Theta(\epsilon^{-3/2})$ samples. Surprisingly, it appears that this question has not been studied at all for agnostic learners. Our agnostic learner for convex sets in 2D under the uniform distribution needs $O\left(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon}\right)$ samples and runs in time $O\left(\frac{1}{\epsilon^8}\right)$.

Finally, we note that for connectedness, we take a different approach. Our algorithms do not try to learn the object first; instead they rely on a combinatorial characterization of distance to connectedness. We show that distance to connectedness can be represented as an average of distances of sub-images to a related property.

Comparison to other related work. Property testing has rich literature on graphs and functions, however, properties of images have been investigated very little. Even though superficially the inputs to various types of testing tasks might look similar, the problems that arise are different. In the line of work on testing dense graphs, started by Goldreich et al. [10], the input is also an $n \times n$ binary matrix, but it represents an adjacency matrix of the dense input graph. So, the problems considered are different than in this work. In the line of work on testing geometric properties, started by Czumaj, Sohler, and Ziegler [7] and Czumaj and Sohler [6], the input is a set of points represented by their coordinates. The allowed queries and the distance measure on the input space are different from ours.

A line of work potentially relevant for understanding connectedness of images is on connectedness of bounded-degree graphs. Goldreich and Ron [11] gave a tester for this property, subsequently improved by Berman et al. [3]. Campagna et al. [5] gave a tolerant tester for this problem. Even though we view our image as a graph in order to define connectedness of images, there is a significant difference in how distances between instances are measured (see [19] for details). We also note, that unlike in [5], our tolerant tester for connectedness is fully tolerant, i.e., it works for all settings of parameters.

The only previously known tolerant tester for image properties was given by Kleiner et al. [15]. They consider the following class of image partitioning problems, each specified by a $k \times k$ binary template matrix T for a small *constant* k. The image satisfies the property corresponding to T if it can be partitioned by $k-1$ horizontal and $k-1$ vertical lines into blocks, where each block has the same color as the corresponding entry of T. Kleiner et al. prove that $O(1/\epsilon^2)$ samples suffice for tolerant testing of image partitioning properties. Note that VC dimension of such a property is $O(1)$, so by Footnote 2, we can get a $O(1/\epsilon^2 \log 1/\epsilon)$ bound. Our algorithms required numerous new ideas to significantly beat VC dimension bounds (for convexity and connectedness) and to get low running time.

For the properties we study, distance approximation algorithms and tolerant testers were not investigated previously. In the standard property testing model, the half-plane property can be tested in $O(\epsilon^{-1})$ time [19], convexity can be tested in $O(\epsilon^{-4/3})$ time [2], and connectedness can be tested in $O(\epsilon^{-2}\log\epsilon^{-1})$ time [19, 3]. As we explained, property testers with running time independent of ϵ do not necessarily imply tolerant testers with that feature. Many new ideas are needed to obtain our tolerant testers. In particular, the standard testers for half-plane and connectedness are adaptive while the testers here need only random samples from the image, so the techniques used for analyzing them are different. The tester for convexity in [2] uses only random samples, but it is not based on dynamic programming.

Open questions. In this paper we give tolerant testers for several important problems on images. It is open whether these testers are optimal. No nontrivial lower bounds are

known for these problems. (For any non-trivial property, an easy lower bound on the query complexity of a distance approximation algorithm is $\Omega(1/\epsilon^2)$. This follows from the fact that $\Omega(1/\epsilon^2)$ coin flips are needed to distinguish between a fair coin and a coin that lands heads with probability $1/2+\epsilon$.) Thus, our testers for half-plane and convexity are nearly optimal in terms of query complexity (up to a logorithmic factor in $1/\epsilon$). But it is open whether their running time can be improved.

Organization. We give formal definitions and notation in Section 2, deferring some standard definitions to the full version of this article. Algorithms for being a half-plane, convexity, and connectedness are given in Sections 3, 4, and 5, respectively. We view our half-plane result as a good preparation for our distance approximation algorithm for convexity, the most technically difficult result in the paper. Corollaries about PAC learners as well as all omitted proofs and numerous figures can be found in the full version of this article.

2 Definitions and Notation

We use $[0..n)$ to denote the set of integers $\{0,1,\ldots,n-1\}$ and $[n]$ to denote $\{1,2,\ldots,n\}$.

Image representation. We focus on black and white images. For simplicity, we only consider square images, but everything in this paper can be easily generalized to rectangular images. We represent an image by an $n\times n$ binary matrix M of pixel values, where 0 denotes white and 1 denotes black. We index the matrix by $[0..n)^2$. The object is a subset of $[0..n)^2$ corresponding to black pixels; namely, $\{(i,j)\mid M[i,j]=1\}$.

The *absolute distance*, $Dist(M_1,M_2)$, between matrices M_1 and M_2 is the number of the entries on which they differ. The *relative distance* between them is $dist(M_1,M_2)=Dist(M_1,M_2)/n^2$. A property $\mathcal{P}$ is a subset of binary matrices.

Access to the input. A *query-based* algorithm accesses its $n\times n$ input matrix M by specifying a query pixel (i,j) and obtaining $M[i,j]$. A *uniform* algorithm accesses its $n\times n$ input matrix by drawing independent samples (i,j) from the uniform distribution over the domain (i.e., $[0..n)^2$) and obtaining $M[i,j]$. A *block-uniform algorithm* accesses its $n\times n$ input matrix by specifying a block length $r\in[n]$. For a block length r of its choice, the algorithm draws $x,y\in[\lceil n/r\rceil]$ uniformly at random and obtains set $\{(i,j)\mid \lfloor i/r\rfloor=x \text{ and } \lfloor j/r\rfloor=y\}$ and $M[i,j]$ for all (i,j) in this set. The sample complexity of a *uniform* or a *block-uniform* algorithm is the number of pixels of the image it examines.

▶ Remark 2.1. Uniform algorithms have access to independent (labeled) samples from the uniform distribution over the domain. *Bernoulli algorithms* only have access to (labeled) Bernoulli samples from the image: namely, each pixel appears in the sample with probability s/n^2, where s is the sample parameter that controls the expected sample complexity. By standard arguments, a Bernoulli algorithm with the sample parameter s can be used to obtain a uniform algorithm that takes $O(s)$ samples and has the same guarantees as the original algorithm (and vice versa).

3 Distance Approximation to the Nearest Half-Plane

An image is a *half-plane* if there exist an angle $\varphi\in[0,2\pi)$ and a real number c such that $M[x,y]=1$ (i.e., pixel (x,y) is black) iff $x\cos\varphi+y\sin\varphi\geq c$. In other words, an image is a half-plane if there is a line, called a *separating line*, that separates black and white pixels of

the image. For all φ and c, let M_c^φ denote the half-plane that satisfies the above inequality with parameters φ and c, and let L_c^φ be the segment of the separating line that belongs to the image. We call φ the *direction* of M_c^φ (and L_c^φ). Note that φ is the oriented angle between the x-axis and a line perpendicular to L_c^φ.

▶ **Theorem 3.1.** *There is a uniform ϵ-additive distance approximation algorithm for the half-plane property with sample complexity $O(\frac{1}{\epsilon^2}\log\frac{1}{\epsilon})$ and time complexity $O(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon})$.*

Proof. At a high level, our algorithm (Algorithm 1) constructs a small set H_ϵ of reference half-planes. It samples pixels uniformly at random and outputs the empirical distance to the closest reference half-plane. The core property of H_ϵ is that the smallest empirical distance to a half-plane in H_ϵ can be computed quickly.

▶ **Definition 3.2** (Reference directions and half-planes)**.** Given $\epsilon \in (0, \frac{1}{4})$, let $a = \epsilon n/\sqrt{2}$. Let D_ϵ be the set of directions of the form $i\epsilon$ for $i \in [0..\lceil 2\pi/\epsilon\rceil)$, called *reference directions*. The set of *reference half-planes*, denoted H_ϵ, consists of half-planes of the form M_c^φ, where $\varphi \in D_\epsilon$, the reference line intersects $[0, n-1]^2$, and c is an integer multiple of a.

In other words, for every reference direction, we space reference half-planes distance a apart. By definition, there are at most $\sqrt{2}n/a = 2/\epsilon$ reference half-planes for each direction in D_ϵ and, consequently, $|H_\epsilon| \le 2\pi/\epsilon \cdot (2/\epsilon) < 13/\epsilon^2$.

Algorithm 1: Distance approximation to being a half-plane.

Input : parameters $n \in \mathbb{N}$, $\epsilon \in (0, 1/4)$; Bernoulli access to an $n \times n$ binary matrix M.

1 Sample a set S of $s = \frac{4}{\epsilon^2}\ln\frac{9}{\epsilon}$ pixels uniformly at random with replacement.
2 Let D_ϵ and H_ϵ be the sets of reference directions and half-planes, respectively (see Definition 3.2) and let $a = \epsilon n/\sqrt{2}$.
`//` Compute $\hat{d} = \min_{M' \in H_\epsilon} \hat{d}(M')$, where $\hat{d}(M') = \frac{1}{s}\cdot|\{p \in S : M[p] \neq M'[p]\}|$:
3 **foreach** $\varphi \in D_\epsilon$ **do**
`//` Lines with direction φ partition the image. Bucket sort samples by position in the partition:
4 Assign each sample $(x, y) \in S$ to bucket $j = \lfloor (x\cos\varphi + y\sin\varphi)/a \rfloor$.
5 For each bucket j, compute w_j and b_j, the number of white and black pixels it has.
6 For each j, where $M_{ja}^\varphi \in H_\epsilon$, compute $\hat{d}(M_{ja}^\varphi) = \frac{1}{s}\sum_{k<j} b_k + \frac{1}{s}\sum_{k\ge j} w_k$.
7 Output $\hat{d}$, the minimum of the values computed in Step 6.

▶ **Lemma 3.3.** *For every half-plane matrix M, there is $M' \in H_\epsilon$ such that $dist(M, M') \le \epsilon/2$.*

Proof. Consider a half-plane M_c^φ. Let φ' be a reference direction closest to φ. Then $|\varphi - \varphi'| \le \epsilon/2$. We consider two cases. See Figures 1 and 2.

Case 1: Suppose that there is a reference half-plane $M_{c'}^{\varphi'}$ such that the separating line segments L_c^φ and $L_{c'}^{\varphi'}$ intersect. Note that the length of every line segment that belongs to the image is at most $\sqrt{2}n$. The symmetric difference of M_c^φ and $M_{c'}^{\varphi'}$ is contained in two regions formed by line segments L_c^φ and $L_{c'}^{\varphi'}$. Each of these regions is either a triangle or (if it contains a corner of the image) a quadrilateral. First, suppose both regions are triangles. The sum of lengths of their bases, that lie on the same line, is at most $\sqrt{2}n$, whereas the

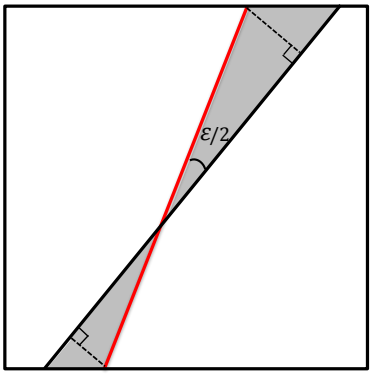


Figure 1 Proof of Lemma 3.3: triangular regions.

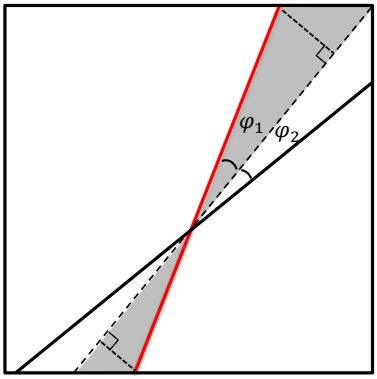


Figure 2 Proof of Lemma 3.3: triangular and quadrilateral regions.

sum of their heights is at most $\sin(\epsilon/2) \times \sqrt{2}n \leq \epsilon n/\sqrt{2}$. Hence, the sum of their areas[4] is at most $\epsilon n^2/2$.

If exactly one of the regions is a quadrilateral, we add a line through the corner of the image contained in the quadrilateral and the intersection point of L_c^{φ} and $L_{c'}^{\varphi'}$. It partitions the symmetric difference of M_c^{φ} and $M_{c'}^{\varphi'}$ into two pairs of triangular regions. Let φ_1 (respectively, φ_2) be the angle between the new line and L_c^{φ} (respectively, $L_{c'}^{\varphi'}$). Then $\varphi_1 + \varphi_2 \leq \epsilon/2$. Applying the same reasoning as before to each pair of regions, we get that the sum of their areas is at most $\varphi_1 n^2 + \varphi_2 n^2 \leq \epsilon n^2/2$. If both regions are quadrilaterals, we add a line as before for each of them and apply the same reasoning as before to the three resulting pairs of regions. Again, the area of the symmetric difference of M_c^{φ} and $M_{c'}^{\varphi'}$ is at most $\epsilon n^2/2$. Thus[4], $M_{c'}^{\varphi'}$ is the required M'.

Case 2: There exist reference half-planes with separating line segments $L = L_{c'}^{\varphi'}$ and $L' = L_{c'+a}^{\varphi'}$ such that the line segment L_c^{φ} is between L and L'. The region between L and L' has length at most $\sqrt{2}n$ and width a. Thus, its area is at most ϵn^2. Partition it into two regions: between L and L_c^{φ} and between L' and L_c^{φ}. One of the two regions has area at most $\epsilon n^2/2$. Thus, $M_{c'}^{\varphi'}$ or $M_{c'+a}^{\varphi'}$ is the required M'. ◀

Analysis of Algorithm 1. Let d_M be the distance of M to being a half-plane. Then there exists a half-plane matrix M^* such that $dist(M, M^*) = d_M$. By a uniform convergence bound (see, e.g., [4]), since $s \geq (2/\epsilon^2)(\ln|H_\epsilon| + \ln 6)$ for all $\epsilon \in (0, 1/4)$, we get that with probability at least 2/3, $|dist(M, M') - \hat{d}(M')| \leq \epsilon/2$ for all $M' \in H_\epsilon$. Suppose this event happened. Then $\hat{d} \geq d_M - \epsilon/2$ because $dist(M, M') \geq d_M$ for all half-planes M'. Moreover, by Lemma 3.3, there is a matrix $\hat{M} \in H_\epsilon$ such that $dist(M, \hat{M}) \leq dist(M, M^*) + dist(M^*, \hat{M}) \leq d_M + \epsilon/2$. For this matrix, $\hat{d}(\hat{M}) \leq dist(M, \hat{M}) + \epsilon/2 \leq d_M + \epsilon$. Thus, $d_M - \epsilon/2 \leq \hat{d} \leq d_M + \epsilon$. That is, $|d_M - \hat{d}| \leq \epsilon$ with probability 2/3, as required.

Sample and time complexity. The number of samples, s, is $O(1/\epsilon^2 \log 1/\epsilon)$. To analyze the running time, recall that $|D_\epsilon| = O(1/\epsilon)$. For each direction in D_ϵ, we bucket sort

[4] For simplicity of presentation, we equate the area of a convex region and the number of pixels in it, thus ignoring additional small-order terms. By Pick's theorem, the number of pixels could be at most the area plus $2n$. It does not affect the asymptotic analysis of our algorithms.

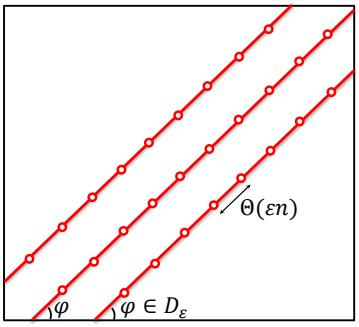


Figure 3 An illustration of reference lines and reference points.

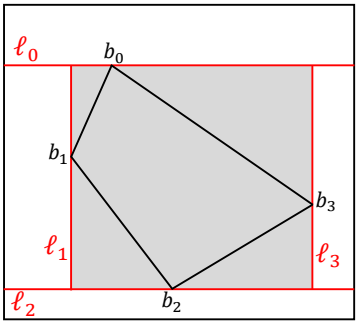


Figure 4 An illustration of a reference box and triangles of $\mathbf{T}_0$.

all samples in expected $O(s)$ time. The remaining steps in the **foreach** loop of Step 3 can be implemented to run in $O(s)$ time. The expected run time of Algorithm 1 is thus $O(1/\epsilon \cdot s) = O(1/\epsilon^3 \log 1/\epsilon)$. Remark 2.1 implies a tester with the same worst case run time. ◀

4 Distance Approximation to the Nearest Convex Image

An image is *convex* if the convex hull of all black pixels contains only black pixels.

▶ **Theorem 4.1.** *There is a uniform ϵ-additive distance approximation algorithm for convexity with sample complexity $O(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$ and running time $O(\frac{1}{\epsilon^8})$.*

Proof. The starting point for our algorithm for approximating the distance to convexity (Algorithm 2) is similar to that of Algorithm 1 that approximates the distance to a nearest half-plane. We define a small set P_ϵ of reference polygons. Algorithm 2 implicitly learns a nearby reference polygon and outputs the empirical distance from the image to that polygon. The key features of P_ϵ is that (1) every convex image has a nearby polygon in P_ϵ, and (2) one can use dynamic programming (DP) to quickly compute the smallest empirical distance to a polygon in P_ϵ.

We start by defining reference directions, lines, points, and line-point pairs that are later used to specify our DP instances. Reference directions are almost the same as in Definition 3.2.

▶ **Definition 4.2** (Reference lines, line-point pairs)**.** Fix $\epsilon_0 = \epsilon/144$. The set of *reference directions* is $D_\epsilon = \{\pi/2\} \cup \{i\epsilon_0 : i \in [0, \lceil 2\pi/\epsilon_0 \rceil)\}$. For every $\varphi \in D_\epsilon$, define the set of *reference lines* $L_\varphi = \{\ell : \ell$ passes through the image and satisfies the equation $x \cos\varphi + y \sin\varphi = c$, where c is an integer multiple of $\epsilon_0 n\}$. For each reference line, the set of *reference points on* ℓ contains points w.r.t. ℓ, which are inside $[0, n-1]^2$, spaced exactly $\epsilon_0 n$ apart (it does not matter how the initial point is picked). A *line-point pair* is a pair (ℓ, b), where ℓ is a reference line and b is a reference point w.r.t. ℓ. (Note that there could be reference points on ℓ that were defined w.r.t. some other reference line. This is why we say "a reference point w.r.t. ℓ", and not "a reference point on ℓ".)

Roughly speaking, a reference polygon is a polygon whose vertices are defined by line-point pairs. There are additional restrictions that stem from the fact that we need to be able to efficiently find a nearby reference polygon for an input image. The actual definition specifies

which actions we can take while constructing a reference polygon. Reference polygons are built starting from reference boxes, which are defined next.

▶ **Definition 4.3** (Reference box)**.** A *reference box* is a set of four line-point pairs (ℓ_i, b_i) for $i = 0, 1, 2, 3$, where ℓ_0, ℓ_2 are distinct horizontal lines, such that ℓ_0 is above ℓ_2, and (ℓ_1, ℓ_3) are distinct vertical lines, such that ℓ_1 is to the left of ℓ_3. The reference box defines a vertex set $B_0 = \{b_0, b_1, b_2, b_3\}$ and a triangle set $\mathbf{T}_0$, formed by removing the quadrilateral $b_0b_1b_2b_3$ from the rectangle delineated by the lines $\ell_0, \ell_1, \ell_2, \ell_3$.

Intuitively, by picking a reference box, we decide to keep the area inside the quadrilateral $b_0b_1b_2b_3$ black, the area outside the rectangle formed by $\ell_0, \ell_1, \ell_2, \ell_3$ white, and the triangles in $\mathbf{T}_0$ gray, i.e., undecided for now.

▶ **Definition 4.4.** For points x, y, let $\ell(x, y)$ denote the line that passes through x and y. Let xy denote the line segment between x and y.

Reference polygons are defined next. Intuitively, to obtain a reference polygon, we keep subdividing "gray" triangles in $\mathbf{T}_0$ into smaller triangles and deciding to color the smaller triangles black or white or keep them gray (i.e., undecided for now). We also allow "cutting off" a quadrilateral that is adjacent to black and coloring it black (a.k.a. "the base change operation"). Even though the definition of reference polygons is somewhat technical, the readers can check their understanding of this concept by following Algorithm 2, as it chooses the best reference polygon to approximate the input image.

▶ **Definition 4.5** (Reference polygon)**.** A *reference polygon* is an image of a polygon Hull(B), where the set B can be obtained from a reference box with a vertex set B_0 and a triangle set $\mathbf{T}_0$ by the following recursive process. Initially, $\mathbf{T}_{\text{end}} = \emptyset$ and $B = B_0$. While $\mathbf{T}_0 \neq \emptyset$, move a triangle T from $\mathbf{T}_0$ to $\mathbf{T}_{\text{end}}$ and perform the following steps:

1. (Base Change). Let $T = \triangle b'b''v$, where $b', b'' \in B$. Select reference point b'_0 on $b'v$ w.r.t. line $\ell(b', v)$, and reference point b''_0 on $b''v$ w.r.t. line $\ell(b'', v)$. Add b'_0, b''_0 to B. (This corresponds to coloring the quadrilateral $b'b'_0b''_0b''$ black.) Let h be the height of $\triangle b'_0b''_0v$ w.r.t. the base $b'_0b''_0$.
2. (Subdivision Step) If $h > 6\epsilon_0 n$, choose whether to proceed with this step or go to Step 3 (both choices correspond to a legal reference polygon); otherwise, go to Step 3. Let φ be the angle between $\ell(b'_0, b''_0)$ and the x-axis, and $\hat{\varphi} \in D_\epsilon$ be such that $|\hat{\varphi} - \varphi| \leq \epsilon_0/2$. Select a reference line-point pair (ℓ, b), where the line $\ell \in L_{\hat{\varphi}}$ crosses b'_0v and b''_0v, whereas b is in the triangle $\triangle b'_0b''_0v$. Let v' (resp., v'') be the point of intersection of ℓ and b'_0v (resp., ℓ and b''_0v). Let $T' = \triangle b'_0bv'$, $T'' = \triangle b''_0bv''$. Add b to B and triangles T', T'' to $\mathbf{T}_0$. (This represents coloring $\triangle b'_0b''_0b$ black and keeping T' and T'' gray.)
3. (End of Processing) Do nothing. (This represents coloring $\triangle b'_0b''_0v$ white).

By Remark 2.1, to prove Theorem 4.1, it suffices to design a Bernoulli tester that takes $s = O(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$ samples in expectation and runs in time $O(\frac{1}{\epsilon^8})$. Our Bernoulli tester is Algorithm 2. In Algorithm 2, we use the following notation for the (relative) empirical error with respect to an input image M, a set of sampled pixels S, and the size parameter s. For an image M', let $\hat{d}(M') = \frac{1}{s} \cdot |\{u \in S : M[u] \neq M'[u]\}|$. For every region $R \subseteq [0..n)^2$, we let $\hat{d}_+(R) = \frac{1}{s} \cdot |\{u \in S \cap R : M[u] = 0\}|$, and $\hat{d}_-(R) = \frac{1}{s} \cdot |\{u \in S \cap R : M[u] = 1\}|$, i.e., the empirical error if we make R black/white, respectively.

Subroutine Best chooses the option with the smallest empirical relative error among those given in Definition 4.5, items 1-3. Its pseudocode is in the full version of this article.

Our set of reference polygons has two critical features. First, for each convex image there is a nearby reference polygon. It turns out that the empirical error for a region is proportional

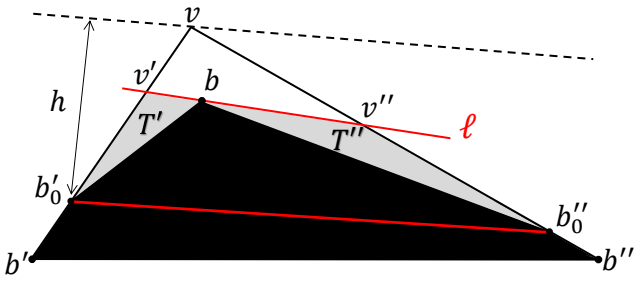


Figure 5 An illustration to Definition 4.5: Triangle $\triangle b'b''v$.

Algorithm 2: Bernoulli approximation algorithm for distance to convexity.

Input : parameters $n \in \mathbb{N}$, $\epsilon \in (0, 1/4)$; Bernoulli access to an $n \times n$ binary matrix M.

1 Set $s = \Theta(\frac{1}{\epsilon^2} \log \frac{1}{\epsilon})$. Include each image pixel in the sample S w.p. $p = s/n^2$.

`// Run the algorithm to find` $\hat{d}$`, the smallest fraction of samples misclassified by a reference polygon in` P_ϵ`. A dynamic programming implementation of the algorithm is given in the full version.`

2 Let W_{ℓ_0} (resp., W_{ℓ_2}) be the set of pixels of the image M that lie either above ℓ_0 or to the left of b_0 on ℓ_0 (resp., either below ℓ_2 or to the left of b_2 on ℓ_2). Let W_{ℓ_1} (resp., W_{ℓ_3}) be the set of pixels of $M - W_{\ell_0} - W_{\ell_2}$ to the left of ℓ_1 (resp., to the right of ℓ_3).

3 Set $\hat{d} = 1$.

4 **forall** *line-point pairs* $(\ell_0, b_0), (\ell_2, b_2)$*, where* ℓ_0, ℓ_2 *are horizontal lines* **do**

5 Set $\hat{d}_{\text{left}} = 1$.

`// The best error for the region to the left of` b_0b_2`, between` ℓ_0 `and` ℓ_2`.`

6 **foreach** *line-point pair* (ℓ_1, b_1)*, where* ℓ_1 *is a vertical line* **do**

7 Let v_0 (resp., v_2) be the point where ℓ_1 intersects ℓ_0 (resp., ℓ_1 intersects ℓ_2).

8 $\hat{d}_{\text{left}} = \min(\hat{d}_{\text{left}}, \hat{d}_-(W_{\ell_1}) + \hat{d}_+(\triangle b_0b_1b_2) + \mathsf{Best}(\triangle b_0b_1v_0) + \mathsf{Best}(\triangle b_1b_2v_2))$

9 Similarly to Steps 5–8, compute $\hat{d}_{\text{right}}$.

`// The best error for the region to the right of` b_0b_2`, between` ℓ_0 `and` ℓ_2`.`

10 Compute $\hat{d} = \min(\hat{d}, \hat{d}_-(W_{\ell_0} \cup W_{\ell_2}) + \hat{d}_{\text{left}} + \hat{d}_{\text{right}})$.

11 **return** $\hat{d}$.

to the square root of its area. The second key feature of our reference polygons is that, for each of them, the set of considered triangles, $\mathbf{T}_{\text{end}}$, has small $\sum_{T \in \mathbf{T}_{\text{end}}} \sqrt{A(T)}$, where $A(T)$ denotes the area of triangle T. The proofs of both features, as well as the analysis of the empirical error, are quite technical and appear in the full version of this article. ◀

Here, we state and partially prove a lemma that puts together different parts of the analysis. It makes it clear why the empirical error of each region is proportional to the square root of its area which is, as explained in Footnote 4, a proxy for the number of pixels in it.

▶ **Lemma 4.6.** *With probability at least* $2/3$ *over the choice of the samples taken by Algorithm 2,* $|\hat{d}(M') - dist(M, M')| \leq 5\epsilon/6$ *for all reference polygons* M'*.*

Proof. Consider a region $R = (R_+, R_-)$, partitioned into two regions R_+ and R_-, such that in some step, the algorithm checks the assumption that R_+ is black and R_- is white, i.e., evaluates $\hat{d}_+(R_+) + \hat{d}_-(R_-)$. Let $\mathbf{R}$ be the set of all such regions R. We show that with probability at least $2/3$, the estimates $\hat{d}_+(R_+) + \hat{d}_-(R_-)$ are accurate on all regions in $\mathbf{R}$.

Fix $R = (R_+, R_-) \in \mathbf{R}$. Let Γ be the set of misclassified pixels in R, i.e., pixels in R_+ which are white in M and pixels in R_- which are black in M. Define $\gamma = |\Gamma|/n^2$. Algorithm 2 approximates γ by $\hat{d}_+(R_+) + \hat{d}_-(R_-) = \frac{1}{s}|\Gamma \cap S|$. Equivalently, it uses the estimate $\frac{1}{p}|\Gamma \cap S|$ for $|\Gamma|$ (recall that $p = s/n^2$). The error of the estimate is $err_S(R) = \frac{1}{p}|\Gamma \cap S| - |\Gamma|$.

▶ **Claim 4.7.** $\Pr[|err_S(R)| > \sqrt{\gamma} \cdot c\epsilon n^2] \leq 2\exp(-\frac{3}{8}c^2\epsilon^2 s)$, *where* $c = 1/21$.

Proof. For each pixel u, define random variables χ_u and X_u, where χ_u is the indicator for the event $u \in S$ (i.e., a Bernoulli variable with the probability parameter p), whereas $X_u = \frac{\chi_u}{p} - 1$. Then our estimate of $|\Gamma|$ is $\frac{1}{p}|\Gamma \cap S| = \frac{1}{p}\sum_{u\in\Gamma}\chi_u$, whereas $err_S(R) = \sum_{u\in\Gamma} X_u$. We use Bernstein inequality to bound $\Pr[\sum_{u\in\Gamma} X_u > \sqrt{\gamma} \cdot c\epsilon n^2]$. The variables X_u are identically distributed. The maximum value of $|X_u|$ is $a = \frac{1-p}{p}$. Note that $\mathbb{E}[X_u^2] = \frac{1}{p^2}\mathbb{E}[(\chi_u - p)^2] = \frac{1}{p^2}\operatorname{Var}[\chi_u] = \frac{1-p}{p} = a$. Assume w.l.o.g. that $z < |\Gamma|$. (If $z \geq |\Gamma|$ then $\sum_{u\in\Gamma} X_u$ cannot exceed z, and the probability we are bounding is 0.) By Bernstein inequality,

$$\begin{aligned}\Pr\left[\sum_{u\in\Gamma} X_u > z\right] &\leq \exp\left(\frac{-z^2/2}{a|\Gamma| + a\cdot z/3}\right) < \exp\left(-\frac{3}{8}\cdot\frac{z^2\cdot p}{|\Gamma|}\right) = \exp\left(-\frac{3}{8}\frac{\gamma c^2\epsilon^2 n^4}{\gamma n^2}\frac{s}{n^2}\right)\\ &= \exp(-\frac{3}{8}c^2\epsilon^2 s).\end{aligned}$$

The second inequality holds because $a < 1/p$ and $z < |\Gamma|$. The equalities are obtained by substituting the expressions for z, $|\Gamma|$, and p, and simplifying. By symmetry, $\Pr[|err_S(R)| \geq z] \leq 2\exp(-\frac{3}{8}c^2\epsilon^2 s)$. ◀

The rest of the proof appears in the full version of this article. ◀

5 Distance Approximation to the Nearest Connected Image

To define *connectedness*, we consider *the image graph* G_M of an image M. The vertices of G_M are $\{(i,j) \mid M[i,j] = 1\}$, and two vertices (i,j) and (i',j') are connected by an edge if $|i - i'| + |j - j'| = 1$. In other words, the image graph consists of black pixels connected by the grid lines. The image is *connected* if its image graph is connected.

▶ **Theorem 5.1.** *There is a block-uniform ϵ-additive distance approximation algorithm for connectedness with sample complexity $O(\frac{1}{\epsilon^4})$ and running time* $\exp\left(O\left(\frac{1}{\epsilon}\right)\right)$.

The first idea in our algorithms for connectedness is that we can modify an image by superimposing a grid on it, and as a result obtain a nearby image whose distance to connectedness is determined by the properties of individual squares into which the grid lines partition the image. The squares and the relevant property of the squares are defined next.

For a set $S \subset [0..n)^2$ and $(i,j) \in [0..n)^2$, we define $S + (i,j) = \{(x+i, y+j) : (x,y) \in S\}$.

▶ **Definition 5.2** (Squares and grid pixels)**.** Fix a side length $n \equiv 1 \pmod r$. For all integers $i, j \in [0..n-r)$, where i and j are divisible by r, the $(r-1)\times(r-1)$ image that consists of all pixels in $[r-1]^2 + (i,j)$ is called an *r-square* of M. The set of all r-squares of M is denoted S_r.

The pixels that do not lie in any squares of S_r, i.e., pixels (i,j) where i or j is divisible by r, are called *grid pixels*. The set of all grid pixels is denoted by GP_r.

▶ **Claim 5.3.** $|GP_r| \leq 2n^2/r$.

Note that a square consists of pixels of an r-block, with the pixels of the first row and column removed. Therefore, a block-uniform algorithm can obtain a uniformly random r-square.

Recall the definition of the border of an image from Section 2.

▶ **Definition 5.4** (Border connectedness)**.** A (sub)image S is *border-connected* if for every black pixel (i,j) of S, the image graph G_S contains a path from (i,j) to a pixel on the border. The property *border connectedness,* denoted $\mathcal{C}'$, is the set of all border-connected images.

The main idea behind Algorithm 3, used to prove Theorem 5.1, is to relate the distance to connectedness to the distance to another property, which we call *grid connectedness*. The latter distance is the average over squares of the distances of these squares to *border connectedness*. The average can be easily estimated by looking at a sample of the squares.

W.l.o.g. assume that $n \equiv 1 \pmod{4/\epsilon}$. (Otherwise, we can pad the image with white pixels without changing whether it is connected and adjust the accuracy parameter.)

Algorithm 3: Distance approximation to connectedness.

Input : $n \in \mathbb{N}$ and $\epsilon \in (0, 1/4)$; block-sample access to an $n \times n$ binary matrix M.

1 Sample $s = 4/\epsilon^2$ squares uniformly and independently from $S_{4/\epsilon}$ (see Definition 5.2).
`// Tho do this draw random blocks from the` $4/\epsilon$`-partition of` $[0..n)^2$`.`

2 For each such square S, compute $dist(S, \mathcal{C}')$, where $\mathcal{C}'$ is border connectedness (see Definition 5.4). Let $\hat{d}_{\text{squares}}$ be the average of computed distances $dist(S, \mathcal{C}')$.

3 **return** $\hat{d} = \left((1 - \frac{\epsilon}{4})(1 - \frac{1}{n})\right)^2 \cdot \hat{d}_{\text{squares}}$.

▶ **Definition 5.5.** Fix $\epsilon \in (0, 1/4)$. Let image M_ϵ be a *gridded image* obtained from image M as follows:

$$M_\epsilon[i,j] = \begin{cases} 1 & \text{if } (i,j) \text{ is a grid pixel from } \mathrm{GP}_{4/\epsilon}; \\ M[i,j] & \text{otherwise.} \end{cases}$$

Let $\mathcal{C}$ be the set of all connected images. For $\epsilon \in (0, 1/4)$, define *grid connectedness* $\mathcal{C}_\epsilon = \{M \mid M \in \mathcal{C}, \text{ and } M[i,j] = 1 \text{ for all } (i,j) \in \mathrm{GP}_{4/\epsilon}\}$.

▶ **Lemma 5.6.** *Let $d_M = dist(M, \mathcal{C})$ and $d_\epsilon = dist(M_\epsilon, \mathcal{C}_\epsilon)$. Then $d_M - \frac{\epsilon}{2} \le d_\epsilon \le d_M$. Moreover,*

$$d_\epsilon = \left(\left(1 - \frac{\epsilon}{4}\right)\left(1 - \frac{1}{n}\right)\right)^2 \cdot \frac{1}{|S_{4/\epsilon}|} \sum_{S \in S_{4/\epsilon}} dist(S, \mathcal{C}').$$

Proof. First, we prove that $d_\epsilon \le d_M$. Let M' be a connected image such that $dist(M, M') = d_M$. Then M'_ϵ, the gridded image obtained from M', satisfies $\mathcal{C}_\epsilon$. Since $dist(M_\epsilon, M'_\epsilon) \le d_M$, it follows that $d_\epsilon \le d_M$. Now we show that $d_M - \frac{\epsilon}{2} \le d_\epsilon$. Let $M''_\epsilon \in \mathcal{C}_\epsilon$ be such that $dist(M_\epsilon, M''_\epsilon) = d_\epsilon$. Then $M''_\epsilon \in \mathcal{C}$ and, by Claim 5.3, $dist(M, M''_\epsilon) \le |\mathrm{GP}_{4/\epsilon}|/n^2 + d_\epsilon \le \epsilon/2 + d_\epsilon$, implying $d_M \le \epsilon/2 + d_\epsilon$, as required.

Finally, observe that to make M_ϵ satisfy $\mathcal{C}_\epsilon$, it is necessary and sufficient to ensure that each square satisfies $\mathcal{C}'$. In other words,

$$d_\epsilon n^2 = \sum_{S \in S_{4/\epsilon}} Dist(S, \mathcal{C}') = (4/\epsilon - 1)^2 \sum_{S \in S_{4/\epsilon}} dist(S, \mathcal{C}').$$

Since $|S_{4/\epsilon}| = (\frac{n-1}{4/\epsilon})^2$, the desired expression for d_ϵ follows. ◀

The rest of the analysis is completed in the full version of this article.

References

1 Imre Barany. Extremal problems for convex lattice polytopes: a survey. *Contemporary Mathematics*, 2000.

2 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing convexity of figures under the uniform distribution. In *SoCG*, 2016.

3 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. L_p-testing. In *STOC*, pages 164–173, 2014. doi:10.1145/2591796.2591887.

4 Avrim Blum. Machine learning theory. Lecture notes. URL: http://www.cs.cmu.edu/~avrim/ML12/lect0201.pdf.

5 Andrea Campagna, Alan Guo, and Ronitt Rubinfeld. Local reconstructors and tolerant testers for connectivity and diameter. In *APPROX-RANDOM*, pages 411–424, 2013. doi:10.1007/978-3-642-40328-6_29.

6 Artur Czumaj and Christian Sohler. Property testing with geometric queries. In *Algorithms – ESA 2001, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, Proceedings*, pages 266–277, 2001. doi:10.1007/3-540-44676-1_22.

7 Artur Czumaj, Christian Sohler, and Martin Ziegler. Property testing in computational geometry. In *Algorithms – ESA 2000, 8th Annual European Symposium, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, pages 155–166, 2000. doi:10.1007/3-540-45253-2_15.

8 Andrzej Ehrenfeucht, David Haussler, Michael J. Kearns, and Leslie G. Valiant. A general lower bound on the number of examples needed for learning. *Inf. Comput.*, 82(3):247–261, 1989.

9 Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for boolean properties. *Theory of Computing*, 2(1):173–183, 2006. doi:10.4086/toc.2006.v002a009.

10 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

11 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.

12 David Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inf. Comput.*, 100(1):78–150, 1992. doi:10.1016/0890-5401(92)90010-D.

13 Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008.

14 Michael J. Kearns, Robert E. Schapire, and Linda Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994. doi:10.1007/BF00993468.

15 Igor Kleiner, Daniel Keren, Ilan Newman, and Oren Ben-Zwi. Applying property testing to an image partitioning problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(2):256–265, 2011. doi:10.1109/TPAMI.2010.165.

16 Simon Korman, Daniel Reichman, and Gilad Tsur. Tight approximation of image matching. *CoRR*, abs/1111.1713, 2011. URL: http://arxiv.org/abs/1111.1713.

17 Simon Korman, Daniel Reichman, Gilad Tsur, and Shai Avidan. Fast-match: Fast affine template matching. In *CVPR*, pages 2331–2338, 2013. doi:10.1109/CVPR.2013.302.

18 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. Syst. Sci.*, 72(6):1012–1042, 2006. doi:10.1016/j.jcss.2006.03.002.

19 Sofya Raskhodnikova. Approximate testing of visual properties. In *RANDOM-APPROX*, pages 370–381, 2003. doi:10.1007/978-3-540-45198-3_31.

20 Dana Ron and Gilad Tsur. Testing properties of sparse images. *ACM Trans. Algorithms*, 10(4):17:1–17:52, 2014. doi:10.1145/2635806.

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

22 Bernd Schmeltz. Learning convex sets under uniform distribution. In *Data Structures and Efficient Algorithms, Final Report on the DFG Special Joint Initiative*, pages 204–213, 1992.

23 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.

Erasure-Resilient Property Testing*†

Kashyap Dixit[1], Sofya Raskhodnikova[2], Abhradeep Thakurta[3], and Nithin Varma[4]

1 **Pennsylvania State University, University Park, USA**
kashyap@cse.psu.edu
2 **Pennsylvania State University, University Park, USA**
sofya@cse.psu.edu
3 **Work done while at Yahoo Labs**
guhathakurta.abhradeep@gmail.com
4 **Pennsylvania State University, University Park, USA**
nzm154@psu.edu

Abstract

Property testers form an important class of sublinear algorithms. In the standard property testing model, an algorithm accesses the input function $f : \mathcal{D} \mapsto \mathcal{R}$ via an oracle. With very few exceptions, all property testers studied in this model rely on the oracle to provide function values at all queried domain points. However, in many realistic situations, the oracle may be unable to reveal the function values at some domain points due to privacy concerns, or when some of the values get erased by mistake or by an adversary. The testers do not learn anything useful about the property by querying those *erased* points. Moreover, the knowledge of a tester may enable an adversary to erase some of the values so as to increase the query complexity of the tester arbitrarily or, in some cases, make the tester entirely useless.

In this work, we initiate a study of property testers that are resilient to the presence of *adversarially erased* function values. An α-erasure-resilient ε-tester is given parameters $\alpha, \varepsilon \in (0, 1)$, along with oracle access to a function f such that at most an α fraction of function values have been erased. The tester does not know whether a value is erased until it queries the corresponding domain point. The tester has to accept with high probability if there is a way to assign values to the erased points such that the resulting function satisfies the desired property $\mathcal{P}$. It has to reject with high probability if, for every assignment of values to the erased points, the resulting function has to be changed in at least an ε-fraction of the non-erased domain points to satisfy $\mathcal{P}$.

We design erasure-resilient property testers for a large class of properties. For some properties, it is possible to obtain erasure-resilient testers by simply using standard testers as a black box. However, there are more challenging properties for which all known testers rely on querying a specific point. If this point is erased, all these testers break. We give efficient erasure-resilient testers for several important classes of such properties of functions including monotonicity, the Lipschitz property, and convexity. Finally, we show a separation between the standard testing and erasure-resilient testing. Specifically, we describe a property that can be ε-tested with $O(\frac{1}{\varepsilon})$ queries in the standard model, whereas testing it in the erasure-resilient model requires number of queries polynomial in the input size.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

* A full version of the paper is available at http://arxiv.org/abs/1607.05786.

† This work was supported in part by NSF award CCF-1320814 , NSF CAREER award CCF-0845701, NSF award CCF-1422975, Pennsylvania State University College of Engineering Fellowship and Pennsylvania State University Graduate Fellowship.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 91; pp. 91:1–91:15

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

Keywords and phrases Randomized algorithms, property testing, error correction, monotone and Lipschitz functions

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.91

1 Introduction

In this paper, we revisit the question of how sublinear-time algorithms access their input. With very few exceptions, all algorithms studied in the literature on sublinear-time algorithms have *oracle* access to their input[1]. However, in many applications, this assumption is unrealistic. The oracle may be unable to reveal parts of the data due to privacy concerns, or when some of the values get erased by mistake or by an adversary. Motivated by these scenarios, we propose to study sublinear algorithms that work with partially erased data.

Formally, we view a dataset as a function over some discrete domain $\mathcal{D}$, such as $[n] = \{1, \ldots, n\}$ or $[n]^d$. For example, the classical problem of testing whether a list of n numbers is sorted in nondecreasing order can be viewed as a problem of testing whether a function $f : [n] \to \mathbb{R}$ is monotone (nondecreasing). Given a parameter $\alpha \in (0, 1)$, we say that a function is *α-erased* if at most an α fraction of its domain points are marked as "erased" or protected (that is, an algorithm is denied access to these values). An algorithm that takes an α-erased function as its input does not know which values are erased until it queries the corresponding domain points. For each queried point x, the algorithm either learns $f(x)$ or, if x is an erased point, gets back a special symbol $\perp$. We study algorithms that work in the presence of *adversarial erasures*. In other words, the query complexity of an algorithm is the number of queries it makes in the *worst case* over all α-erased input functions.

In this work, we initiate a systematic study of property testers that are resilient to the presence of adversarial erasures. An α-erasure-resilient ε-tester is given parameters $\alpha, \varepsilon \in (0, 1)$, along with oracle access to an α-erased function f. The tester has to accept with high probability if f can be restored to a function on the whole domain that satisfies the desired property $\mathcal{P}$ and reject with high probability if every restoration of f is ε-far from $\mathcal{P}$ on the nonerased part of the domain.

Generic transformations. Our first goal is to understand which existing algorithms in the standard property testing model [35, 24] can be easily made erasure-resilient. We show (in Section 2) how to obtain erasure-resilient testers for some properties by using standard testers for these properties as black box. Our transformations apply to testers that query uniformly and independently sampled points, with some additional restrictions. More specifically, our transformations work for uniform *proximity oblivious testers* (POTs) [25, 27] and uniform testers for *extendable properties*. As a result, we are able to obtain erasure-resilient testers for being a low-degree polynomial [35], monotonicity over general poset domains [22], convexity of black and white images [7], and having k runs of 0s and 1s.

Erasure-resilient testers for more challenging properties. One challenge in designing erasure-resilient testers by using existing algorithms in the standard model as a starting

[1] Sublinear-time algorithms with various distributional assumptions on the positions of the input the algorithms access have been investigated, for example, in [24, 4, 26]. There is also a line of work, initiated by [6], that studies sublinear algorithms that access distributions, as opposed to fixed datasets. In this work, we focus on fixed datasets.

point is that many existing algorithms are more likely to query certain points in the domain. Therefore, if these points are erased, the algorithms break. Specifically, the optimal algorithms for testing whether a list of numbers is sorted (and there are at least three different algorithms for this problem [17, 8, 13]) have this feature. Moreover, it is known that an algorithm that makes uniformly random queries is far from optimal: it needs $\Theta(\sqrt{n})$ queries instead of $\Theta(\log n)$ for n-element lists [17, 20].

There is a number of well studied properties for which all known optimal algorithms heavily rely on querying specific points. Most prominent examples include monotonicity, the Lipschitz properties and, more generally, bounded-derivative properties of real-valued functions on $[n]$ and $[n]^d$, as well as convexity of real-valued functions on $[n]$. It is especially challenging to deal with real-valued functions in our model, because there are many possibilities for erased values. We give efficient erasure-resilient testers for all aforementioned properties of real-valued functions in Sections 3–5.

Relationships with other models. In the full version of our paper, we provide a separation between our erasure-resilient model and the standard model. Specifically, we prove the existence of a property that can be tested with $O(1/\varepsilon)$ queries in the standard model, but requires polynomially many queries in the length of the input in the erasure-resilient model. This result builds on the ideas of Fischer and Fortnow [21] that separate tolerant testing, defined by Parnas et al. [32], from standard testing.

A tolerant tester for a property $\mathcal{P}$, given two parameters $\varepsilon_1, \varepsilon_2 \in (0,1)$, where $\varepsilon_1 < \varepsilon_2$, is required to, with probability at least $2/3$, accept inputs that are ε_1-close to $\mathcal{P}$ and reject inputs that are ε_2-far from $\mathcal{P}$. Intuitively, the relationship of our erasure-resilient model to tolerant testing is akin to the relationship between error-correcting codes that withstand erasures and error-correcting codes that withstand general errors. In the full version, we prove that the existence of tolerant testers implies the existence of erasure-resilient testers with related parameters. Using this implication and existing tolerant testers for sortedness [36], monotonicity [19], and convexity [18], we get erasure-resilient testers for these properties as corollaries. However, we obtain erasure-resilient testers for these properties with much better parameters in the technical sections of this article. We conjecture that erasure-resilient testing can be separated from tolerant testing in the same strong sense as in our separation of standard testing from erasure-resilient testing.

1.1 The Erasure-Resilient Testing Model

We formalize our erasure-resilient model for the case of property testing. Erasure-resilient versions of other computational models, such as tolerant testing, can be defined analogously.

▶ **Definition 1.1** (α-erased function)**.** Let $\mathcal{D}$ be a domain, $\mathcal{R}$ be a range, and $\alpha \in (0,1)$. A function[2] $f : \mathcal{D} \mapsto \mathcal{R} \cup \{\perp\}$ is *α-erased* if f evaluates to $\perp$ on at most an α fraction of domain points. The points on which f evaluates to $\perp$ are called *erased*. The set of remaining (nonerased) points is denoted by $\mathcal{N}$.

A function f is ε-far from a property (set) $\mathcal{P}$ if it needs to be changed on at least an ε fraction of domain points to obtain a function in $\mathcal{P}$. A function $f' : \mathcal{D} \to \mathcal{R}$ that differs from a function f only on points erased in f is called a *restoration* of f.

[2] Any object can be viewed as a function. E.g., an n-element array of real numbers can be viewed as a function $f : [n] \to \mathbb{R}$, an image—as a map from the plane to the set of colors, and a graph—as a map from the set of vertex pairs to $\{0,1\}$.

▶ **Definition 1.2** (Erasure-resilient tester)**.** An *α-erasure-resilient ε-tester* of property $\mathcal{P}$ gets input parameters $\alpha, \varepsilon \in (0,1)$ and oracle access to an α-erased function $f : \mathcal{D} \to \mathcal{R} \cup \{\perp\}$. It outputs, with probability[3] at least 2/3,

- **accept** if there is a restoration $f' : \mathcal{D} \to \mathcal{R}$ of f that satisfies $\mathcal{P}$;
- **reject** if every restoration $f' : \mathcal{D} \to \mathcal{R}$ of f needs to be changed on at least an ε fraction of $\mathcal{N}$, the nonerased portion of f's domain, to satisfy $\mathcal{P}$ (that is, f' is $\varepsilon \cdot \frac{|\mathcal{N}|}{|\mathcal{D}|}$-far from $\mathcal{P}$).

The tester has *1-sided error* if the first item holds with probability 1.

Let $f_{|\mathcal{N}}$ denote the function f restricted to the set $\mathcal{N}$ of nonerased points. In the full version, we show that if property $\mathcal{P}$ is extendable, we can define a property $\mathcal{P}_{\mathcal{N}}$ such that the erasure-resilient tester is simply required to distinguish the case that $f_{|\mathcal{N}}$ satisfies $\mathcal{P}_{\mathcal{N}}$ from the case that it is ε-far from satisfying it. For example, if $\mathcal{P}$ is monotonicity of functions on a partially-ordered domain $\mathcal{D}$ then $\mathcal{P}_{\mathcal{N}}$ is monotonicity of functions on $\mathcal{N}$. (Most of the properties we consider in this article, including monotonicity, Lipschitz properties and convexity, are extendable properties.) Note that, even for the case of extendable properties, our problem is different from the standard property testing problem because the tester does not know in advance which points are erased.

1.2 Properties We Consider

Next we define properties of real-valued functions considered in this article and summarize previous work on testing them. Most properties of real-valued functions studied in the property testing framework are for functions over the *line* domain $[n]$ and, more generally, the *hypergrid* domain $[n]^d$.

▶ **Definition 1.3** (Hypergrid, line)**.** Given $n, d \in \mathbb{N}$, the hypergrid of size n and dimension d is the set $[n]^d$ associated with an order relation $\preceq$, such that $x \preceq y$ for all $x, y \in [n]^d$ iff $x_i \leq y_i$ for all $i \in [d]$, where x_i (respectively y_i) denotes the i^{th} coordinate of x (respectively, y). The special case $[n]$ is called a *line*.

We consider domains that are subsets of $[n]^d$ to be able to handle arbitrary erasures on $[n]^d$.

Monotonicity. Monotonicity of functions, first studied in the context of property testing in [23], is one of the most widely investigated properties in this model [17, 16, 31, 22, 1, 20, 28, 5, 32, 2, 8, 11, 9, 13, 14, 10, 12]. A function $f : \mathcal{D} \mapsto \mathbb{R}$, defined on a partially ordered domain $\mathcal{D}$ with order $\preceq$, is monotone if $x \preceq y$ implies $f(x) \leq f(y)$ for all $x, y \in \mathcal{D}$. The query complexity of testing monotonicity of functions $f : [n] \mapsto \mathbb{R}$ is $\Theta(\log n/\varepsilon)$ [17, 20]; for functions $f : [n]^d \mapsto \mathbb{R}$, it is $\Theta(d \log n/\varepsilon)$ [13, 14], and for functions over arbitrary partially ordered domains $\mathcal{D}$, it is $O\left(\sqrt{|\mathcal{D}|/\varepsilon}\right)$ [22].

Lipschitz properties. Lipschitz continuity is defined for functions between arbitrary metric spaces, but was specifically studied for real-valued functions on hypergrid domains [29, 3, 13, 15, 10, 12] because of applications to privacy [29, 15]. For $\mathcal{D} \subseteq [n]^d$ and $c \in \mathbb{R}$, a function $f : \mathcal{D} \mapsto \mathbb{R}$ is c-Lipschitz if $|f(x) - f(y)| \leq c \cdot ||x - y||_1$ for all $x, y \in \mathcal{D}$, where $||x - y||_1$ is the L_1 distance between x and y. More generally, f is (α, β)-Lipschitz, where $\alpha < \beta$, if

[3] In general, the error probability can be any $\delta \in (0,1)$. For simplicity, we formulate our model and the results with $\delta = 1/3$. To get results for general δ, by standard arguments, it is enough to multiply the complexity of an algorithm by $\log 1/\delta$.

$\alpha \cdot ||x-y||_1 \leq |f(x)-f(y)| \leq \beta \cdot ||x-y||_1$ for all $x, y \in [n]^d$. All (α, β)-Lipschitz properties can be tested with $O(d \log n/\varepsilon)$ queries [13].

Bounded derivative properties. The class of bounded derivative properties (BDPs), defined by Chakrabarty et al. [12], is a natural generalization of monotonicity and the (α, β)-Lipschitz properties. An ordered set $\mathbf{B}$ of $2d$ functions $l_1, u_1, l_2, u_2, \ldots, l_d, u_d : [n-1] \mapsto \mathbb{R} \cup \{\pm\infty\}$ is a *bounding family* if for all $r \in [d]$ and $y \in [n-1]$, $l_r(y) < u_r(y)$. Let $\mathbf{B}$ be a bounding family of functions and let $\mathbf{e}_r$ be the unit vector along dimension r. The property $\mathcal{P}(\mathbf{B})$ of being $\mathbf{B}$-*derivative bounded* is the set of functions $f : [n]^d \mapsto \mathbb{R}$ such that $l_r(x_r) \leq f(x+\mathbf{e}_r) - f(x) \leq u_r(x_r)$ for all $r \in [d]$ and $x \in [n]^d$ with $x_r \neq n$, where x_r is the r^{th} coordinate of x. Chakrabarty et al. [12] showed that the complexity of testing BDPs of functions $f : [n]^d \mapsto \mathbb{R}$ is $\Theta(d \log n/\varepsilon)$.

Convexity of functions. A function $f : \mathcal{D} \mapsto \mathbb{R}$ is convex if $f(t\mathbf{x} + (1-t)\mathbf{y}) \leq tf(\mathbf{x}) + (1-t)f(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{D}$ and $t \in [0,1]$. If $\mathcal{D} \subseteq [n]$, equivalently, f is convex if $\frac{f(y)-f(x)}{y-x} \leq \frac{f(z)-f(y)}{z-y}$ for all $x < y < z$. Parnas et al. [33] gave a convexity tester for functions $f : [n] \mapsto \mathbb{R}$ with query complexity $O(\log n/\varepsilon)$. Blais et al. [10] gave an $\Omega(\log n)$ bound for nonadaptive testers for this problem.

1.3 Our Results

We give efficient erasure-resilient testers for all properties discussed in Section 1.2. All our testers have optimal complexity for the case with no erasures and have an additional benefit of not relying too heavily on the value of the input function at any specific point.

Monotonicity on the line. We start by giving (in Section 3) an erasure-resilient monotonicity tester on $[n]$.

▶ **Theorem 1.4** (Monotonicity tester on the line). *There exists a one-sided error α-erasure-resilient ε-tester for monotonicity of real-valued functions on the line $[n]$ that works for all $\alpha, \varepsilon \in (0,1)$, with query complexity $O\left(\frac{\log n}{\varepsilon(1-\alpha)}\right)$.*

Without erasure resilience, the complexity of testing monotonicity of functions $f : [n] \mapsto \mathbb{R}$ is $\Theta(\log n/\varepsilon)$ [17, 20]. Thus, the query complexity of our erasure-resilient tester has optimal dependence on the domain size and on ε.

The starting point of our algorithm is the tester for sortedness from [17]. This tester picks a random element of the input array and performs a binary search for that element. It rejects if the binary search does not lead to the right position. The first challenge is that the tester always queries the middle element of the array and is very likely to query other elements that are close to the root in the binary search tree. So, it will break if these elements are erased. To make it resilient to erasures, we randomize the binary tree with respect to which it performs the binary search. The second challenge is that the tester does not know which points are erased. To counteract that, our tester samples points from appropriate intervals until it encounters a nonerased point.

To analyze the tester, we bound the expected number of queries required to traverse a uniformly random search path in an arbitrary binary search tree built over the nonerased points in an α-erased n-element array (Claim 3.2). This expectation depends only on the depth of the tree and α. This is the most interesting part of our analysis and captures the intuition that a randomized binary search for a uniformly random search point is biased towards visiting intervals containing a larger fraction of nonerased points.

BDPs on the hypergrid. In Section 4, we generalize our monotonicity tester in two ways: (1) to work over general hypergrid domains, and (2) to apply to all BDPs. We achieve it by giving (1) a reduction from testing BDPs on the line to testing monotonicity on the line that applies to erasure-resilient testers and (2) an erasure-resilient version of the dimension reduction from [12]. We obtain the following result.

▶ **Theorem 1.5** (BDP tester on the hypergrid)**.** *For every BDP $\mathcal{P}$, there exists a one-sided error α-erasure-resilient ε-tester for $\mathcal{P}$ of real-valued functions on the hypergrid $[n]^d$ that works for all $\alpha, \varepsilon \in (0,1)$, where $\alpha \leq \varepsilon/970d$, with query complexity $O\left(\frac{d \log n}{\varepsilon(1-\alpha)}\right)$.*

Every known tester of a BDP for real-valued functions over general hypergrid domains work by sampling an *axis-parallel line* uniformly at random and checking for violations on the sampled line. Our erasure-resilient testers also follow this paradigm. To check for violations on the sampled line, we use one iteration of our BDP tester for the line. In the full version, we show the existence of α-erased functions $f : \{0,1\}^d \mapsto \mathbb{R}$ that are ε-far from monotone for $\alpha = \Theta(\varepsilon/\sqrt{d})$ but do not have violations to monotonicity along any of the axis parallel lines (which are the edges of the hypercube, in this case). It implies that every tester for monotonicity that follows the paradigm above will fail when $\alpha = \Omega(\varepsilon/\sqrt{d})$. Thus, some restriction on α in terms of d and ε is necessary for such testers.

Convexity on the line. Finally, in Section 5, we develop additional techniques to design a tester for convexity (which is not a BDP) on the line. The query complexity of our tester has the same dependence on n and ε as in the standard convexity tester of Parnas et al. [33]. The dependence on n is known to be optimal for nonadaptive testers [10], and the tester from [33] is conjectured to be optimal in the standard model.

▶ **Theorem 1.6** (Convexity tester on the line)**.** *There exists a one-sided error α-erasure-resilient ε-tester for convexity of real-valued functions on the line $[n]$ that works for all $\alpha, \varepsilon \in (0,1)$, with query complexity $O\left(\frac{\log n}{\varepsilon(1-\alpha)}\right)$.*

Our algorithm for testing convexity combines ideas on testing convexity from [33], testing sortedness from [17], and our idea of randomizing the search. The tester of [33] traverses a uniformly random path in a binary tree on the array $[n]$ by selecting one of the half-intervals of an interval uniformly at random at each step. Instead of doing this, our tester samples a uniformly random nonerased search point and traverses the path to that point in a uniformly random binary search tree just as in our modification of the tester of [17]. This is done to bias our algorithm to traverse paths containing intervals that have a larger fraction of nonerased points. However, instead of checking whether the selected point can be found, as in our monotonicity tester, the convexity tester checks a more complicated "goodness condition" in each visited interval of the binary search tree. It boils down to checking that the slope of the functions between pairs of carefully selected points satisfies the convexity condition. In addition to spending queries on erased points due to sampling, like in the monotonicity tester, our tester also performs "walking queries" to find the nearest nonerased points for the pivots in our random binary search tree. We show that the overhead in the query complexity due to querying erased points is at most a factor of $O(1/(1-\alpha))$.

2 Generic transformations

In this section, we state our transformation theorems that can be applied to some classes of testers that use only uniform samples. We show how to make two classes of testers erasure-resilient: (1) uniform proximity oblivious testers (POTs) defined in [25, 27] (Theorem 2.2),

and (2) uniform testers for extendable properties (Theorem 2.4). All omitted details appear in the full version. We first define POTs.

▶ **Definition 2.1** (POT [25, 27])**.** Let $\mathcal{P}$ be a property of functions of the form $f : \mathcal{D} \mapsto \mathcal{R}$. Let $\rho : (0, 1] \mapsto (0, 1]$ be a monotone function and $c \in (0, 1]$ be a constant. A tester T is a (ρ, c)-*POT* for $\mathcal{P}$ if it decides to accept or reject based only on the answers to the queries that it makes, such that

- for every function $f \in \mathcal{P}$, the probability that T accepts is at least c, and
- for every function $f \notin \mathcal{P}$, the probability that T accepts is at most $c - \rho(\varepsilon_f)$, where ε_f denotes the relative Hamming distance of f from $\mathcal{P}$.

A POT that queries points sampled uniformly and independently at random from $\mathcal{D}$ is called a uniform POT. Next, we state our first generic transformation.

▶ **Theorem 2.2.** *If T is a uniform (ρ,c)-POT for a property $\mathcal{P}$ of functions of the form $f : \mathcal{D} \mapsto \mathcal{R}$ making q queries, then there exists a uniform α-erasure-resilient (ρ',c)-POT T' for $\mathcal{P}$ that makes q queries for all $\alpha < \rho(\varepsilon_f \cdot (1 - \alpha))/q$, where $\rho'(x) = \rho(x \cdot (1 - \alpha)) - \alpha \cdot q$ for $x \in (0, 1]$.*

In the full version, we apply Theorem 2.2 to a tester for the property of being a polynomial of degree at most d due to Rubinfeld and Sudan [35] and get an erasure-resilient tester.

Next, we define extendable properties and state our second transformation. Given $\mathcal{S} \subseteq \mathcal{T}$, the *extension* of a function $f : \mathcal{S} \mapsto \mathcal{R}$ to $\mathcal{T}$ is a function $g : \mathcal{T} \mapsto \mathcal{R}$ that agrees with f on every point in $\mathcal{S}$.

▶ **Definition 2.3** (Extendable properties)**.** For a domain $\mathcal{D}$ and all $\mathcal{S} \subseteq \mathcal{D}$, let $\mathcal{P}_\mathcal{S}$ denote a property of functions $f : \mathcal{S} \mapsto \mathcal{R}$. The class of properties $\{\mathcal{P}_\mathcal{S} : \mathcal{S} \subseteq \mathcal{D}\}$ is extendable if, for all $\mathcal{S}, \mathcal{T} : \mathcal{S} \subseteq \mathcal{T} \subseteq \mathcal{D}$,

- for every function $f : \mathcal{S} \mapsto \mathcal{R}$ satisfying $\mathcal{P}_\mathcal{S}$, there is an extension $f' : \mathcal{T} \mapsto \mathcal{R}$ satisfying $\mathcal{P}_\mathcal{T}$, and
- for every function $f : \mathcal{S} \mapsto \mathcal{R}$ that is ε-far from $\mathcal{P}_\mathcal{S}$, every function $f' : \mathcal{T} \mapsto \mathcal{R}$ satisfying $\mathcal{P}_\mathcal{T}$ differs from f on at least an ε fraction of points in $\mathcal{S}$.

▶ **Theorem 2.4.** *Let $q(\cdot, \cdot)$ be a function that is monotone non-decreasing in the first argument and monotone non-increasing in the second argument. Let $\{\mathcal{P}_\mathcal{S} : \mathcal{S} \subseteq \mathcal{D}\}$ be a class of extendable properties, where $\mathcal{P}_\mathcal{S}$ is a property of functions of the form $f : \mathcal{S} \mapsto \mathcal{R}$. Suppose $T_\mathcal{S}$ is a uniform one-sided error ε-tester of $\mathcal{P}_\mathcal{S}$ that makes $q(|\mathcal{S}|, \varepsilon)$ queries from $\mathcal{S}$, for every $\mathcal{S} \subseteq \mathcal{D}$. Assume also that for each $\mathcal{S} \subseteq \mathcal{D}$, the probability that $T_\mathcal{S}$ tests correctly does not decrease when it makes more queries. Then, there exists a uniform one-sided error α-erasure-resilient ε-tester T' of $\mathcal{P}_\mathcal{D}$ that makes $O\left(q(|\mathcal{D}|, \varepsilon)/(1 - \alpha)\right)$ queries for all $\alpha \in [0, 1)$.*

Using Theorem 2.4, we can make the uniform tester for convexity of black and white images [7] and the uniform tester for monotonicity of real-valued functions over arbitrary partially ordered domains [22] erasure-resilient.

We also develop a uniform tester for the property of being a k-run Boolean function and make it erasure-resilient by applying Theorem 2.4. A function $f : [n] \mapsto \{0, 1\}$ has k *runs* if the list $f(1), f(2), \ldots, f(n)$ has at most $k - 1$ alternations of values. The problem at hand is to test whether a given function $f : [n] \mapsto \{0, 1\}$ has k runs or is ε-far from this property. Kearns and Ron [30] studied a relaxation of this problem. Specifically, they showed that $O(1/\varepsilon^2)$ queries suffice to test whether a Boolean function has k runs or is ε-far from being a k/ε-run function. They also developed a uniform $O(\sqrt{k}/\varepsilon^{2.5})$-query tester for this relaxation

and proved that every uniform ε-tester for the k-run property requires $\Omega(\sqrt{k})$ queries. Balcan et al. [4] obtained a $O(1/\varepsilon^4)$-query tester for this property in the active testing model. They also developed a uniform $O(\sqrt{k}/\varepsilon^5)$-query tester. We prove the following theorem.

▶ **Theorem 2.5.** *There is an ε-tester for the property of having k runs for functions of the form $f : [n] \mapsto \{0, 1\}$ that makes $O\left(\frac{k \log k}{\varepsilon}\right)$ uniform independent queries, where $\varepsilon > k^2/n$.*

3 Erasure-Resilient Monotonicity Tester for the Line

In this section, we prove Theorem 1.4. Recall that, for a function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$, the set of nonerased points (the ones that map to $\mathbb{R}$) is denoted by $\mathcal{N}$. The function f is monotone if $x < y$ implies $f(x) \leq f(y)$ for all $x, y \in \mathcal{N}$. The tester does not know $\mathcal{N}$ in advance.

We present our tester in Algorithm 1. It has oracle access to f and takes α and ε as inputs. In each iteration, it performs a randomized binary search for a nonerased index sampled uniformly at random (u.a.r.) from $\mathcal{N}$ and rejects if it finds violations to monotonicity. In the description of our tester, we use $I[i, j]$ to denote the set of natural numbers from i until and including j. We alternatively refer to it as the interval from i to j.

Algorithm 1 Erasure-Resilient Monotonicity Tester for the Line

1: **Set** $Q = \left\lceil \frac{60 \log n}{\varepsilon(1-\alpha)} \right\rceil$.
2: **Accept** at any point if the number of queries exceeds Q.
3: **loop** $2/\varepsilon$ times:
4: Sample points u.a.r. from $I[1, n]$ and query them until we get a point $s \in \mathcal{N}$.
5: **Set** $\ell \leftarrow 1$, $r \leftarrow n$.
6: **while** $\ell \leq r$ **do**
7: Sample points u.a.r. from $I[\ell, r]$ and query them until we get a point $m \in \mathcal{N}$.
8: **if** $s < m$ **then set** $r \leftarrow m - 1$ and **Reject** if $f(s) \geq f(m)$.
9: **if** $s > m$ **then set** $\ell \leftarrow m + 1$ and **Reject** if $f(s) \leq f(m)$.
10: **if** $s = m$ **then** Go to Step 3. ▷ Search completed.
11: **Accept**.

Every iteration of Algorithm 1 can be viewed as a traversal of a uniformly random search path in a uniformly random binary search tree defined on the set $\mathcal{N}$ of nonerased points. Given a binary search tree T over $\mathcal{N}$, we associate every node of T with a unique sub-interval I of $I[1, n]$ as follows. The root of T is associated with $I[1, n]$. Suppose the interval associated with a node Γ in T that contains $s \in \mathcal{N}$ is $I[i, j]$. Then the interval associated with the left child of Γ is $I[i, s - 1]$ and the interval associated with the right child of Γ is $I[s + 1, j]$. A search path is a path from the root to some node Γ of T.

If f is ε-far from monotone, we prove that, with high probability, the tester finds a violation. It is easy to prove this, using a generalization of an argument from [17], for the case when Algorithm 1 manages to complete all iterations of Step 3 before it runs out of queries. The challenge is that the algorithm might get stuck pursuing long paths in a random search tree and waste many queries on erased points. To resolve the issue of many possible queries to erased points, we prove an upper bound on the expected number of queries made while traversing a uniformly random search path in a binary search tree on $\mathcal{N}$. We combine this with the fact that the expected depth of a random binary search tree is $O(\log n)$ to obtain the final bound on the probability that the algorithm exceeds its query budget.

3.1 Analysis

We analyze the tester in this section. The query complexity of the tester is clear from its description. The main statement of Theorem 1.4 follows from Lemma 3.1, proved next.

▶ **Lemma 3.1.** *Algorithm 1 accepts if f is monotone, and rejects with probability at least $2/3$ if f is ε-far from monotone.*

Proof. The tester accepts whenever f is monotone. To prove the other part of the lemma, assume that f is ε-far from monotone. Let A be the event that the tester accepts f. Let q denote the total number of queries made. We prove that $\Pr[A] \leq 1/3$. The event A occurs if either $q > Q$ or the tester does not find a violation in any of the $2/\varepsilon$ iterations of Step 3. Thus, $\Pr[A] \leq \Pr[A|q \leq Q] + \Pr[q > Q]$.

First we bound the probability that the tester does not find a violation in one iteration of Step 3, conditioned on the event that $q \leq Q$. Consider an arbitrary binary search tree T defined over points in $\mathcal{N}$. A point $s \in \mathcal{N}$ is called *searchable* with respect to T if Algorithm 1 does not detect a violation to monotonicity while traversing the search path to s in T. Consider two indices $i, j \in \mathcal{N}$, where $i < j$, both searchable with respect to T. Let $a \in \mathcal{N}$ be the pivot corresponding to the lowest common ancestor of the leaves containing i and j. Since i and j are both *searchable*, it must be the case that $f(i) < f(a)$ and $f(a) < f(j)$ and hence, $f(i) < f(j)$. Thus, for every tree T, the function restricted to the domain points that are *searchable* with respect to T is monotone. Therefore, if f is ε-far from monotone, for every binary search tree T, at least an ε-fraction of the points in $\mathcal{N}$ are not *searchable*. Thus, the tester detects a violation with probability ε in each iteration. Consequently, $\Pr[A|q \leq Q] \leq (1-\varepsilon)^{\frac{2}{\varepsilon}} < 1/4$.

In the rest of the proof, we will bound $\Pr[q > Q]$. We will first state and prove a claim that bounds the expected number of queries to traverse a search path, for every binary search tree. Recall that a search path in a search tree T is a path from the root to some node in T. Let I be an interval associated with a node v of T and let α_I denote the fraction of erased points in I. The number of queries to be made to sample a nonerased point from I with uniform sampling is a geometric random variable with expectation $1/(1-\alpha_I)$. We define the *query-weight* of node v to be this expectation. The query-weight of a search path is the sum of query-weights of the nodes on the path (which is the expected number of queries that the algorithm makes while traversing that path).

▶ **Claim 3.2.** *Consider an arbitrary binary search tree T on $\mathcal{N}$ of height h. The expected query-weight of a uniformly random search path in T is at most $h/(1-\alpha)$.*

Proof. There are exactly $|\mathcal{N}|$ search paths in T. Let S denote the sum of query-weights of all the search paths. The expected query-weight is equal to $S/|\mathcal{N}|$.

Consider a node v in T associated with an interval I. There are $|I|(1-\alpha_I)$ nonerased points in I. The search paths from the root of T to all these nonerased points pass through v, and hence, the query-weight of v gets added to the query-weights of all of those paths. Therefore, the total contribution of v towards S is $|I|$, since the query-weight of v is $1/(1-\alpha_I)$. Note that the intervals associated with nodes at the same level of T are disjoint from each other. Therefore, the total contribution to S from all nodes on the same level of T is at most n. Hence the value of S is at most $n \cdot h$. Observe that this quantity is independent of the fraction of erasures α. Therefore, the expected query-weight of a search path is at most $n \cdot h/|\mathcal{N}|$, which is at most $h/(1-\alpha)$, since $|\mathcal{N}| \geq n \cdot (1-\alpha)$. ◀

We will next see a fact on the expected depth of a uniformly random binary search tree and combine it with the above claim to prove the required bound on the expected query-weight of a uniformly random search path in a uniformly random binary search tree.

▶ **Claim 3.3** ([34]). *If H_n is the random variable denoting the height of a random binary search tree on n nodes, then $E[H_n] \leq 5 \log n$.*

▶ **Corollary 3.4.** *The expected number of queries made by Algorithm 1 to traverse a uniformly random search path in a uniformly random binary search tree on $\mathcal{N}$ is at most $5 \log n/(1-\alpha)$.*

By linearity of expectation, the expected number of queries made by the tester over all its iterations is at most $10 \log n/(\varepsilon \cdot (1-\alpha))$. Applying Markov's inequality to q, we can then see that $\Pr[q > Q] \leq 1/6$. Therefore, the probability of the tester not finding a violation is at most $1/3$. This completes the proof of the lemma. ◀

4 Erasure-Resilient Monotonicity Testers for the Hypergrid

In this section, we present our erasure-resilient tester for monotonicity over hypergrid domains and prove the following theorem, which is a special case of Theorem 1.5. We defer the discussion of erasure-resilient testers for all BDPs to the full version.

▶ **Theorem 4.1.** *There exists a one-sided error α-erasure-resilient ε-tester for monotonicity of real-valued functions on the hypergrid $[n]^d$ that works for all $\alpha, \varepsilon \in (0,1)$, where $\alpha \leq \varepsilon/250d$, with query complexity $O(\frac{d \log n}{\varepsilon(1-\alpha)})$.*

Let $\mathcal{L}$ denote the set of all *axis-parallel lines* in the hypergrid. Our monotonicity tester, which is described in Algorithm 2, samples an *axis-parallel line* uniformly at random in each iteration and does a randomized binary search for a uniformly randomly sampled nonerased point on that line. It rejects if and only if a violation to monotonicity is found within its query budget. To analyze the tester, we first state two important properties of a uniformly random axis-parallel line in Lemma 4.2 and Lemma 4.3, which we jointly call the erasure-resilient dimension reduction. The statements and proofs of more general versions of these lemmas, applicable to all BDPs, are given in the full version.

▶ **Lemma 4.2** (Dimension reduction: distance). *Let ε_f be the relative Hamming distance of an α-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$ from monotonicity. Given an axis-parallel line $\ell \in \mathcal{L}$, let $f_\ell : [n] \mapsto \mathbb{R} \cup \{\perp\}$ denote the restriction of f to ℓ and let ε_ℓ denote the relative Hamming distance of f_ℓ from monotonicity. Then $E_{\ell \sim \mathcal{L}}[\varepsilon_\ell] \geq (((1-\alpha) \cdot \varepsilon_f)/4d) - \alpha$.*

▶ **Lemma 4.3** (Dimension reduction: fraction of erasures). *Consider an α-erased function $f : [n]^d \mapsto \mathbb{R} \cup \{\perp\}$. Given $\ell \in \mathcal{L}$, let α_ℓ denote the fraction of erased points in ℓ. Then, for every $\eta \in (0,1)$, we have, $\Pr_{\ell \sim \mathcal{L}}[\alpha_\ell > \alpha/\eta] \leq \eta$.*

The query complexity of the tester is evident from its description. We will now prove its correctness in the following lemma, which will then imply Theorem 4.1.

▶ **Lemma 4.4.** *Algorithm 2 accepts if f is monotone, and rejects with probability at least $2/3$ if f is ε-far from monotone.*

Proof. The tester accepts if f is monotone. So, assume f is ε-far from being monotone. Let A denote the event that no iteration of the tester finds a violation to monotonicity. If q denotes the total number of queries made by the tester, we have, $\Pr[A] \leq \Pr[A | q \leq Q] + \Pr[q > Q]$.

Algorithm 2 Erasure-Resilient Monotonicity Tester for $[n]^d$

Require: parameters $\varepsilon \in (0,1), \alpha \in [0, \varepsilon/250d]$; oracle access to $f : [n]^d \to \mathbb{R}$

1: **Set** $Q = \lceil \frac{1200d \cdot \log n}{\varepsilon(1-\alpha)} \rceil$.
2: **loop** $\frac{12d}{\varepsilon(1-\alpha)-4d\alpha}$ times:
3: Sample a line $\ell \in \mathcal{L}$ uniformly at random.
4: Sample and query points u.a.r. from ℓ and query them until we get a point $s \in \mathcal{N}$.
5: Perform a randomized binary search for s on ℓ as in Algorithm 1.
6: **Reject** if any violation to monotonicity is found.
7: **Accept** at any point if the number of queries exceed Q.

Let t denote the number of iterations of the tester. Let A_i denote the event that the tester does not find a violation in its i-th iteration. For $\ell \in \mathcal{L}$, let f_ℓ denote f restricted to the line ℓ. Let ε_ℓ denote the relative Hamming distance of f_ℓ from monotonicity. We have, $\Pr[A_i | q \leq Q] = \sum_{\ell \in \mathcal{L}} (1 - \varepsilon_\ell) \Pr[\ell] = 1 - \mathrm{E}_{\ell \sim \mathcal{L}}[\varepsilon_\ell]$. By Lemma 4.2 and the fact that $\varepsilon_f \geq \varepsilon$, we have, $\mathrm{E}_{\ell \sim \mathcal{L}}[\varepsilon_\ell] \geq \frac{(1-\alpha)\cdot\varepsilon_f}{4d} - \alpha \geq \frac{(1-\alpha)\cdot\varepsilon}{4d} - \alpha$. Therefore,

$$\Pr[A | q \leq Q] = \prod_{i=1}^{t} \Pr[A_i | q \leq Q] \leq \left(1 - \frac{(1-\alpha)\cdot\varepsilon - 4d\alpha}{4d}\right)^t < \frac{1}{10}.$$

It remains to bound $\Pr[q > Q]$. Let $\eta = 1/10t$. Let α_i be the fraction of erasures in the line sampled during iteration i and let q_i be the number of queries the tester makes in iteration i. Let G be the (good) event that $\alpha_i \leq \alpha/\eta$ for all iterations $i \in [t]$. By Corollary 3.4, $\mathrm{E}[q_i | G] \leq 5\eta \cdot \log n/(\eta - \alpha)$. By the linearity of expectation, $\mathrm{E}[q|G] \leq \log n/(2(\eta - \alpha)) \leq 120d \log n/(\varepsilon(1-\alpha))$, where the last inequality follows from our assumption that $\alpha \leq \varepsilon/250d$. Using Markov's inequality, $\Pr[q > Q | G] \leq 1/10$. Also, by combining Lemma 4.3 with a union bound, we can see that $\Pr[\overline{G}] \leq 1/10$. Therefore, $\Pr[q > Q] \leq \Pr[q > Q|G] + \Pr[\overline{G}] \leq 1/5$. ◀

5 Erasure-Resilient Convexity Tester for the Line

In this section, we prove Theorem 1.6. Given an α-erased function $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$, let ν_i be the i-th nonerased domain point in $[n]$. The derivative of f at a point $\nu_i \in \mathcal{N}$, denoted by $\Delta f(\nu_i)$, is $\frac{f(\nu_{i+1}) - f(\nu_i)}{\nu_{i+1} - \nu_i}$, whenever $\nu_{i+1} \leq n$. The function f is convex iff $\Delta f(\nu_i) \leq \Delta f(\nu_{i+1})$ for all $i \in [|\mathcal{N}| - 2]$. Our tester builds upon the ideas in the convexity tester from [33].

A high level idea of the tester is as follows. Our tester (Algorithm 3) has several iterations. Every iteration of the tester can be thought of as a traversal of a uniformly random *search* path of a uniformly random binary search tree on $\mathcal{N}$, just as Algorithm 1. For each interval on such a path, we check a set of conditions computed based on the values at some nonerased points in the interval, called *anchor points*, and two real numbers, called the left and right slopes. More specifically, we verify that the function restricted to the sampled nonerased points in the interval is convex, by comparing the slopes across consecutive points. The algorithm accepts if all the intervals it sees pass these checks.

The main steps in the analysis of the tester follows that of the analysis of Algorithm 1. To prove that, with high probability, the algorithm does not run out of its budget of queries Q, we classify the queries that the tester makes into two kinds and analyze them separately. The queries where the tester repeatedly samples and queries from an interval until it finds a nonerased domain point are called *sampling queries*. The queries where the tester keeps querying consecutive points, starting from a nonerased point, until it gets the next nonerased

Algorithm 3 Erasure-Resilient Convexity Tester

Require: parameters $\varepsilon, \alpha \in (0,1)$; oracle access to $f : [n] \mapsto \mathbb{R} \cup \{\perp\}$.

Set $Q = \lceil \frac{180 \log n}{\varepsilon(1-\alpha)} \rceil$.
Accept at any point if the number of queries exceeds Q.
loop $2/\varepsilon$ times
    Sample points in $I[1,n]$ u.a.r and query them until we get a point $s \in \mathcal{N}$.
    Test-Interval$(I[1,n], \emptyset, -\infty, +\infty, s)$ and **Reject** if it rejects.
Accept.

point are called *walking queries.* We show that the expected number of walking queries is at most twice the number of the expected number of the sampling queries and use Corollary 3.4 to bound the expected number of sampling queries. We then prove that, conditioned on the tester not running out of its queries, in every iteration, with probability ε, the tester will detect a violation while testing for a function that is ε-far from being convex. This part draws ideas from the proof of correctness of the tester in [33]. The analysis of the tester is deferred to the full version.

Procedure 4 Test-Interval$(I[i,j], \mathcal{A} = \{a_1, a_2, \ldots, a_k\}, m_\ell, m_r, s)$

Require: interval $I[i,j]$; a set of nonerased points $\mathcal{A}$; left slope $m_\ell \in \mathbb{R}$; right slope $m_r \in \mathbb{R}$; search point $s \in \mathcal{N}$.

Sample points u.a.r. from $I[i,j]$ and query them until we get a point $x \in \mathcal{N}$.
Sequentially query points $x+1, x+2 \ldots$ until we get a nonerased point y.
Sequentially query points $x-1, x-2 \ldots$ until we get the nonerased point z.
Let $(a_1, a_2, \ldots, a_k)$ denote the sorted list of points in the set $\mathcal{A} \cup \{x, y, z\}$.
Let $m_i = (f(a_{i+1}) - f(a_i))/(a_{i+1} - a_i)$ for all $i \in [k-1]$.
Reject if $m_\ell \le m_1 \le m_2 \le \cdots \le m_{k-1} \le m_r$ is not true.
Let $\mathcal{A}'_\ell$ and $\mathcal{A}'_r$ be the sets of points in $\mathcal{A}$ that are smaller and larger than x, respectively.
if $s < x$ **then**
    Reject if Test-Interval$(I[i,z], \mathcal{A}'_\ell, m_\ell, \Delta f(z), s)$ rejects.
if $s > x$ **then**
    Reject if Test-Interval$(I[y,j], \mathcal{A}'_r, \Delta f(x), m_r, s)$ rejects.
Accept.

6 Conclusions and Open Problems

In this paper, we initiate a study of property testing in the presence of adversarial erasures. We design efficient erasure-resilient testers for several important properties such as monotonicity, the Lipschitz properties and convexity over different domains. All our testers for properties of functions on the line domain work for an arbitrary fraction of erasures. All our testers have only a small additional overhead of $O(1/(1-\alpha))$ in their query complexity in comparison to the query complexity of the currently best, and, in some cases, optimal, standard testers for the same properties. We also show that not all properties are easy to test in the erasure-resilient testing model by proving the existence of a property that is easy to test in the standard model but hard to test in the erasure-resilient model even for a small fraction of erasures. We now list some open problems.

- We show that tolerant testing is at least as hard as erasure-resilient testing. Determining if tolerant testing is strictly harder than erasure-resilient testing is an interesting direction.

- The fraction of erasures that our monotonicity tester for hypergrid domains ($[n]^d$) can tolerate decreases inversely with d. We also show that an inverse dependence on $\sqrt{d}$ is necessary for testers that work by sampling axis-parallel lines uniformly at random and then test for the property on them. It is an interesting combinatorial question to determine the exact tradeoff between the fraction of erasures and the fraction of axis parallel lines that are far from monotone.

Acknowledgements. We thank Jalaj Upadhyay for comments on a draft of this article.

References

1. N. Ailon and B. Chazelle. Information theory in property testing and monotonicity testing in higher dimension. *Inform. and Comput.*, 204(11):1704–1717, 2006.
2. N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Estimating the distance to a monotone function. *Random Structures Algorithms*, 31(3):371–383, 2007.
3. Pranjal Awasthi, Madhav Jha, Marco Molinaro, and Sofya Raskhodnikova. Testing Lipschitz functions on hypergrid domains. *Algorithmica*, 74(3):1055–1081, 2016. doi:10.1007/s00453-015-9984-y.
4. Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 21–30, 2012. doi:10.1109/FOCS.2012.64.
5. T. Batu, R. Rubinfeld, and P. White. Fast approximate PCPs for multidimensional bin-packing problems. *Inform. and Comput.*, 196(1):42–56, 2005.
6. Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing closeness of discrete distributions. *J. ACM*, 60(1):4, 2013. doi:10.1145/2432622.2432626.
7. Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. *Testing Convexity of Figures Under the Uniform Distribution*, 2015. To appear in *SoCG 2016*.
8. Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM J. Comput.*, 41(6):1380–1425, 2012.
9. E. Blais, J. Brody, and K. Matulef. Property testing lower bounds via communication complexity. *Comp. Complexity*, 21(2):311–358, 2012.
10. Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 309–320, 2014.
11. J. Briët, S. Chakraborty, D. García-Soriano, and A. Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.
12. Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1809–1828, 2015. doi:10.1137/1.9781611973730.121.
13. Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and lipschitz testing over hypercubes and hypergrids. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 419–428, 2013. doi:10.1145/2488608.2488661.
14. Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014. doi:10.4086/toc.2014.v010a017.

15 Kashyap Dixit, Madhav Jha, Sofya Raskhodnikova, and Abhradeep Thakurta. Testing the lipschitz property over product distributions with applications to data privacy. In *TCC*, pages 418–436, 2013. doi:10.1007/978-3-642-36594-2_24.

16 Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *Randomization, Approximation, and Combinatorial Algorithms and Techniques, Third International Workshop on Randomization and Approximation Techniques in Computer Science, and Second International Workshop on Approximation Algorithms for Combinatorial Optimization Problems RANDOM-APPROX'99, Berkeley, CA, USA, August 8-11, 1999, Proceedings*, pages 97–108, 1999.

17 Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000. doi:10.1006/jcss.1999.1692.

18 Shahar Fattal and Dana Ron. Approximating the distance to convexity. Unpublished manuscript. Uploaded at http://www.eng.tau.ac.il/~danar/Public-pdf/app-conv.pdf.

19 Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Transactions on Algorithms*, 6(3), 2010. doi:10.1145/1798596.1798605.

20 E. Fischer. On the strength of comparisons in property testing. *Inform. and Comput.*, 189(1):107–116, 2004.

21 Eldar Fischer and Lance Fortnow. Tolerant versus intolerant testing for boolean properties. *Theory of Computing*, 2(9):173–183, 2006. doi:10.4086/toc.2006.v002a009.

22 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, STOC'02, pages 474–483, New York, NY, USA, 2002. ACM. doi:10.1145/509907.509977.

23 O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. *Combinatorica*, 20:301–337, 2000.

24 O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.

25 Oded Goldreich and Dana Ron. On proximity-oblivious testing. *SIAM J. Comput.*, 40(2):534–566, 2011. doi:10.1137/100789646.

26 Oded Goldreich and Dana Ron. On sample-based testers. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 337–345, 2015. doi:10.1145/2688073.2688080.

27 Oded Goldreich and Igor Shinkar. Two-sided error proximity oblivious testing. *Random Struct. Algorithms*, 48(2):341–383, 2016. doi:10.1002/rsa.20582.

28 S. Halevy and E. Kushilevitz. Testing monotonicity over graph products. *Random Structures Algorithms*, 33(1):44–67, 2008.

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

30 Michael J. Kearns and Dana Ron. Testing problems with sublearning sample complexity. *J. Comput. Syst. Sci.*, 61(3):428–456, 2000. doi:10.1006/jcss.1999.1656.

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

32 M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *J. Comput. System Sci.*, 6(72):1012–1042, 2006.

33 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. On testing convexity and submodularity. *SIAM J. Comput.*, 32(5):1158–1184, 2003. doi:10.1137/S0097539702414026.

34 Bruce A. Reed. The height of a random binary search tree. *J. ACM*, 50(3):306–332, 2003. doi:10.1145/765568.765571.

35 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. URL: 10.1137/S0097539793255151, doi:10.1137/S0097539793255151.

36 Michael E. Saks and C. Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 458–467, 2010. doi:10.1109/FOCS.2010.51.

Towards Tight Lower Bounds for Range Reporting on the RAM*†

Allan Grønlund[1] and Kasper Green Larsen[2]

1 **Department of Computer Science, Aarhus University, Aarhus, Denmark**
`jallan@cs.au.dk`
2 **Department of Computer Science, Aarhus University, Aarhus, Denmark**
`larsen@cs.au.dk`

Abstract

In the orthogonal range reporting problem, we are to preprocess a set of n points with integer coordinates on a $U \times U$ grid. The goal is to support reporting all k points inside an axis-aligned query rectangle. This is one of the most fundamental data structure problems in databases and computational geometry. Despite the importance of the problem its complexity remains unresolved in the word-RAM.

On the upper bound side, three best tradeoffs exist, all derived by reducing range reporting to a ball-inheritance problem. Ball-inheritance is a problem that essentially encapsulates all previous attempts at solving range reporting in the word-RAM. In this paper we make progress towards closing the gap between the upper and lower bounds for range reporting by proving cell probe lower bounds for ball-inheritance. Our lower bounds are tight for a large range of parameters, excluding any further progress for range reporting using the ball-inheritance reduction.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Data Structures, Lower Bounds, Cell Probe Model, Range Reporting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.92

1 Introduction

In the orthogonal range reporting problem, we are to preprocess a set of n points with integer coordinates on a $U \times U$ grid. The goal is to support reporting all k points inside an axis-aligned query rectangle. This is one of the most fundamental data structure problems in databases and computational geometry. Given the importance of the problem, it has been extensively studied in all the relevant models of computation including e.g. the word-RAM, pointer machine and external memory model. In the latter two models, we typically work under an assumption of *indivisibility*, meaning that input points have to be stored as they are, i.e. compression techniques such as rank-space reduction and word-packing cannot be used to reduce the space consumption of data structures. The indivisibility assumption greatly alleviates the task of proving lower bounds, which has resulted in a completely tight characterisation of the complexity of orthogonal range reporting in these two models. More specifically, Chazelle [7] presented a pointer machine data structure answering queries in

* A full version of this paper is available at `http://arxiv.org/abs/1411.0644`.
† Both authors are supported in part by Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, grant DNRF84. Larsen is also supported by a Villum Young Investigator Grant and an AUFF Starting Grant. Grønlund is also supported by an Industrial Post Doc Grant from Innovation Fund Denmark.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 92; pp. 92:1–92:12


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

optimal $O(\lg n + k)$ time using $O(n \lg n / \lg \lg n)$ space and later proved that this space bound is optimal for any query time of the form $O(\lg^c n + k)$, where $c \geq 1$ is an arbitrary constant [8]. In the external memory model, Arge et al. [2] presented a data structure answering queries in optimal $O(\lg_B n + k/B)$ I/Os with $O(n \lg n / \lg \lg_B n)$ space and also proved the space bound to be optimal for any query time of the form $O(\lg_B^c n + k/B)$, where $c \geq 1$ is a constant. Here B is the disk block size. Thus the orthogonal range reporting problem has been completely closed for at least 15 years in both these models of computation. If we instead abandon the indivisibility assumption and consider orthogonal range reporting in the arguably more realistic model of computation, the word-RAM, our understanding of the problem is much more disappointing. Assuming the coordinates are polynomial in n ($U = n^{O(1)}$), the current best word-RAM data structures, by Chan et al. [5], achieve the following tradeoffs:

1. Optimal query time $O(\lg \lg n + k)$ with $O(n \lg^\varepsilon n)$ words of space for any constant $\varepsilon > 0$.
2. Query time $O((1 + k) \lg \lg n)$ with $O(n \lg \lg n)$ words of space.
3. Query time $O((1 + k) \lg^\varepsilon n)$ with optimal $O(n)$ words of space.

Thus we can achieve linear space by paying a $\lg^\varepsilon n$ penalty per point reported. And even if we insist on an optimal $O(\lg \lg n + k)$ query time, it is possible to improve over the optimal space bound in the pointer machine and external memory model by almost a $\lg n$ factor. Naturally the improvements rely heavily on points not being indivisible.

On the lower bounds side, Pătraşcu and Thorup [12, 14] proved that the query time must be $\Omega(\lg \lg n + k)$ for space $n \lg^{O(1)} n$. This lower bound was obtained by reduction from the predecessor search problem. For predecessor search, the query time of $\lg \lg n$ is known to be achievable with linear space. Thus the reduction is incapable of distinguishing the three space regimes of the current best data structures for range reporting. Perhaps it might just be possible to construct a linear space data structure with $O(\lg \lg n + k)$ query time. This would have a huge impact in practice, since the non-linear space solutions are most often abandoned for the kd-trees [3], using linear space and answering queries in $O(\sqrt{n} + k)$ time. This is simply because more than a constant factor above linear space is prohibitive for most applications. Thus ruling out the existence of fast linear space data structures would be a major contribution. The focus of this paper is on understanding this gap and the complexity of orthogonal range reporting in the word-RAM. This boils down to understanding how much compression and word-packing techniques can help us in the regime between linear space and $O(n \lg^\varepsilon n)$ space. Since our results concern definitions made by Chan et al. [5], we first give a more formal definition of the word-RAM and briefly review the technique of rank space reduction and the main ideas in [5].

1.1 Range Reporting in the word-RAM

The word-RAM model was designed to mimic what is possible in modern imperative programming languages such as C. In the word-RAM, the memory is divided into words of $\Theta(\lg n)$ bits. The words have integer addresses and we allow random access to any word in constant time. We also assume all standard word operations from modern programming languages takes constant time. This includes e.g. integer addition, subtraction, multiplication, division, bit-wise AND, OR, XOR, SHIFT etc. Having $\Theta(\lg n)$ bit words is a reasonable assumption since machine words on standard computers have enough bits to address the input and to store pointers into a data structure.

Rank Space Reduction. Most of the previous range reporting data structures for the word-RAM have used rank space reduction (or variants thereof) to save space, see e.g. [1, 11]. Rank space reduction is the following: Given a set P of n points on a $U \times U$ grid, compute

for each point $(x, y) \in P$ the rank $r_x(x)$ of x amongst the x-coordinates of points in P. Similarly compute the rank $r_y(x)$ of y amongst the y-coordinates of points in P. Construct a new point set P^* with each point $(x, y) \in P$ replaced by $(r_x(x), r_y(y))$. The point set P^* is said to be in *rank space*. A point $(x, y) \in P$ lies inside a query range $q = [x_0; x_1] \times [y_0; y_1]$ precisely if $(r_x(x), r_y(y))$ lies inside the range $q^* = [r_x(x_0); r_x(x_1)] \times [r_y(y_0); r_y(y_1)]$. Thus if we store a data structure for mapping q to q^* and a table mapping points in P^* back to points in P, the output of a query q can be computed from the output of the query q^* on P^*. Since the coordinates of a point in P^* can be represented using $\lg n$ bits, this gives a saving in space if $\lg n \ll \lg U$.

In previous range reporting data structures, rank space reductions are often used recursively on smaller and smaller point sets $P_t \subset P_{t-1} \subset \cdots \subset P_1 \subset P$. Applying t rounds of rank space reduction however results in a query time of $O(f(n) + tk)$ since each reported point has to be *decompressed* through t rank space reduction tables.

The Ball-Inheritance Problem. In the following, we present the main ideas of the current best data structures, due to Chan et al. [5]. Their solution is based on an elegant way of combining rank space reductions over all levels of a range tree:

Construct a complete binary tree with the points of P stored in the leaves ordered from left to right by their x-coordinate. Every internal node v is associated with the subset of points P_v stored in the leaves of the subtree rooted at v. For every internal node v, map the points P_v to rank space and denote the resulting set of points P_v^*. Store in v a data structure for answering 3-sided range queries on P_v^*. Here a 3-sided query is either of the form $[x_0; \infty) \times [y_0; y_1]$ or $(-\infty, x_1] \times [y_0; y_1]$. If we require that only the rank space y-coordinate of a point is reported (and not the rank space x-coordinate), these 3-sided data structures can be implemented in $O(n)$ bits and with $O(k)$ query time using succinct data structures for range minimum queries, see e.g. [9]. For each leaf, we simply store the associated point. The total space usage is $O(n \lg n + n \lg U)$ bits, which is $O(n)$ words.

To answer a query $q = [x_0; x_1] \times [y_0; y_1]$, find the lowest common ancestor, w, of the leaves storing the successor of x_0 and the predecessor of x_1 respectively. Let w_ℓ be the left child of w and w_r the right child. The points inside q are precisely the points $P_{w_\ell} \cap [x_0; \infty) \times [y_0; y_1]$ plus $P_{w_r} \cap (-\infty, x_1] \times [y_0; y_1]$. The data structures of Chan et al. now proceeds by mapping these two 3-sided queries to rank space amongst points in $P^*_{w_\ell}$ and $P^*_{w_r}$ respectively and answering the two queries using the 3-sided data structures stored at w_ℓ and w_r. This reports, for every point $(x, y) \in P_{w_\ell} \cap q$ (and $(x, y) \in P_{w_r} \cap q$), the rank of y amongst the y-coordinates of all points in P_{w_ℓ} (P_{w_r}). Assuming one can build an S word auxiliary data structure that supports mapping these rank space y-coordinates back to the original points in t time per point (i.e. through t rank space decompressions), this gives a data structure for orthogonal range reporting that answers queries in $O(\lg \lg n + t(1 + k))$ time using $S + O(n)$ space, see [5] for full details. Chan et al. named this abstract decompression problem *the ball-inheritance problem* and defined it as follows:

▶ **Definition 1** (Chan et al. [5])**.** In the ball-inheritance problem, the input is a complete binary tree with n leaves. In the root node, there is an ordered list of n balls. Each ball is associated with a unique leaf of the tree and we say the ball *reaches* that leaf. Every internal node v also has an associated list of balls, containing those balls reaching a leaf in the subtree rooted at v. The ordering of the balls in v's list is the same as their ordering in the root's list. We think of each ball in v's list as being inherited from v's parent.

A query is specified by a pair (v, i) where v is a node in the tree and i is an index into v's list of balls. The goal is to return the index of the leaf reached by the i'th ball in v's list of balls.

It is not hard to see that a solution to the ball-inheritance problem is precisely what is needed in Chan et al.'s data structures: We have one ball per point. The ball corresponding to a point (x, y) reaches the $r_x(x)$'th leaf, where $r_x(x)$ is the rank of x amongst all x-coordinates. The ordering of the balls inside the lists is just the ordering on the y-coordinates of the corresponding points. Thus answering a ball-inheritance query (v, i) corresponds exactly to determining the leaf storing the point from P_v having a rank space y-coordinate of i. Since Chan et al. stored the points in the leaves, this also recovers the original point.

All three tradeoffs by Chan et al. come from solving the ball-inheritance problem with the following bounds:

▶ **Theorem 2** (Chan et al. [5]). *For any $2 \leq B \leq \lg^{\varepsilon} n$, we can solve the ball-inheritance problem with:* (1) *space $O(nB \lg\lg n)$ and query time $O(\lg_B \lg n)$; or* (2) *space $O(n \lg_B \lg n)$ and query time $O(B \lg\lg n)$.*

While not all previous range reporting data structures directly solve the ball-inheritance problem, they are all based on rank space reductions and decompression of one point at a time, just in less efficient ways. Thus the ball-inheritance problem in some sense captures the essence of all previous approaches to solving range reporting and the bounds obtained for the ball-inheritance problem also sets the current limits for orthogonal range reporting.

We remark that the ball-inheritance problem also has been used to improve the upper bounds for various other problems with a range reporting flavor to them, see e.g. [6, 4]. Thus in light of the lack of progress in proving tight lower bounds for range reporting, it seems like a natural goal to understand the complexity of the ball-inheritance problem.

1.2 Our Results

In this paper, we prove a lower bound for the ball-inheritance problem. Our lower bound is tight for a large range of parameters and is as follows:

▶ **Theorem 3.** *Any word-RAM data structure for the ball-inheritance problem which uses S words of space, must have query time t satisfying:*

$$t = \Omega\left(\frac{\lg\lg n}{\lg(S/n) + \lg\lg\lg n}\right).$$

Comparing to the ball-inheritance upper bounds of Chan et al. (Theorem 2), we see that this essentially matches their first tradeoff and is tight for any $S = \Omega(n \lg^{1+\varepsilon} \lg n)$ where $\varepsilon > 0$ is an arbitrarily small constant. Most importantly, it implies that for constant query time, one needs space $n \lg^{\varepsilon} n$ words. Thus any range reporting data structure based on the ball-inheritance problem cannot improve over the bounds of Chan et al. in the regime of space $S = \Omega(n \lg^{1+\varepsilon} \lg n)$ words. We believe this holds true for any data structure that is based on *decompressing* one point at a time from some subproblem in rank space. Since decompressing from a subproblem in rank space is hard to formalize exactly, we leave it at this.

One can view our lower bound in two ways: Either as a strong indicator that the data structure of Chan et al. is optimal, or as a suggestion for how to find better upper bounds. The lower bound above shows that if we want to develop faster data structures, we have to find a technique that in some sense allows us to decompress $\omega(1)$ points in one batch, faster than decompressing each point in turn. This is not necessarily impossible given the large success of batched evaluations in other problems such as matrix multiplication and multipoint evaluation of polynomials.

We also want to make a remark regarding the gap between the second tradeoff of Chan et al. and our lower bound. We conjecture that the upper bound of Chan et al. is tight, but note that current lower bound techniques (in the cell probe model) are incapable of proving any lower bounds exceeding the one we obtain in Theorem 3: The ball-inheritance problem has only $n \lg n$ queries and the strongest lower bound for any data structure problem with m queries (for any m) is $t = \Omega(\lg(m/n)/\lg(S/n))$ [10], thus apart from our $\lg\lg\lg n$ "dirt factor", our lower bound is as strong as it possibly can be with current techniques. Note that previous papers have stated their lower bounds with a $\lg m$ rather than $\lg(m/n)$ since the data structure problems considered there have m polynomially larger than n. It is however not too difficult to see that previous techniques really are limited to $\lg(m/n)$. For the cell sampling technique [10], one needs to have $n/\operatorname{poly}(\lg n)$ queries surviving the cell sampling, hence queries have to survive the sampling with probability at least $(n/(m \operatorname{poly}(\lg n)))$, thus requiring an m/n term. For the communication game by Pătraşcu and Thorup [13], the player holding queries has $n/\operatorname{poly}(\lg n)$ queries, thus she can specify her set of queries with $\lg \binom{m}{n/\operatorname{poly}(\lg n)}$ bits of communication. This again results in a $\lg(m/n)$ term rather than $\lg m$.

Technical Contributions. As a side remark, we believe our lower bound proof has interest from a purely technical point of view. In the lower bound proof, we carefully exploit that a data structure is *not* non-deterministic. While this might sound odd at first, Wang and Yin [15] recently showed that, with only few exceptions (e.g. the predecessor lower bounds), all previous lower bound techniques yield lower bounds that hold non-deterministically. Thus having a new proof outside this category is an important contribution and may hopefully help in closing fundamental problems where avoiding non-determinism in proofs is crucial. This is e.g. the case for the deterministic dictionaries problem, which is amongst the most fundamental open problems in the field of data structures. This problem is trivially solved with constant update time and query time non-deterministically (just maintain a sorted linked list) and hence lower bound proofs must use ideas similar to those we present in this paper to prove super constant lower bounds for this important problem.

2 Lower Bound Proof

We prove our lower bound in the cell probe model [16], where the complexity of a data structure is the number of cells it reads/probes. More specifically, a data structure with query time t and space S consists of memory of S cells with consecutive integer addresses $0, \ldots, S-1$. Each cell stores w bits and we assume $w = \Omega(\lg n)$. When answering a query, the data structure may probe up to t cells and must announce the answer to the query solely based on the contents of the probed cells. The cell to probe in each step may depend arbitrarily on the query and the contents of previously probed cells. Thus computation is free of charge in the cell probe model and lower bounds proved in this model clearly applies to word-RAM data structures.

2.1 Main Ideas

In the following, we sketch the overall approach in our proof. Assume we have a data structure for the ball-inheritance problem, having space S cells of w bits and with query time t. Assume furthermore that the data structure performs very poorly in the following sense: For every input I to the ball-inheritance problem and every leaf index $b \in [n] = \{0, \ldots, n-1\}$, let $Q(b, I)$ be the set of queries that have b as its answer. Each such query probes at most t

cells of the data structure on input I. Assume these sets of cells are *disjoint*, i.e. information about the leaf b is stored in $|Q(b, I)| = \lg n$ disjoint t-sized locations in the memory.

Now pick a uniform random set C of $\lg(n!)/(4w)$ memory cells. For a query q, we say that q survives if all its t probes lie in C. Then by the disjointness of the probed cells, there will be a surviving query in $Q(b, I)$ with probability roughly $1 - (1 - (|C|/S)^t)^{\lg n}$. If $t = o(\lg \lg n / \lg(S/|C|))$, this is about $1 - \exp(\lg n \cdot (|C|/S)^t) = 1 - \exp(\lg^{1-o(1)} n)$, i.e. each leaf index is almost certainly the answer to a surviving query. Thus C must basically store the entire input. But $|C|$ is too small for this and we get a contradiction, i.e. $t = \Omega(\lg \lg n / \lg(Sw/(n \lg n)))$, which roughly equals the lower bound we claim. There are obviously a few more details to it, but this is the main idea.

Of course any realistic attempt at designing a data structure for the ball-inheritance problem would try to make the queries in $Q(b, I)$ probe the same cells (which is exactly what Chan et al.'s solution does [5]). In our actual proof, we get around this using the following observation: Consider two queries q_1, q_2 to the ball-inheritance problem, where q_2 is asked in a node d levels below the node of q_1. The probability q_1 and q_2 return the same leaf index is exponentially decreasing in d for a uniform random input. In particular this means that for the very first probe, the queries in $Q(b, I)$ will almost certainly read different cells, which is precisely the property we exploited above. If we pick a random sample of cells, there will be many queries in $Q(b, I)$ that have their first probe in the sample. To handle the remaining $t - 1$ probes, we follow [12] and extend the cell probe model with the concepts *published bits* and *accepted queries*. A data structure is allowed to publish bits at preprocessing time that the query algorithm may read free of charge. After inspecting a given query and the published bits, a data structure can choose to reject the query and not return an answer. Otherwise, the query is accepted and the algorithm must output the correct answer. Note that it is only allowed to reject queries before performing any probes.

The crucial idea is now the following: If the data structure has few published bits, then for most leaves $b \in [n]$, the published bits simply contains too little information to make the queries in $Q(b, I)$ probe the same cells. Thus for t rounds, we can pick a random sample of cells and *publish* their contents. For every accepted query, we check if its first probe is amongst the published cells. If so, we continue to accept it and may skip the first probe since we know the contents of the requested cell. Otherwise we simply reject it. If the published cell sets are small enough, there continues to be too little information in the published bits to make the queries in $Q(b, I)$ meet. Since this holds for all t probes, the argument above for the poorly performing data structures carry through and we get our lower bound.

2.2 Deriving the Lower Bound

With the ideas from Section 2.1 in mind, we present our technical lemma that allows us to publish bits for t rounds to eliminate probes while ensuring that most leaves are still the answer to many accepted queries. Before we present the lemma, consider partitioning the ball-inheritance tree into into $\lg n/Y$ disjoint layers of Y consecutive tree levels and group the accepted queries by these layers. Think of Y as looking at the queries at a given zoom level. To measure how much information we have left about the different leaves, we count for each leaf $b \in [n]$ how many layers that have at least one accepted query with b as its answer. If this count is large, then intuitively the answers to all accepted queries carry much information.

Formally, given a data structure for the ball-inheritance problem, define for every $1 \leq Y \leq \lg n$ and index $i \in [\lg n/Y]$ the *query-support set* of a leaf $b \in [n]$ on an input I as the set $Q_i^Y(b, I)$ of accepted queries in the tree levels $\{iY, \ldots, (i+1)Y - 1\}$ that has b as its

answer. Observe that $|Q_i^Y(b,I)| \in \{0,\ldots,Y\}$ since there is precisely one query in each tree level that has b as its answer (it may be less than Y since some queries might be rejected). Define also the *Y-level-support* of an input I, denoted $L^Y(I)$, as the the number of pairs (b,i) such that $Q_i^Y(b,I)$ is non-empty.

With this notation in hand we are ready to state our main Probe Elimination Lemma.

▶ **Lemma 4.** *Let $\mathcal{I}$ be a set of inputs to the ball-inheritance problem where $|\mathcal{I}| \geq n!/2^n$. Assume a ball-inheritance data structure uses S cells of w bits, answers queries in t probes, has $p < n\lg n/\lg^9 \lg n$ published bits and satisfies $L^Y(I) \geq (1-1/Z)n\lg n/Y$ for all $I \in \mathcal{I}$ for some parameters $Z \geq 2$ and $64\lg w \leq Y \leq \lg n/\alpha$, where $\alpha = (Sw\lg^{18}\lg n)/(n\lg n)$. Then there exists a subset of inputs $\mathcal{I}^* \subseteq \mathcal{I}$, with $|\mathcal{I}^*| \geq |\mathcal{I}|/2$, and another ball-inheritance data structure using S cells of w bits, answering queries in $t-1$ probes with $p + O(n\lg n/\lg^{10}\lg n)$ published bits, and satisfying $L^{\alpha Y}(I) \geq (1-1/Z-1/\lg\lg^3 n)n\lg n/(\alpha Y)$ for all $I \in \mathcal{I}^*$.*

In laymans terms, the lemma states that we can decrease the number of probes of a data structure by one, while only increasing the published bits with a lower order term. When we do this, we maintain the essential property that the leaves still have high support, just on a coarser zoom level. The Z factor is basically just a *dirt* factor. The proof of Lemma 4 can be found in the full version of the paper. In the following we use Lemma 4 to prove our main result, Theorem 3.

Assume for contradiction that a ball-inheritance data structure exists satisfying $t = o(\lg\lg_w n/\lg\alpha)$, where $\alpha = (Sw\lg^{18}\lg n)/(n\lg n)$. We proceed by repeatedly applying Lemma 4 to eliminate all t probes of the data structure. In order to guarantee we can apply Lemma 4 t times, we check the conditions for applying it. The conditions involve the number of published bits p, the parameters Z and Y and $|\mathcal{I}|$. The values of these parameters will change for each application, thus we use $p^{(i)}, Z^{(i)}, Y^{(i)}$ and $|\mathcal{I}^{(i)}|$ to denote these parameters just before the i'th invocation of the lemma. For the first round, we have $p^{(1)} = 0$ and $|\mathcal{I}^{(1)}| = n!$. Note also that $L^Y(I) = n\lg n/Y$ for any Y before the first round. Thus we choose $Y^{(1)} = 64\lg w$ to satisfy the conditions $64\lg w \leq Y^{(1)} \leq \lg n/\alpha$. This also means that we are free to choose $Z^{(1)} \geq 2$ as we wish. We simply let $Z^{(1)} = \lg^3\lg n$. Examining the lemma, we conclude that our parameters evolve in the following way (assuming we do not violate any of the conditions):

$$p^{(1+i)} = O(i(n\lg n/\lg^{10}\lg n)), \quad |\mathcal{I}^{(1+i)}| \geq n!/2^i, \quad Y^{(1+i)} = 64\lg w\cdot\alpha^i, \quad Z^{(i)} \geq \lg^3\lg n/i.$$

Since we assumed $t = o(\lg\lg_w n/\lg\alpha)$, this means that

$$p^{(1+t)} = o(n\lg n/\lg^9\lg n), \quad |\mathcal{I}^{(1+t)}| \geq n!/\lg n, \quad Y^{(1+t)} = o(\lg n), \quad Z^{(1+t)} \geq \lg^2\lg n.$$

We conclude that we can apply our lemma for t rounds under the contradictory assumption. Furthermore, the data structure we are left with answers queries in 0 probes on a subset $\mathcal{I}^* = \mathcal{I}^{(1+t)}$ of inputs, where $|\mathcal{I}^*| \geq n!/\lg n$. It has $o(n\lg n/\lg^9\lg n)$ published bits and there is some $Y^* = o(\lg n)$ such that $L^{Y^*}(I) \geq (1-1/\lg^2\lg n)n\lg n/Y^*$ for all $I \in \mathcal{I}^*$. That this is contradictory should not come as a surprise: our 0-probe data structure is capable of answering queries about almost all leaves using only the $o(n\lg n/\lg^9\lg n) \ll \lg|\mathcal{I}^*|$ published bits. The formal argument we use to reach the contradiction is as follows: we show that the 0-probe data structure can be used to uniquely encode every input $I \in \mathcal{I}^*$ into a bit string of length less than $\lg(|\mathcal{I}^*|) = \lg(n!) - \lg\lg n$ bits. This gives the contradiction since there are fewer such bit strings than inputs. We present the encoding and decoding algorithms in the following:

Encoding. Let $I \in \mathcal{I}^*$ be an input to encode. Observe that if we manage to encode the leaf index reached by each ball in the root node's list of balls, then that information completely specifies I. With this in mind, we implement the 0-probe data structure above on I and proceed as follows:

1. First we write down the published bits on input I. This cost $o(n \lg n / \lg^9 \lg n)$ bits.
2. For $i = 1, \ldots, n$ consider the i'th ball in the root node's list of balls. Let b_i denote the index of the leaf reached by that ball. We write down $\lg n/2$ bits for each such ball in turn, specifying the subtree at depth $\lg n/2$ that contains the leaf b_i. This costs $n \lg n/2$ bits.
3. Finally, we go through all leaf nodes from left to right. For a leaf b, we check if there is an accepted query returning b as its answer amongst all queries in all nodes of depth at most $\lg n/2$. If so, we continue to the next leaf. Otherwise we write $\lg n$ bits denoting the rank of the ball reaching b amongst balls the root node's list of balls. If X is the number of leaves with no accepted query reporting it in tree levels $\{0, \ldots, \lg n/2\}$, this step costs $X \lg n$ bits.

Decoding. To recover I from the above encoding, we do as follows.

1. We first go through all nodes v of depth d for $d = 0, \ldots, \lg n/2$. For each such node, let $q_1^v, \ldots, q_{n/2^d}^v$ denote the queries we can ask at node v, i.e. q_i^v asks for the leaf reached by the i'th ball in v's list of balls. We run the query algorithm for each such query in turn using the published bits written in step 1. of the encoding procedure. Since our data structure makes 0 probes, this returns the answer to each such accepted query, i.e. we have collected a set $\mathcal{Q}$ of pairs (q_i^v, b) such that b is the index of the leaf reached by the i'th ball in v's list of balls.
2. We now partition $\mathcal{Q}$ into one set $\mathcal{Q}_b$ for each leaf index b. The set $\mathcal{Q}_b$ contains all pairs $(q_i^v, b') \in \mathcal{Q}$ such that $b' = b$. There are precisely X empty such sets.
3. For each empty set $\mathcal{Q}_b$ in turn (ordered based on b), we use the bits written in step 3. of the encoding procedure to recover the rank of the ball reaching b amongst all balls in the root node's list of balls.
4. For every non-empty set $\mathcal{Q}_b$, pick an arbitrary pair $(q_i^v, b) \in \mathcal{Q}_b$. From this pair alone, we know that the ball reaching b has rank i amongst all balls ending in a leaf of the subtree rooted at v. Now initialize a counter Δ to 0. Using the bits written in step 2. of the encoding procedure, we now go through all balls in the root node's list of balls in turn. For the r'th ball, $r = 1, \ldots, n$, we check the $\lg n/2$ bits written for it and from this we determine if the ball reaches a leaf in v's subtree (possible since v can only be in the first $\lg n/2$ levels by construction). If so, we increment Δ by 1. If this causes Δ to reach i, we conclude that the ball ending in b has rank r in the root node's list of balls.
5. From the above steps, we have for every leaf b determined the rank of the ball reaching it amongst all balls in the root node's list of balls. This information completely specifies I.

Analysis. The encoding above costs

$$o(n \lg n / \lg^9 \lg n) + n \lg n/2 + X \lg n$$

bits. Now observe that if $\mathcal{Q}_b$ is empty for a leaf index b, this means $Q_i^{Y^*}(b, I)$ is empty for every $i \in \{0, \ldots, \lg n/(2Y^*) - 1\}$. This gives $L^{Y^*}(I) \leq n \lg n/Y^* - X(\lg n/(2Y^*))$. But we know $L^{Y^*}(I) \geq (1 - 1/\lg^2 \lg n) n \lg n/Y^*$ and we conclude

$$X \leq 2n/\lg^2 \lg n.$$

The encoding thus costs

$$n \lg n/2 + O(n \lg n/\lg^2 \lg n).$$

Since $\lg(n!) = n \lg n - O(n)$, we conclude that our encoding uses no more than

$$\lg(|\mathcal{I}^*|) - n \lg n/2 + O(n \lg n/\lg^2 \lg n) = \lg(|\mathcal{I}^*|) - \Omega(n \lg n)$$

bits, which completes the proof.

We have thus shown $t = \Omega(\lg \lg_w n/\lg \alpha)$ where $\alpha = (Sw \lg^{18} \lg n)/(n \lg n)$. In the word-RAM, we assume $w = \Theta(\lg n)$ and the lower bound becomes the claimed $t = \Omega(\lg \lg n/(\lg(S/n) + \lg \lg \lg n))$.

2.3 Eliminating Probes

In this section we prove Lemma 4. Recalling the intuition presented in Section 2.1, we want to show that for a data structure with few published bits, the different accepted queries reporting a fixed leaf index $b \in [n]$ have to probe distinct cells in their first probe. If we can establish this, we can pick a small random sample of memory cells and there are many of the accepted queries that make their first probe in the sample.

To formalize the above, we define a memory cell c to be k-popular on input I, if at least k accepted queries make their first probe in c on I. Define for every query-support set $Q_i^Y(b, I)$ the *cell-support set* $C_i^Y(b, I)$ as the set of memory cells that are read in the first probe of a query in $Q_i^Y(b, I)$ on input I. We measure to what extend the queries in $Q_i^Y(b, I)$ probe distinct cells using the following definitions.

▶ **Definition 5.** For an input I and value $Y \in \{1, \ldots, \lg n\}$, we say that a pair (b, i), where $b \in [n]$ and $i \in \{0, \ldots, \lg n/Y - 1\}$, is *$Y$-scattered* on input I if one of the following three holds:

1. $Q_i^Y(b, I)$ contains a query making 0 probes.
2. $C_i^Y(b, I)$ contains a w^3-popular cell.
3. $|C_i^Y(b, I)| \geq \alpha/\lg^6 \lg n$.

We define the *Y-scatter-number* of I, denoted $\Gamma^Y(I)$, as the number of pairs (b, i) that are Y-scattered on I.

If a query makes zero probes, all the information needed to answer it is contained in the already published bits. There are very few w^3-popular cells, so publishing all of them costs few bits. Most interestingly, if the queries in each support $Q_i^Y(b, I)$ set probe many distinct cells in their first probe (case 3.), then a random sample of cells will contain at least one of these cells with good probability.

We need the following lemma that captures the correspondence between large support on zoom level Y, the properties maintained by our Probe Elimination Lemma, and large scattering on a higher zoom level αY.

▶ **Lemma 6.** *Let $\mathcal{I}$ be a set of inputs to the ball-inheritance problem where $|\mathcal{I}| \geq n!/2^n$. Assume a ball-inheritance data structure uses S cells of w bits, has $p < n \lg n/\lg^9 \lg n$ published bits and satisfies $L^Y(I) \geq (1 - 1/Z)n \lg n/Y$ for all $I \in \mathcal{I}$ for some parameters $Z \geq 2$ and $64 \lg w \leq Y \leq \lg n/\alpha$, where $\alpha = (Sw \lg^{18} \lg n)/(n \lg n)$. Then there exists a subset $\mathcal{I}^* \subseteq \mathcal{I}$ of inputs such that $|\mathcal{I}^*| \geq |\mathcal{I}|/2$ and*

$$\Gamma^{\alpha Y}(I) \geq \left(1 - \frac{1}{\lg^3 \lg n}\right) \cdot \left(1 - \frac{1}{Z}\right) \cdot \frac{n \lg n}{\alpha Y}.$$

for all $I \in \mathcal{I}^*$.

We refer to the full version for the proof of Lemma 6, and use it to Prove Lemma 4 instead. Let $\mathcal{I}$ be a set of at least $n!/2^n$ inputs to the ball inheritance problem. Assume furthermore we are given a ball inheritance data structure that uses S cells of w bits, answers queries in t probes, has $p < n \lg n / \lg^9 \lg n$ published bits, and satisfies $L^Y(I) \geq (1 - 1/Z)n \lg n/Y$ for all $I \in \mathcal{I}$ for some parameters $Z \geq 2$ and $64 \lg w \leq Y \leq \lg n/\alpha$ where $\alpha = (Sw \lg^{18} \lg n)/(n \lg n)$ (as in the assumptions of Lemma 4 and Lemma 6). Let $\mathcal{I}^* \subseteq \mathcal{I}$ be the subset of $\mathcal{I}$ promised by Lemma 6. Our goal is to construct a new ball inheritance data structure answering queries in $t - 1$ probes for the inputs $\mathcal{I}^*$ while publishing few bits and keeping $L^{\alpha Y}(I)$ fairly large for all $I \in \mathcal{I}^*$. Given an input $I \in \mathcal{I}^*$, we keep the (old) p published bits and publish some additional bits from our data structure as follows:

1. First we publish all memory cells that are w^3-popular on input I. Since there are no more than $n \lg n$ accepted queries, there are no more than $n \lg n/w^3$ popular cells. The addresses and contents of all such cells can be described using $O(n \lg n/w^2) = O(n/\lg n)$ bits.
2. Next we collect all αY-scattered pairs (b, i) for input I. We remove those pairs for which $Q_i^{\alpha Y}(b, I)$ contains a query making 0 probes, or $C_i^{\alpha Y}(b, I)$ contains a w^3-popular cell. By definition, the remaining αY-scattered pairs (b, i) must satisfy $|C_i^{\alpha Y}(b, I)| \geq \alpha/\lg^6 \lg n$. We now consider all subsets of $n \lg n/(w \lg^{10} \lg n)$ memory cells and publish the subset $P^* \subseteq [S]$ for which most remaining pairs (b, i) satisfies $C_i^{\alpha Y}(b, I) \cap P^* \neq \emptyset$. Specifying the addresses and contents of cells in P^* costs $O(n \lg n/\lg^{10} \lg n)$ bits.

The query algorithm of our modified data structure is simple: We start running the old query algorithm with the p "old" published bits and stop once one of the following happens:

1. If the old query algorithm rejects the query, we also reject it.
2. If the old query algorithm answers the query without any probes, we know the answer to the query and return it.
3. Otherwise the old query algorithm makes at least one memory probe. The (address of the) first cell probed, denoted c, can be determined solely from the old published bits. Before making the actual probe, we check the newly published cells to see if c is amongst them. If so, we have the contents of c in the published bits and therefore skip the probe. We then continue executing the old query algorithm and have successfully reduced the number of probes by one. If c was not published, we simply reject the query.

Clearly our new data structure answers queries in $t-1$ probes and has $p+O(n \lg n/\lg^{10} \lg n)$ published bits. What remains is to argue that $L^{\alpha Y}(I)$ is high for all $I \in \mathcal{I}^*$ for this new data structure. To distinguish the new data structure and the old, we use $\bar{L}, \bar{Q}$ and $\bar{\Gamma}$ in place of L, Q and Γ when referring to the new data structure. L, Q and Γ refers to the old data structure.

So fix an $I \in \mathcal{I}^*$. By our choice of $\mathcal{I}^*$, we have

$$\Gamma^{\alpha Y}(I) \geq \left(1 - \frac{1}{\lg^3 \lg n}\right) \cdot \left(1 - \frac{1}{Z}\right) \cdot \frac{n \lg n}{\alpha Y}.$$

i.e. the old data structure has many pairs (b, i) that are αY-scattered on input I. By definition of $\bar{L}^{\alpha Y}(I)$, we need to lower bound the number of such pairs (b, i) that have $\bar{Q}_i^{\alpha Y}(b, I)$ non-empty, i.e. at least one query reporting b in tree-levels $\{i\alpha Y, \ldots, (i + 1)\alpha Y - 1\}$ is accepted by our new query algorithm. For this, let (b, i) be a pair that was αY-scattered for I in the old data structure. By definition of αY-scattered we know that $Q_i^{\alpha Y}(b, I)$ is non-empty. Now observe that if $Q_i^{\alpha Y}(b, I)$ contains a query that made 0 probes, then that query is also in $\bar{Q}^{\alpha Y}(b, I)$. Similarly if $Q_i^{\alpha Y}(b, I)$ contains a query making its first probe in a w^3-popular

cell, then that query is also in $\bar{Q}_i^{\alpha Y}(b,I)$ since we publish all w^3-popular cells. Hence $\bar{Q}_i^{\alpha Y}(b,I)$ can be empty only if $Q_i^{\alpha Y}(b,I)$ contains no queries making 0 probes and no queries probing a w^3-popular cell. Since (b,i) was αY-scattered, this implies $|C_i^{\alpha Y}(b,I)| \geq \alpha/\lg^6 \lg n$. Furthermore, we get that $\bar{Q}_i^{\alpha Y}(b,I)$ becomes empty only if none of these cells are in P^*.

Letting $\mu = n \lg n/(w \lg^{10} \lg n)$, we get that $C_i^{\alpha Y}(b,I)$ has a non-zero intersection with the following fraction of μ-sized cell sets:

$$1 - \frac{\binom{S-|C_i^{\alpha Y}(b,I)|}{\mu}}{\binom{S}{\mu}} \geq 1 - \frac{(S-\alpha/\lg^6 \lg n)!(S-\mu)!\mu!}{S!(S-\alpha/\lg^6 \lg n-\mu)!\mu!} \geq 1 - \frac{(S-\mu)^{\alpha/\lg^6 \lg n}}{(S-\alpha/\lg^6 \lg n)^{\alpha/\lg^6 \lg n}} =$$
$$1 - \left(\frac{S-\alpha/\lg^6 \lg n+\alpha/\lg^6 \lg n-\mu}{S-\alpha/\lg^6 \lg n}\right)^{\alpha/\lg^6 \lg n} = 1 - \left(1 - \frac{\mu-\alpha/\lg^6 \lg n}{S-\alpha/\lg^6 \lg n}\right)^{\alpha/\lg^6 \lg n}.$$

Since $\alpha = (Sw \lg^{18} \lg n)/(n \lg n) = S \lg^8 \lg n/\mu \ll \mu/2$, this is at least a

$$1 - \left(1 - \frac{\mu}{2S}\right)^{\alpha/\lg^6 \lg n} \geq 1 - \exp\left(-\alpha\mu/(2S \lg^6 \lg n)\right) \geq 1 - 1/\lg n$$

fraction. Since we chose P^* to maximize the number sets $C_i^{\alpha Y}(b,I)$ having a non-empty intersection, we conclude that at least

$$\left(1 - \frac{1}{\lg n}\right) \cdot \left(1 - \frac{1}{\lg^3 \lg n}\right) \cdot \left(1 - \frac{1}{Z}\right) \cdot \frac{n \lg n}{\alpha Y} \geq \left(1 - \frac{1}{Z} - \frac{2}{Z \lg^3 \lg n}\right) \frac{n \lg n}{\alpha Y}$$

sets $\bar{Q}_i^{\alpha Y}(b,I)$ must be non-empty. Since $Z \geq 2$, we finally conclude

$$\bar{L}^{\alpha Y}(I) \geq \left(1 - \frac{1}{Z} - \frac{1}{\lg^3 \lg n}\right) \frac{n \lg n}{\alpha Y}.$$

References

1. Stephen Alstrup, Gerth Stølting Brodal, and Theis Rauhe. New data structures for orthogonal range searching. In *Proc. 41st IEEE Symposium on Foundations of Computer Science*, pages 198–207, 2000.
2. Lars Arge, Vasilis Samoladas, and Jeffrey Scott Vitter. On two-dimensional indexability and optimal range search indexing. In *Proc. 18th ACM Symposium on Principles of Database Systems*, pages 346–357, 1999.
3. Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
4. Gerth Stølting Brodal and Kasper Green Larsen. Optimal planar orthogonal skyline counting queries. In *Proc. 14th Scandinavian Workshop on Algorithms Theory*, pages 110–121, 2014.
5. Timothy M. Chan, Kasper Larsen, and Mihai Pătraşcu. Orthogonal range searching on the ram, revisited. In *Proc. 27th ACM Symposium on Computational Geometry*, pages 354–363, 2011. See also arXiv:1011.5200.
6. Timothy M. Chan and Bryan T. Wilkinson. Adaptive and approximate orthogonal range counting. In *Proc. 24th ACM/SIAM Symposium on Discrete Algorithms*, pages 241–251, 2013.
7. Bernard Chazelle. Filtering search: a new approach to query answering. *SIAM Journal on Computing*, 15(3):703–724, 1986. **id**doi:10.1137/0215051.
8. Bernard Chazelle. Lower bounds for orthogonal range searching: I. the reporting case. *Journal of the ACM*, 37(2):200–212, 1990.

9 Johannes Fischer. Optimal succinctness for range minimum queries. In *Proc. 9th Latin American Theoretical Informatics Symposium*, pages 158–169, 2010.

10 Kasper Green Larsen. Higher cell probe lower bounds for evaluating polynomials. In *Proc. 53rd IEEE Symposium on Foundations of Computer Science*, pages 293–301, 2012.

11 Yakov Nekrich. Orthogonal range searching in linear and almost-linear space. *Computational Geometry: Theory and Applications*, 42:342–351, 2009.

12 Mihai Pătraşcu and Mikkel Thorup. Time-space trade-offs for predecessor search. In *Proc. 38th ACM Symposium on Theory of Computation*, pages 232–240, 2006.

13 Mihai Pătraşcu and Mikkel Thorup. Higher lower bounds for near-neighbor and further rich problems. *SIAM Journal on Computing*, 39(2):730–741, 2010.

14 Mihai Pătraşcu and Mikkel Thorup. Randomization does not help searching predecessors. In *Proc. 18th ACM/SIAM Symposium on Discrete Algorithms*, pages 555–564, 2007.

15 Yaoyu Wang and Yitong Yin. Certificates in data structures. In *Proc. 41st International Colloquium on Automata, Languages, and Programming*, pages 1039–1050, 2014.

16 Andrew Chi Chih Yao. Should tables be sorted? *Journal of the ACM*, 28(3):615–628, 1981.

Data Structure Lower Bounds for Document Indexing Problems*

Peyman Afshani[1] and Jesper Sindahl Nielsen[2]

1 **MADALGO, Department of Computer Science, Aarhus University, Denmark**†
peyman@madalgo.au.dk
2 **MADALGO, Department of Computer Science, Aarhus University, Denmark**‡
jasn@cs.au.dk

Abstract

We study data structure problems related to document indexing and pattern matching queries and our main contribution is to show that the pointer machine model of computation can be extremely useful in proving high and unconditional lower bounds that cannot be obtained in any other known model of computation with the current techniques. Often our lower bounds match the known space-query time trade-off curve and in fact for all the problems considered, there is a very good and reasonable match between our lower bounds and the known upper bounds, at least for some choice of input parameters.

The problems that we consider are *set intersection* queries (both the *reporting* variant and the *semi-group counting* variant), indexing a set of documents for *two-pattern* queries, or *forbidden-pattern* queries, or *queries with wild-cards*, and indexing an input set of *gapped-patterns* (or *two-patterns*) to find those matching a document given at the query time.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Data Structure Lower Bounds, Pointer Machine, Set Intersection, Pattern Matching

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.93

1 Introduction

In this paper we study a number of data structure problems related to document indexing and prove space and query time lower bounds that in most cases are almost tight. Unlike many of the previous lower bounds, we disallow random accesses by working in the pointer machine model of computation, however, we obtain high and unconditional space and query time lower bounds that almost match the best known data structures for all the problems considered; at the moment, obtaining such unconditional and tight lower bounds in other models of computation is a hopelessly difficult problem. Furthermore, compared to the previous lower bounds in the area, our lower bounds probe deeper and thus are much more informative. Consequently, these results show the usefulness of the pointer machine model.

Document indexing is an important problem in the field of information retrieval. Generally, the input is a collection of documents $\mathcal{D} = \{d_1, d_2, \ldots d_D\}$ with a total length of n characters

* A full version of this paper is available, see [5].

† Research funded by MADALGO, Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, grant DNRF84

‡ Research funded by MADALGO, Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, grant DNRF84

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 93; pp. 93:1–93:15

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

Table 1 At every row, the 3rd cell presents our space lower bound for data structures that have a query time bounded by the 2nd cell. PM stands for the pointer machine model. n is the input size and t is the output size.

Problem	Query Bound	Space Lower Bound	Assumptions
2P, FP, SI, 2FP (counting)	$Q(n)$	$\Omega(n^{2-o(1)}/Q^2(n))$	Semi-group
2P, FP, SI, 2FP (reporting)	$Q(n)+\mathcal{O}(t)$	$\Omega(n^{2-o(1)}/Q(n))$	PM
2P, FP, SI, 2FP (reporting)	$\mathcal{O}((nt)^{\frac{1}{2}-\alpha}+t)$	$\Omega\left(n^{\frac{1+6\alpha}{1+2\alpha}-o(1)}\right)$	PM, $\alpha>0$ a parameter
WCI (reporting)	$Q(n,\kappa)+\mathcal{O}(t)$	$\Omega\left(\frac{n}{\kappa}\Theta\left(\frac{\log n}{\kappa\log Q(n,\kappa)}\right)^{\kappa-1}\right)$	PM, κ wild-cards, $\kappa\leq\log_{Q(n,\kappa)} n$
WCI (reporting)	$\mathcal{O}(2^{\kappa/2}+t)$	$\Omega\left(n^{1+\Theta(1/\log k)}\right)$	PM, κ wild-cards, $3\sqrt{\log n}\leq\kappa=o(\frac{\log n}{\log\log n})$
κ-GPI	$o\left(\frac{D\gamma^\kappa}{(2+2\log D)^{\kappa+1}}\right)$	$n^{\Omega(\log^{1/(2\kappa)} n)}$	PM, $\alpha_i=0,\beta_i=\gamma$ $\kappa=o(\frac{\log\log n}{\log\log\log n})$

and usually the goal is to index them such that given a query, all the documents matching the query can be either found efficiently (the *reporting variant*) or counted efficiently (the *searching variant*). When the query is just one text pattern, the problem is classical and well-studied and there are linear space solutions with optimal query time [33]. Not surprisingly, there have been various natural extensions of this problem. We summarize the problems we study and our results below.

1.1 New and Previous Results

1.1.1 Two-pattern and the Related Queries

The *two-pattern query problem* (abbreviated as the 2P problem) was introduced in 2001 by Ferragina et al. [22] and since then it has attracted lots of attention. In the 2P problem, each query is composed of two patterns and a document matches the query if both patterns occur in the document. One can also define the *Forbidden Pattern* (FP) problem [23] where a document matches the query if it contains one pattern but not the other. For symmetry, we also introduce and consider the *Two Forbidden Patterns* (2FP) problem where none of the patterns are allowed to match the document.

Previous Results. Ferragina et al. [22] presented a number of solutions for the 2P problem with space and query times that depend on the "average size" of each document but the worst case query time and space is $\mathcal{O}(P_1+P_2+n^\alpha+t)$ and $\mathcal{O}(n^{2-\alpha}\log^{\mathcal{O}(1)} n)$, for any $0<\alpha<1$, respectively. Here P_1 and P_2 are the sizes of the query patterns and t is the output size (see also [33]). Cohen and Porat [18] offered a solution that uses $\mathcal{O}(n\log n)$ space with $\mathcal{O}(P_1+P_2+\sqrt{nt}\log^{2.5} n)$ query time. The space was improved to $\mathcal{O}(n)$ and the query time to $\mathcal{O}(P_1+P_2+\sqrt{nt}\log^{1.5} n)$ by Hon et al. [24]. The query time was reduced by a $\mathcal{O}(\sqrt{\log n})$ in [29] factor and finally the query time $\mathcal{O}(P_1+P_2+\sqrt{nt})$ was achieved in [10].

The FP problem was introduced by Fischer et al. [23] and they presented a data structure that stores $\mathcal{O}(n^{3/2})$ bits and answers queries in $\mathcal{O}(P_1+P_2+\sqrt{n}+t)$ time. Another solution was given by Hon et al. [25] that uses $\mathcal{O}(n)$ space but has $\mathcal{O}(P_1+P_2+\sqrt{nt}\log^{2.5} n)$ query time. For the searching variant (unweighted) their solution can answer queries in

$\mathcal{O}(P_1 + P_2 + \sqrt{n}\log\log n)$ time. As Larsen et al. [29] remark, the $\log\log n$ factor can be removed by using range emptiness data structures and that the same data structure can be used to count the number of matching documents for the two pattern problem.

The difficulty of obtaining fast data structures using (near) linear space has led many to believe that very efficient solutions are impossible to obtain. Larsen et al. [29] specifically focus on proving such impossibility claims and they show that the 2P and FP problems are at least as hard as Boolean Matrix Multiplication, meaning, with current techniques, $P(n) + nQ(n) = \Omega(n^{\omega/2})$ where $P(n)$ and $Q(n)$ are the preprocessing and the query times of the data structure, respectively and ω is the exponent of the best matrix multiplication algorithm (currently $\omega = 2.3728639$). If one assumes that there is no "combinatorial" matrix multiplication algorithm with better running time than $O(n^{3-o(1)})$, then the lower bound becomes $P(n) + nQ(n) = \Omega(n^{1.5-o(1)})$. Other conditional lower bounds for the 2P and FP problems but from the integer 3SUM conjecture were obtained by Kopelowitz, et al. [27, 28].

The above results are conditional. Furthermore, they tell us nothing about the complexity of the space usage, $S(n)$, versus the query time which is what we are truly interested in for data structure problems. Furthermore, even under the relatively generous assumption[1] that $P(n) = \mathcal{O}(S(n)n^{o(1)})$, the space and query lower bounds obtained from the above results have polynomial gaps compared with the current best data structures.

We need to remark that the only unconditional space lower bound is a pointer machine lower bound that shows with query time of $\mathcal{O}(\text{poly}(\log n) + k)$ the space must be $\Omega(n(\log n/\log\log n)^3)$ [23]. However this bound is very far away from the upper bounds (and also much lower than our lower bounds).

Our Results. We show that all the known data structures for 2P and FP problems are optimal within $n^{o(1)}$ factors, at least in the pointer machine model of computation: Consider a pointer machine data structure that uses $S(n)$ space and can report all the t documents that match a given 2P query (or FP query, or 2FP query) in $Q(n) + \mathcal{O}(P_1 + P_2 + t)$ time. We prove that we must have $S(n)Q(n) = \Omega\left(n^{2-o(1)}\right)$. As a corollary of our lower bound construction, we also obtain that any data structure that can answer 2P query (or FP query, or 2FP query) in $\mathcal{O}((nt)^{1/2-\varepsilon} + t)$ time, for any fixed constant $\varepsilon > 0$ must use super-linear space. As a side result, we show that surprisingly, the counting variant of the problem is in fact easier: in the semi-group model of computation (see [15] or the full version of this paper [5] for a description of the semi-group model), we prove that we must have

$$S(n)Q^2(n) = \Omega(n^2/\log^4 n)\,.$$

1.1.2 Set Intersection Queries

The interest in set intersection problems has grown considerably in recent years and variants of the set intersection problem have appeared in many different contexts. Here, we work with the following variants of the problem. The input is m sets, $S_1, \cdots, S_m$ of total size n, from a universe $\mathcal{U}$ and a query is a pair of indices i and j. The *decision variant* asks whether $S_i \cap S_j = \emptyset$. The *reporting variant* asks for all the elements in $S_i \cap S_j$. In the *counting variant*, the result should be $|S_i \cap S_j|$. In the *searching variant*, the input also includes a weight function $w : \mathcal{U} \to G$ where G is a semi-group. The query asks for $\sum_{x \in S_i \cap S_j} w(x)$.

[1] There are problems, such as jumbled indexing [13], where the preprocessing time is a polynomial factor larger than the space complexity.

Previous Results. The set intersection queries have appeared in many different formulations and variants (e.g., see [19, 37, 36, 1, 27, 18]). The most prominent conjecture is that answering the decision variant with constant query time requires $\Omega(n^{2-o(1)})$ space (see [37] for more details). For the reporting variant, Cohen and Porat [18] presented a data structure that uses linear space and answers queries in $\mathcal{O}(\sqrt{nt})$ time, where t is the output size. They also presented a linear-space data structure for the searching variant that answers queries in $\mathcal{O}(\sqrt{n})$ time. In [26] the authors study set intersection queries because of connections to dynamic graph connectivity problems. They offer very similar bounds to those offered by Cohen and Porat (with a $\sqrt{\log n}$ factor worse space and query times) but they allow updates in $\mathcal{O}(\sqrt{n \log n})$ time. It is commonly believed that all set intersection queries are hard. Explicitly stated conjectures on set intersection problems are used to obtain conditional lower bounds for problems such as distance oracles [37, 36, 19] while other well-known conjectures, such as the 3SUM conjecture, can be used to show conditional lower bounds for variants of set intersection problems [27, 1]. For other variants see [9, 34, 21].

Dietz et al. [21] considered set intersection queries in the semi-group model (a.k.a *the arithmetic model*) and they presented near optimal dynamic and offline lower bounds. They proved that given a sequence of n updates and q queries one must spend $\Omega(q + n\sqrt{q})$ time (ignoring polylog factors); in the offline version a sequence of n insertions and q queries are used but in the dynamic version, the lower bound applies to a dynamic data structure that allows insertion and deletion of points, as well as set intersection queries.

Our Results. Our lower bounds for the 2P problem easily extend to the SI problem. Perhaps the most interesting revelation here is that the searching variant is much easier than the reporting variant ($S(n)Q(n) = \Omega(n^{2-o(1)})$ for reporting versus $S(n)Q(n)^2 = \Omega(n^{2-o(1)})$ for searching)[2]. Based on this, we make another conjecture that even in the RAM model, reporting the elements in $S_i \cap S_j$ for two given query indices i and j, in $\mathcal{O}(n^{1-\varepsilon} + |S_i \cap S_j|)$ time, for any fixed constant $\varepsilon > 0$, requires $\omega(n)$ space. Such a separation between counting and reporting is a rare phenomenon with often counting being the difficult variant.

Observe that conditional lower bounds based on the Boolean Matrix Multiplication or the integer 3SUM conjectures have limitations in distinguishing between the counting and the reporting variants: For example, consider the framework of Larsen et al. [29] and for the best outcome, assume $P(n) = S(n)n^{o(1)}$ and also that boolean matrix multiplication requires $\Omega(n^{3-o(1)})$ time; then their framework yields that $S(n) + nQ(n) = \Omega(n^{3/2-o(1)})$. When $Q(n) = \Theta(n^{2/3})$ this does not rule out the possibility of having $S(n) = \mathcal{O}(n)$ (in fact the counting variant *can* be solved with linear space). However, our lower bound shows that even with $Q(n) = \Theta(n^{2/3})$ the reporting variant requires $\Omega(n^{4/3-o(1)})$ space.

1.1.3 Wild Card Indexing

We study the document indexing problem of matching with "don't cares", also known as *wild card matching* or *indexing* (WCI). The setup is the same as for the 2P problem, except a query is a single pattern but it also contains wild cards denoted by a special character "$*$". A "$*$" matches any *one* character. The task is to report all documents matched by the query. This is a well-studied problem from the upper bound perspective and there are generally two variations: either the maximum number of wild cards is bounded or it supports any number

[2] While we believe our lower bounds are certainly interesting (particularly since they separate counting and reporting variants), they do not make any progress towards resolving the status of the decision version which is considered a major open problem.

of wild cards. We consider the bounded version where patterns contain up to κ wild cards and κ is known in advance by the data structure.

Previous Results. Cole et al. [20] presented a data structure that uses $\mathcal{O}(n\frac{\log^\kappa n}{\kappa!})$ words of space and answers queries in $\mathcal{O}(P + 2^\kappa \log\log n + t)$, where t is the number of occurrences and P is the length of the query pattern. The space has been improved to $\mathcal{O}(n\log^{\kappa+\varepsilon} n)$ bits while keeping the same query time [30]. Another improvement came as a trade-off that increased query time to $\mathcal{O}(P + \beta^j \log\log n + t)$ and reduced space usage $\mathcal{O}(n\log n\log_\beta^{\kappa-1} n)$ for any $2 \le \beta \le \sigma$ where σ is the alphabet size and $j \le \kappa$ is the number of wild cards in the query [8]. In the same paper an alternate solution with $\mathcal{O}(P + t)$ query time and $\mathcal{O}(n\sigma^{\kappa^2}\log^\kappa n\log n)$ space usage was also presented. Other results have focused on reducing space while increasing the query time but their query time now depends on the alphabet size, e.g., in [31] the authors provide a data structure with $\mathcal{O}(n\log^\varepsilon n\log\sigma)$ space but with the query time of $\mathcal{O}(m + \sigma^\kappa\sqrt{\log\log\log n} + t)$.

From these bounds we note three things, first that all solutions have some exponential dependency on κ and second the alternate solution by Bille et al. [8] has an odd σ^{κ^2} factor, which is exponential on a *quadratic* function of κ as opposed to being exponential on a linear function of κ (such as 2^κ, σ^κ, or $\log^\kappa n$). Third, when the query time is forced to be independent of σ, there is a discrepancy between query and space when varying κ: Increasing κ (when it is small) by one increases the space by approximately a $\log n$ factor, while to increase the query time by a $\log n$ factor, κ needs to be increased by $\log\log n$. Based on the third point, it is quite likely that unlike SI or 2P problems, WCI does not have a simple trade-off curve.

Other results that are not directly comparable to ours include the following: One is an $\mathcal{O}(n)$ space index with $\mathcal{O}(P + \alpha)$ query time [38] where α is the number of occurrences of all the subpatterns separated by wild card characters. Note that α could be much larger than t and in fact, this can result in a worst case linear query time, even with small values of t. Nonetheless, it could perform reasonably well in practice. Two, there are lower bounds for the *partial match* problem, which is a related problem (see [35] for more details).

Our Results. For WCI with κ wild cards, we prove two results, both in the pointer machine model. In summary, we show that the exponential dependency of space complexity or query time on κ generally cannot be avoided.

As our first result and for a binary alphabet ($\sigma = 2$), we prove that for $3\sqrt{\log n} \le \kappa = o(\frac{\log n}{\log\log n})$, any data structure that answers queries in $\mathcal{O}(2^{\kappa/2} + P + t)$ time must consume $n^{1+\Theta(1/\log\kappa)}$ space. This result rules out the possibility of lowering the σ^{κ^2} factor in the alternate solution offered by Bille et al. [8], over all values of κ, to $\sigma^{\kappa^{2-\varepsilon}}$ for any constant $\varepsilon > 0$: by setting $\kappa = 3\sqrt{\log n}$ (and $\sigma = 2$), such a bound will be much smaller than our space lower bound (essentially involves comparing $2^{\mathcal{O}(\log^{1-\varepsilon/2} n)}$ factor to a $2^{\Omega(\log n/\log\log n)}$ factor). While this does not rule out improving the space bound for small values of κ, it shows that the exponential dependency on κ^2 is almost tight at least in a particular point in the range of parameters.

As our second result, we prove that answering WCI queries in $Q(n,\kappa) + \mathcal{O}(P + t)$ time requires $\Omega(\frac{n}{\kappa}\Theta\left(\frac{\log_{Q(n,\kappa)} n}{\kappa}\right)^{\kappa-1})$ space, as long as $\kappa < \log_{Q(n,\kappa)} n$. Note that this query time is assumed to be independent of σ. This result also has a number of consequences. One, it shows that any data structure with query time of $\mathcal{O}(\log^{\mathcal{O}(1)} n + P + t)$ requires $\Omega(\frac{n}{\kappa}\left(\frac{\log n}{\kappa\log\log n}\right)^{\kappa-1})$ space. Note that this is rather close to the space complexity of the

data structure of Cole et al. [20] ($2^{\mathcal{O}(\kappa)}(\log\log n)^\kappa$ factor away). In other words, the data structure of Cole et al. cannot be significantly improved both in space complexity and query time; e.g., it is impossible to answer WCI queries in $\mathcal{O}(\log^{\mathcal{O}(1)} n + P + t)$ time using $\mathcal{O}\left(n\left(\frac{\log n}{\kappa}\right)^{\kappa(1-\varepsilon)}\right)$ space, for any constant $\varepsilon > 0$.

Two, $\kappa = 2$ is the smallest value where linear space becomes impossible with polylogarithmic query time. This is very nice since $\kappa = 1$ can be solved with almost linear space [30]. Furthermore this shows that the increase by a $\log n$ factor in the space complexity for every increase of κ is necessary (for small values of κ).

Three, we can combine our second result with our first result when $\kappa = 3\sqrt{\log n}$. As discussed before, our first result rules out fast queries (e.g., when $Q(n) \leq 2^{\kappa/2}$), unless the data structure uses large amounts of space, so consider the case when $Q(n) = \mathcal{O}(2^\kappa)$. In this case, we can rule out lowering the space usage of the data structure of Cole et al. to $\Omega\left(n\left(\frac{\log n}{\kappa}\right)^{\kappa^{1-\varepsilon}}\right)$ for any constant $\varepsilon > 0$: apply our second lower bound with fewer wild cards, specifically, with $\kappa' = \kappa^{1-\delta}$ wild cards, for a small enough constant $\delta > 0$ that depends on ε. Observe that $\kappa' < \log_{Q(n)} n$, so the second result lower bounds the space by $\Omega\left(\frac{n}{\kappa'}\Theta\left(\log^{\frac{\delta}{2}\kappa^{1-\delta}} n\right)\right)$, which for a sufficiently small δ is greater than $\Omega\left(n\left(\frac{\log n}{\kappa}\right)^{\kappa^{1-\varepsilon}}\right)$.

As mentioned in the beginning, our results show that the exponential dependency of space or query time on κ cannot be improved in general. Furthermore, at a particular point in the range of parameters (when $\kappa = 3\sqrt{\log n}$), all the known exponential dependencies on κ are almost tight and cannot be lowered to an exponential dependency on $\kappa^{1-\varepsilon}$ (or on $\kappa^{2-\varepsilon}$ for the alternate solution) for any constant $\varepsilon > 0$. Nonetheless, there are still gaps between our lower bounds and the known data structures. We believe it is quite likely both our lower bounds and the existing data structures can be improved to narrow the gap.

1.1.4 Gapped Pattern Indexing

A κ-gapped pattern is a pattern $p_1\{\alpha_1, \beta_1\}p_2\{\alpha_2, \beta_2\}, \cdots, p_\kappa\{\alpha_\kappa, \beta_\kappa\}p_{\kappa+1}$ where α_i and β_i, $1 \leq i \leq \kappa$ are integers, and each p_i, $1 \leq i \leq \kappa + 1$, is a string over an alphabet of size σ. Such a κ-gapped pattern matches a documents in which one can find one occurrence of every p_i such that there are at least α_i and at most β_i characters between the occurrence of p_i and the occurrence of p_{i+1}, $1 \leq i \leq \kappa$.

Previous Results. The gapped pattern indexing is often considered both in the online and the offline version (e.g., see [7, 40]). However, the result most relevant to us is [6], where they consider the following data structure problem: given a set of 1-gapped patterns of total size n, where all the patterns are in the form of $p_1\{\alpha, \beta\}p_2$, store them in a data structure such that given a document of length D at the query time, one can find all the gapped patterns that match the query document (in general we call this the *κ-gapped pattern indexing (κ-GPI)* problem). In [6], the authors give a number of upper bounds and conditional lower bounds for the problem. Among a number of results, they can build a data structure of linear size that can answer queries in $\tilde{\mathcal{O}}(D(\beta - \alpha) + t)$ where $\tilde{\mathcal{O}}$ notation hides polylogarithmic factors and t is the output size. For the lower bound and among a number of results, they can show that with linear space $\Omega(D(\beta - \alpha)^{1-o(1)} + t)$ query time is needed.

Our Results. We consider the general κ-GPI problem where $\beta_i - \alpha_i = \gamma$ for all $1 \leq i \leq \kappa$, and a prove lower bound that is surprisingly very high: any pointer machine data structure

that can answer queries in $o(D\gamma^\kappa/(2\log D)^{\kappa+1})$ time must use *super polynomial* space of $n^{\Omega(\log^{1/(\kappa+1)} n)}$. By construction, this result also holds if the input is a set of $\kappa+1$ patterns where they all need to match the query document (regardless of their order and size of the gaps). In this case, answering queries in $o(D^{\kappa+1}/(2\log D)^{\kappa+1})$ requires the same super-polynomial space. Note that in this case $\kappa = 1$ is the "dual" of the 2P problem: store a set of two-patterns in data structure such that given a query document, we can output the subset of two-patterns that match the document.

1.2 Technical Preliminaries

The Pointer Machine Model [39]. This models data structures that solely use pointers to access memory locations (e.g., any tree-based data structure)[3]. We focus on a variant that is the popular choice when proving lower bounds [14]. Consider an abstract "reporting" problem where the input includes a universe set $\mathcal{U}$ where each query q reports a subset, $q_\mathcal{U}$, of $\mathcal{U}$. The data structure is modelled as a directed graph G with outdegree two (and a root node $r(G)$) where each vertex represents one memory cell and each memory cell can store one element of $\mathcal{U}$; edges between vertices represent pointers between the memory cells. All other information can be stored and accessed for free by the data structure. The only requirement is that given the query q, the data structure must start at $r(G)$ and explore a connected subgraph of G and find its way to vertices of G that store the elements of $q_\mathcal{U}$. The size of the subgraph explored is a lower bound on the query time and the size of G is a lower bound on the space.

An important remark. The pointer-machine can be used to prove lower bounds for data structures with query time $Q(n) + \mathcal{O}(t)$ where t is the output size and $Q(n)$ is "search overhead". Since we can simulate any RAM algorithm on a pointer-machine with $\log n$ factor slow down, we cannot hope to get high unconditional lower bounds if we assume the query time is $Q(n) + \mathcal{O}(t \log n)$, since that would automatically imply RAM lower bounds for data structures with $Q(n)/\log n + \mathcal{O}(t)$ query time, something that is hopelessly impossible with current techniques. However, when restricted to query time of $Q(n) + \mathcal{O}(t)$, the pointer-machine model is an attractive choice and it has an impressive track record of proving lower bounds that match the best known data structures up to very small factors, even when compared to RAM data structures; we mention two prominent examples here. For the fundamental *simplex range reporting* problem, all known solutions are pointer-machine data structures [32, 11, 17] and the known pointer machine lower bounds match these up to an $n^{o(1)}$ factor [2, 16]. One can argue that it is difficult to use the power of RAM for simplex range reporting problem. However, for the other fundamental *orthogonal range reporting*, where it is easy to do various RAM tricks, the best RAM data structures save at most a $\log n$ factor compared to the best known pointer machine solutions (e.g., see [3, 4, 12]). Also, where cell-probe lower bounds cannot break the $\log n$ query barrier, very high lower bounds are known for the orthogonal range reporting problem in the pointer machine model [3, 4, 14].

Known Frameworks. The fundamental limitation in the pointer machine model is that starting from a memory cell v, one can visit at most 2^ℓ other memory cells using ℓ pointer

[3] Many of the known solutions for various indexing problems use tree structures, such as suffix trees or wavelet trees. While sometimes trees are can be encoded using bit-vectors with rank/select structures on top, these tricks can only save polylogarithmic factors in space and query times.

navigations. There are two known methods that exploit this limitation and build two different frameworks for proving lower bounds.

The first lower bound framework was given by Bernard Chazelle [14, 16]. However, we will need a slightly improved version of his framework that is presented in the following lemma; essentially, we need a slightly tighter analysis on a parameter that was originally intended as a large constant. Due to lack of space, the proof is only available in the full version [5].

▶ **Theorem 1.** *Let $\mathcal{U}$ be a set of n input elements and $\mathcal{Q}$ a set of queries where each query outputs a subset of $\mathcal{U}$. Assume there exists a data structure that uses $S(n)$ space and answers each query in $Q(n) + \alpha k$ time, where k is the output size. Assume (i) the output size of any query $q \in \mathcal{Q}$, denoted by $|\mathcal{U} \cap q|$, is at least t, for a parameter $t \geq Q(n)$ and (ii) for integers ℓ and β, and indices, $i_1, \cdots, i_\ell$, $|\mathcal{U} \cap q_{i_1} \cap \cdots \cap q_{i_\ell}| < \beta$. Then, $S(n) = \Omega\left(\frac{|\mathcal{Q}|t}{\ell \cdot 2^{O(\alpha\beta)}}\right)$.*

The second framework is due to Afshani [2] and it is designed for "geometric stabbing problems": given an input set of n geometric regions, the goal is store them in a data structure such that given a query point q, one can output the subset of regions that contain q. The framework is summarized below.

▶ **Theorem 2** ([2])**.** *Assume one can construct a set of n geometric regions inside the d-dimensional unit cube such that (i) every point of the unit cube is contained in at least t regions [4], and (ii) the volume of the intersection of every β regions is at most v, for some parameters β, t, and v. Then, for any pointer-machine data structure that uses $S(n)$ space and can answer geometric stabbing queries on the above input in time $g(n)+O(k)$, where k is the output size and $g(\cdot)$ is some increasing function, if $g(n) \leq t$ then $S(n) = \Omega(tv^{-1}2^{-O(\beta)})$.*

These two frameworks are not easily comparable. In fact, for many constructions, often only one of them gives a non-trivial lower bound. Furthermore, as remarked by Afshani [2], Theorem 2 does not need to be operated in the d-dimensional unit cube and in fact any measure could be substituted instead of the d-dimensional Lebesgue measure.

Our Techniques. Our first technical contribution is to use Theorem 2 in a non-geometric setting by representing queries as abstract points under a discrete measure and each input object as a range that contains all the matching query points. Our lower bound for the κ-GPI problem and one of our WCI lower bounds are proved in this way. The second technical contribution is actually building and analyzing proper input and query sets to be used in the lower bound frameworks. In general, this is not easy and in fact for some problems it is highly challenging[5]. Also see Section 5 (Conclusions) for a list of open problems.

In the rest of this article, we present the technical details behind our κ-GPI lower bound and most of the details of our first WCI lower bound. Due to lack of space, the rest of the technical details can be found in the full version [5].

We begin with the κ-GPI problem since it turns out for this particular problem we can get away with a simple deterministic construction. For WCI, we need a more complicated randomized construction to get the best result and thus it is presented next.

---

[4] In [2] this is stated as "exactly t ranges" but the proof works with only a lower bound on t.

[5] A good example is the classical halfspace range reporting problem where constructing proper input and query sets has been a longstanding open problem; the current best lower bound uses a highly inefficient reduction from the simple range reporting problem [2].

2 Gapped Pattern Lower Bound

In this section we deal with the data structure version of the gapped pattern problem. The input is a collection of κ-gapped patterns (typically called a dictionary), with total length n (in characters). The goal is to store the input in a data structure such that given a document of size D, one can report all the input gapped patterns that match the query. We focus on special κ-gapped patterns that we call *standard*: a standard κ-gapped pattern in the form of $p_1 \{0,\gamma\} p_2 \{0,\gamma\} \dots \{0,\gamma\} p_{\kappa+1}$ where each p_i is a string (which we call a *subpattern*) and γ is an integer.

▶ **Theorem 3.** *For $\kappa = o(\frac{\log\log n}{\log\log\log n})$ and in the pointer machine model, answering κ-GPI queries in $o\left(\frac{D\gamma^\kappa}{(2+2\log D)^{\kappa+1}}\right) + O(t)$ time requires $n^{\Omega(\log^{1/(2\kappa)} n)}$ space.*

To prove this theorem, we build a particular input set of standard κ-gapped patterns. We pick the alphabet $\Sigma = \{0, 1, \#\}$, and the gapped patterns only use $\{0, 1\}$. Each subpattern in the input is a binary string of length p. The subpatterns in any gapped pattern are in lexicographic order, and a subpattern occurs at most once in a pattern (i.e., no two subpatterns in a pattern are identical). The input set, S, contains all possible gapped patterns obeying these conditions. Thus, $|S| = \binom{2^p}{\kappa+1}$. Each query is a text composed of concatenation of $D/(p+1)$ substrings (for simplicity, we assume $D/(p+1)$ is an integer) and each substring is in the form '$\#\{0,1\}^p$'. We restrict the query substrings to be in lexicographic order and without repetitions (no two substrings in a query are identical). The set of all query texts satisfying these constraints is denoted by Q. Thus, $|Q| = \binom{2^p}{D/(p+1)}$.

▶ **Lemma 4.** *For $D \geq 2\kappa\gamma$, any text $T \in Q$ matches $\Theta\left(\frac{D\gamma^\kappa}{(p+1)^{\kappa+1}}\right)$ gapped patterns in S.*

Proof. Consider a text $T \in Q$. To count the number of gapped patterns that can match it, we count the different ways of selecting $\kappa+1$ positions that correspond to the starting positions of a matching subpattern. Each position starts immediately after '#', with at most γ characters between consecutive positions. Since $D \geq 2\kappa\gamma$, we have $\Theta(D/p)$ choices for picking the first position, i.e., the starting position of a gapped pattern matching T. After fixing the first match, there are at most $\gamma/(p+1)$ choices for the position of the next match between a subpattern and substring. However, if the first match happens in the first half of text T, there are always $\gamma/(p+1)$ choices for the position of each subpattern match (since $D \geq 2\kappa\gamma$). Thus, we have $\Theta\left(D\gamma^k/(p+1)^{\kappa+1}\right)$ choices. As input subpatterns are in lexicographically increasing order, different choices result in distinct gapped patterns that match the query. ◀

To apply Theorem 2, we consider each query text T in Q as a "discrete" point with measure $1/|Q|$. Thus, the total measure of Q (i.e., the query space) is one and Q functions as the "unit cube" within the framework of Theorem 2. We consider an input gapped pattern P in S as a range that contains all the points of Q that match P. Thus, to apply Theorem 2, we need to find a lower bound on the output size of every query (condition (i)) and an upper bound on v, the measure of the intersection of β inputs (condition (ii)). By the above lemma, meeting the first condition is quite easy: we pick $t = \Theta\left(\frac{D\gamma^\kappa}{(p+1)^{\kappa+1}}\right)$ (with the right hidden constant). Later we shall see that $p = 1 + 2\log D$ so this can be written as $t = \Theta\left(\frac{D\gamma^\kappa}{(2+2\log D)^{\kappa+1}}\right)$ Thus, we only need to upper bound v which we do below.

▶ **Lemma 5.** *Consider β patterns $P_1, \cdots, P_\beta \in S$. At most $\binom{2^p - \beta^{1/(\kappa+1)}}{D/(p+1) - \beta^{1/(\kappa+1)}}$ texts in Q can match all patterns $P_1, \cdots, P_\beta$.*

Proof. Collectively, $P_1, \cdots, P_\beta$ must contain at least $r = \beta^{1/(\kappa+1)}$ distinct subpatterns: otherwise, we can form at most $\binom{r}{\kappa+1} < \beta$ different gapped patterns, a contradiction. This in turn implies that any text T matching $P_1, \cdots, P_\beta$ must contain all these at least r distinct subpatterns. Clearly, the number of such texts is at most $\binom{2^p - \beta^{1/(\kappa+1)}}{D/(p+1) - \beta^{1/(\kappa+1)}}$. ◀

As the measure of each query in Q is $1/|Q|$, by the above theorem, we have $v \leq \binom{2^p - \beta^{1/(\kappa+1)}}{D/(p+1) - \beta^{1/(\kappa+1)}}/|Q|$. We now can apply Theorem 2. Each pattern in the input has $\Theta(p\kappa)$ characters and thus the total input size, n, is $\Theta(p\kappa|S|) = \Theta(p\kappa\binom{2^p}{\kappa+1})$. By the framework, and Lemmata 5 and 4, we know that the space usage of the data structure is at least

$$\Omega\left(\frac{D\gamma^\kappa}{(p+1)^{\kappa+1}} \cdot \frac{\binom{2^p}{D/(p+1)}}{\binom{2^p - \beta^{1/(\kappa+1)}}{D/(p+1) - \beta^{1/(\kappa+1)}}} \cdot 2^{-\mathcal{O}(\beta)}\right) = \Omega\left(1 \cdot \left(\frac{(p+1)2^{(p-1)}}{D}\right)^{\beta^{1/(\kappa+1)}} \cdot 2^{-\mathcal{O}(\beta)}\right)$$

where to obtain the rightmost equation we expand the binomials, simplify, and constrain $\beta^{1/(\kappa+1)} < 2^p/2$ to lower bound $2^p - \beta^{1/(\kappa+1)}$ with 2^{p-1}. Now we work out the parameters. We know that $n = \Theta(p\kappa\binom{2^p}{\kappa+1}) = 2^{\Theta(p(\kappa+1))}$; this is satisfied by setting $p = c_p(\log n)/(\kappa+1)$ for some constant c_p. Observe that there is an implicit constraint on D and p: there should be sufficient bits in the subpatterns to fill out a query with distinct subpatterns, i.e. $2^p > D/(p+1)$; we pick $D = n^{c_D 1/(\kappa+1)}$ for some other constant c_D such that $D = 2^{p/2-1}$ and thus $(p+1)2^{p-1}/D = 2^{p/2}$. Using these values, the space lower bound is simplified to

$$\Omega\left(2^{\beta^{1/(\kappa+1)}\frac{c_p}{2}\frac{\log n}{\kappa+1}} \cdot 2^{-c_\beta\beta}\right)$$

where c_β is another constant. We now optimize the lower bound by picking β such that $c_\beta\beta = \frac{1}{2}\beta^{1/(\kappa+1)}\frac{c_p}{2}\frac{\log n}{\kappa+1}$ which solves to $\beta = \Theta((\frac{\log n}{\kappa+1})^{1+1/\kappa})$. Thus, for constant c, the space complexity of the data structure is

$$\Omega\left(2^{c\left(\frac{\log n}{\kappa+1}\right)^{1+\frac{1}{\kappa}}}\right) = \Omega\left(2^{\log n \cdot c \log^{\frac{1}{\kappa}} n\left(\frac{1}{\kappa+1}\right)^{1+\frac{1}{\kappa}}}\right) = n^{\Omega\left(\log^{\frac{1}{\kappa}} n\left(\frac{1}{\kappa+1}\right)^{1+\frac{1}{\kappa}}\right)} = n^{\Omega\left(\log^{\frac{1}{2\kappa}} n\right)}$$

where the last part follows from $\kappa = o(\log\log n/\log\log\log n)$.

3 Wild Card Indexing

In this section we consider the wild card indexing (WCI) problem and prove both space and query lower bounds in the pointer machine model of computation. Note that our query lower bound applies to an alphabet size of two (i.e., binary strings).

3.1 The Query Lower Bound

Assume for any input set of documents of total size n, we can build a data structure such that given a WCI query of length m containing κ wild cards, we can find all the documents that match the query in $Q(n,\kappa) + O(m+t)$ time, where t is the output size. Furthermore, assume κ is a fixed parameter known by the data structure and that the data structure consumes $S(n,\kappa)$ space. Our main result here is the following.

▶ **Theorem 6.** *If* $3\sqrt{\log n} \leq \kappa = o(\log n)$ *and* $Q(n,\kappa) = O(2^{\frac{\kappa}{2}})$, *then* $S(n,\kappa) \geq n^{1+\Theta(\frac{1}{\log\kappa})}$.

To prove the lower bound, we build a *particular* set of documents and patterns and prove that if the data structure can answer the queries fast, then it must consume lots of space, for this particular input, meaning, we get lower bounds for the function $S(\cdot)$. We now present the details. We assume $Q(n,\kappa) \leq 2^{\kappa/2}$, as otherwise the theorem is trivial.

Documents and patterns. We build the set of documents in two stages. Consider the set of all bit strings of length m with exactly $\ell = \kappa/2$ "1"s. In the first stage, we sample each such string independently with probability r^{-1} where $r = 2^{\kappa/3}$. Let $\mathcal{D}$ be the set of sampled strings. In the second stage, for every set of $\ell + \ell'$ indices, $1 \leq i_1 < i_2 < \cdots < i_{\ell+\ell'} \leq m$, where $\ell' = (\log_\ell r)/2 = \Theta(\kappa/\log\kappa)$, we perform the following operation, given another parameter β: if there are more than β strings in $\mathcal{D}$ that have "1"s only among positions $i_1, \cdots, i_{\ell+\ell'}$, then we remove all such strings from $\mathcal{D}$. Consequently, among the remaining strings in $\mathcal{D}$, "1"s in every subset of β strings will be spread over at least $\ell + \ell' + 1$ positions. The set of remaining strings $\mathcal{D}$ will form our input set of documents. Now we consider the set $\mathcal{P}$ of all the patterns of length m that have exactly κ wild cards and $m - \kappa$ "0"s. We remove from $\mathcal{P}$ any pattern that matches fewer than $\binom{\kappa}{\ell}/(2r)$ documents from $\mathcal{D}$. The remaining patterns in $\mathcal{P}$ will form our query set of patterns. In the full version [5], we prove the following.

▶ **Lemma 7.** *With positive probability, we get a set $\mathcal{D}$ of $\Theta(\binom{m}{\ell}/r)$ documents and a set $\mathcal{P}$ of $\Theta(\binom{m}{\kappa})$ patterns such that (i) each pattern matches $\Theta(\binom{\kappa}{\ell}/r)$ documents, and (ii) there are no $\beta = \Theta(\log_\kappa m)$ documents whose "1"s are contained in a set of $\ell + \ell'$ indices.*

To prove the lower bound, we use Theorem 2. We use a discrete measure here: each pattern is modelled as a "discrete" point with measure $\frac{1}{|\mathcal{P}|}$, meaning, the space of all patterns has measure one. Each document forms a range: a document d_i contains all the patterns (discrete points) that match d_i. Thus, the measure of every document $d_i \in \mathcal{D}$ is $t_i/|\mathcal{P}|$, where t_i is the number of patterns that match d_i. We consider the measure of the intersection of β documents (regions) $d_1, \ldots, d_\beta$. By Lemma 7, there are $\ell + \ell'$ indices where one of these documents has a "1"; any pattern that matches all of these documents must have a wild card in all of those positions. This means, there are at most $\binom{m-\ell-\ell'}{\kappa-\ell-\ell'}$ patterns that could match documents $d_1, \ldots, d_\beta$. This means, when we consider documents as ranges, the intersection of every β documents has measure at most $\binom{m-\ell-\ell'}{\kappa-\ell-\ell'}/|\mathcal{P}|$ which is an upper bound for parameter v in Theorem 2. For the two other parameters t and $g(n)$ in the theorem we have, $t = \Theta(\binom{\kappa}{\ell}/r)$ (by Lemma 7) and $g(n) = Q(n,\kappa) + O(m)$. To obtain a space lower bound from Theorem 2, we must check if $g(n) \leq t = \Theta(\binom{\kappa}{\ell}/r)$. Observe that $\binom{\kappa}{\ell} = \binom{\kappa}{\kappa/2} \geq 2^\kappa/\kappa$ since the binomials $\binom{\kappa}{i}$ sum to 2^κ for $0 \leq i \leq \kappa$ and $\binom{\kappa}{\kappa/2}$ is the largest one. As $r = 2^{-\kappa/3}$, we have $t = \Omega(2^\kappa 2^{-\kappa/3}/\kappa) = \omega(2^{\kappa/2}) = \omega(Q(n,k))$. However, $g(n)$ also involves an additive $O(m)$ term. Thus, we must also have $t = \omega(m)$ which will hold for our choice of parameters but we will verify it later.

By guaranteeing that $g(n) \leq t$, Theorem 2 gives a space lower bound of $\Omega(tv^{-1}2^{-O(\beta)})$. However, we would like to create an input of size $\Theta(n)$ which means the number of sampled documents must be $\Theta(n/m)$ and thus we must have $\binom{m}{\ell}/r = \Theta(n/m)$. As $m = \omega(\kappa)$, it follows that $\binom{m}{\ell} = (\Theta(1)m/\ell)^\ell$. Thus, $(\Theta(1)m/\ell)^\ell = \Theta(rn/m)$. Thus, we have that

$$v^{-1} \geq \frac{|P|}{\binom{m-\ell-\ell'}{\kappa-\ell-\ell'}} \geq \frac{\Theta\left(\binom{m}{\kappa}\right)}{\binom{m-\ell-\ell'}{\kappa-\ell-\ell'}} = \Theta(1)\cdot\frac{\frac{m!}{\kappa!(m-\kappa)!}}{\frac{(m-\ell-\ell')!}{(\kappa-\ell-\ell')!(m-\kappa)!}} \geq \Theta(1)\cdot\frac{m^{\ell+\ell'}}{(2\kappa)^{\ell+\ell'}} = \frac{\Theta(1)\cdot n}{m\Theta(1)^\kappa}\cdot\left(\frac{m}{2\kappa}\right)^{\ell'}$$

where the last step follows from $(\Theta(1)m/\ell)^\ell = \Theta(rn/m)$, $r = 2^{\kappa/3}$ and $\ell = \kappa/2$.

Now we bound m in terms of n. From $(m/(2\kappa))^\ell = \frac{n}{mc^\kappa}$ we obtain that $m = \kappa^{\ell/(\ell+1)} \cdot n^{1/(\ell+1)}/\Theta(1)^\kappa = n^{2/(\kappa+2)}/\Theta(1)^\kappa$. Remember that $\ell' = \Theta(\kappa/\log\kappa)$. Based on this, we get that $v^{-1} = n \cdot n^{\Theta(1/\log\kappa)}/\Theta(1)^\kappa$ and since $\kappa = o(\log n/\log\log n)$ the $\Theta(1)^\kappa$ term is dominated and we have $v^{-1} = n \cdot n^{\Theta(1/\log\kappa)}$. It remains to handle the extra $2^{-O(\beta)}$ factor in the space lower bound. From Lemma 7, we know that $\beta = c\log_\kappa m$. Based on the value of m, this means $\beta = \Theta(\log n/(\kappa\log\kappa))$ which means $2^{-O(\beta)}$ is also absorbed in $n^{\Theta(1/\log\kappa)}$

factor. It remains to verify one last thing: previously, we claimed that we would verify that $t = \omega(m)$. Using the bound $t = \omega(2^{\kappa/2})$ this can be written as $2^{\kappa/2} = \omega(m)$ which translates to $\kappa/2 = (2\log n)/(\kappa+2) + \omega(1)$ which clearly holds if $\kappa \geq 3\sqrt{\log n}$.

3.2 The Space Lower Bound

The details of our space lower can be found in the full version [5], where we prove the following.

▶ **Theorem 8.** *labelthm:wcis Any pointer-machine data structure that answers WCI queries with κ wild cards in time $Q(n) + O(m + t)$ over an input of size n must use $\Omega\left(\frac{n}{\kappa}\Theta\left(\frac{\log_{Q(n)} n}{\kappa}\right)^{\kappa-1}\right)$ space, as long as $\kappa < \log_{Q(n)} n$, where t is the output size, and m is the pattern length.*

Refer to the introduction for a discussion of the consequences of these lower bounds.

4 Two Pattern Document Indexing and Related Problems

Due to lack of space we only state the primary results for 2P, FP, 2FP, and SI. The proofs for Theorems 9 and 10 can be found in the full version [5].

▶ **Theorem 9.** *Any data structure on the Pointer Machine for the 2P, FP, 2FP, and SI problems with query time $Q(n)$ and space usage $S(n)$ must obey $S(n)Q(n) = \Omega\left(n^{2-o(1)}\right)$.*

Also, if $Q(n,k) = O((nk)^{1/2-\alpha} + k)$ for a constant $0 < \alpha < 1/2$, then $S(n) = \Omega\left(n^{\frac{1+6\alpha}{1+2\alpha}-o(1)}\right)$.

The above theorem is proved using Theorem 1 which necessitates a randomized construction involving various high probability bounds. Unlike our lower bound for κ-GPI we were unable to find a deterministic construction that uses Theorem 2.

We also prove the following lower bound in the semi-group model which addresses the difficulty of the counting variants of 2P and the related problems.

▶ **Theorem 10.** *Answering 2P, FP, 2FP, and SI queries in the semi-group model requires $S(n)Q^2(n) = \Omega(n^2/\log^4 n)$.*

5 Conclusions

In this paper we proved unconditional and high space and query lower bounds for a number of problems in string indexing. Our main message is that the pointer machine model remains an extremely useful tool for proving lower bounds, that are close to the *true complexity* of many problems. We have successfully demonstrated this fact in the area of string and document indexing. Within the landscape of lower bound techniques, the pointer machine model, fortunately or unfortunately, is the only model where we can achieve *unconditional, versatile, and high* lower bounds and we believe more problems from the area of string and document indexing deserve to be considered in this model. To this end, we outline a number of open problems connected to our results.

1. Is it possible to generalize the lower bound for 2P to the case where the two patterns are required to match within distance γ? This is essentially a "dual" of the 1-GPI problem.
2. Recall that our space lower bound for the WCI problem (Subsection 3.2) assumes that the query time is independent of the alphabet size σ. What if the query is allowed to increase with σ?

3. Our query lower bound for the WCI (Subsection 3.1) is proved for a binary alphabet. Is it possible to prove lower bounds that take σ into account? Intuitively the problem should become more difficult as σ increases, but we were unable to obtain such bounds.
4. We require certain bounds on κ for the WCI problem. Is it possible to remove or at least loosen them? Or perhaps, can the upper bounds be substantially improved?
5. What is the difficulty of the κ-GPI problem when κ is large?

Acknowledgements. We thank Karl Bringmann for simplifying the construction of our documents for the 2P problem. We thank Moshe Lewenstein for bringing the WCI problem to our attention. We also thank the anonymous referees for pointing us to [21].

References

1 Amir Abboud and Virginia Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 434–443, 2014.

2 Peyman Afshani. Improved pointer machine and I/O lower bounds for simplex range reporting and related problems. In *Symposium on Computational Geometry (SoCG)*, pages 339–346, 2012.

3 Peyman Afshani, Lars Arge, and Kasper Dalgaard Larsen. Orthogonal range reporting: query lower bounds, optimal structures in 3-d, and higher-dimensional improvements. In *Symposium on Computational Geometry (SoCG)*, pages 240–246, 2010.

4 Peyman Afshani, Lars Arge, and Kasper Green Larsen. Higher-dimensional orthogonal range reporting and rectangle stabbing in the pointer machine model. In *Symposium on Computational Geometry (SoCG)*, pages 323–332, 2012. doi:10.1145/2261250.2261299.

5 Peyman Afshani and Jesper Sindahl Nielsen. Data structure lower bounds for document indexing problems. *CoRR*, abs/1604.06264, 2016. URL: http://arxiv.org/abs/1604.06264.

6 Amihood Amir, Tsvi Kopelowitz, Avivit Levy, Seth Pettie, Ely Porat, and B. Riva Shalom. Online Dictionary Matching with One Gap. *CoRR*, abs/1503.07563, 2015. URL: http://arxiv.org/abs/1503.07563.

7 Amihood Amir, Avivit Levy, Ely Porat, and B. Riva Shalom. Dictionary Matching with One Gap. In *Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 11–20, 2014.

8 Philip Bille, Inge Li Gørtz, Hjalte Wedel Vildhøj, and Søren Vind. String indexing for patterns with wildcards. In *Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 283–294, 2012.

9 Philip Bille, Anna Pagh, and Rasmus Pagh. Fast evaluation of union-intersection expressions. In *Algorithms and Computation*, volume 4835 of *Lecture Notes in Computer Science*, pages 739–750. Springer Berlin Heidelberg, 2007.

10 Sudip Biswas, Arnab Ganguly, Rahul Shah, and Sharma V. Thankachan. Ranked Document Retrieval with Forbidden Pattern. In *Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 77–88, 2015.

11 Timothy M. Chan. Optimal partition trees. In *Symposium on Computational Geometry (SoCG)*, pages 1–10. ACM, 2010.

12 Timothy M. Chan, Kasper Green Larsen, and Mihai Pătraşcu. Orthogonal range searching on the RAM, revisited. In *Symposium on Computational Geometry (SoCG)*, pages 1–10, 2011.

13 Timothy M. Chan and Moshe Lewenstein. Clustered integer 3SUM via additive combinatorics. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 31–40, 2015.

14 Bernard Chazelle. Lower Bounds for Orthogonal Range Searching: I. The Reporting Case. *J. ACM*, 37(2):200–212, 1990.

15 Bernard Chazelle. Lower Bounds for Orthogonal Range Searching II. The Arithmetic Model. *J. ACM*, 37(3):439–463, 1990.

16 Bernard Chazelle and Burton Rosenberg. Simplex Range Reporting on a Pointer Machine. *Comput. Geom.*, 5:237–247, 1995.

17 Bernard Chazelle, Micha Sharir, and Emo Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Algorithmica*, 8:407–429, December 1992.

18 Hagai Cohen and Ely Porat. Fast set intersection and two-patterns matching. *Theor. Comput. Sci.*, 411(40-42):3795–3800, 2010.

19 Hagai Cohen and Ely Porat. On the hardness of distance oracle for sparse graph. http://arxiv.org/abs/1006.1117, 2010.

20 Richard Cole, Lee-Ad Gottlieb, and Moshe Lewenstein. Dictionary matching and indexing with errors and don't cares. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 91–100, 2004.

21 Paul F. Dietz, Kurt Mehlhorn, Rajeev Raman, and Christian Uhrig. Lower bounds for set intersection queries. *Algorithmica*, 14(2):154–168, 1995.

22 Paolo Ferragina, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava. Two-dimensional substring indexing. *Journal of Computer and System Sciences*, 66(4):763–774, 2003. Special Issue on PODS 2001.

23 Johannes Fischer, Travis Gagie, Tsvi Kopelowitz, Moshe Lewenstein, Veli Mäkinen, Leena Salmela, and Niko Välimäki. Forbidden patterns. In *LATIN 2012: Theoretical Informatics – 10th Latin American Symposium, Arequipa, Peru, April 16-20, 2012. Proceedings*, pages 327–337, 2012.

24 Wing-Kai Hon, Rahul Shah, Sharma V. Thankachan, and Jeffrey Scott Vitter. String Retrieval for Multi-pattern Queries. In *String Processing and Information Retrieval – 17th International Symposium, SPIRE 2010, Los Cabos, Mexico, October 11-13, 2010. Proceedings*, pages 55–66, 2010.

25 Wing-Kai Hon, Rahul Shah, Sharma V. Thankachan, and Jeffrey Scott Vitter. Document Listing for Queries with Excluded Pattern. In *Combinatorial Pattern Matching – 23rd Annual Symposium, CPM 2012, Helsinki, Finland, July 3-5, 2012. Proceedings*, pages 185–195, 2012.

26 Casper Kejlberg-Rasmussen, Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Word-packing algorithms for dynamic connectivity and dynamic sets. http://arxiv.org/abs/1407.6755, 2014.

27 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. http://arxiv.org/abs/1407.6756, 2016.

28 Tsvi Kopelowitz, Seth Pettie, and Ely Porat. Higher lower bounds from the 3SUM conjecture. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1272–1287, 2016.

29 Kasper Green Larsen, J. Ian Munro, Jesper Sindahl Nielsen, and Sharma V. Thankachan. On Hardness of Several String Indexing Problems. In *Combinatorial Pattern Matching – 25th Annual Symposium, CPM 2014, Moscow, Russia, June 16-18, 2014. Proceedings*, pages 242–251, 2014.

30 Moshe Lewenstein, J. Ian Munro, Venkatesh Raman, and Sharma V. Thankachan. Less space: Indexing for queries with wildcards. *Theoretical Computer Science*, 557:120–127, 2014.

31 Moshe Lewenstein, Yakov Nekrich, and Jeffrey Scott Vitter. Space-efficient string indexing for wildcard pattern matching. *CoRR*, abs/1401.0625, 2014. URL: `http://arxiv.org/abs/1401.0625`.

32 Jiří Matoušek. Range searching with efficient hierarchical cuttings. *Discrete & Computational Geometry*, 10(2):157–182, 1993.

33 S. Muthukrishnan. Efficient algorithms for document retrieval problems. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 657–666, 2002.

34 Mihai Pătraşcu. Towards polynomial lower bounds for dynamic problems. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 603–610, 2010.

35 Mihai Pătraşcu. Unifying the landscape of cell-probe lower bounds. *SIAM Journal of Computing*, 40(3):827–847, 2011.

36 Mihai Pătraşcu, L. Roditty, and M. Thorup. A new infinity of distance oracles for sparse graphs. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 738–747, 2012.

37 Mihai Pătraşcu and Liam Roditty. Distance oracles beyond the thorup–zwick bound. *SIAM Journal on Computing*, 43(1):300–311, 2014.

38 M. Sohel Rahman and Costas S. Iliopoulos. Pattern matching algorithms with don't cares. In *SOFSEM 2007: Theory and Practice of Computer Science, 33rd Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 20-26, 2007, Proceedings Volume II*, pages 116–126, 2007.

39 Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *Journal of Computer and System Sciences*, 18(2):110–127, 1979.

40 Hing-Fung Ting and Yilin Yang. Dictionary matching with uneven gaps. In *Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 9133, page 247, 2015.

Proof Complexity Modulo the Polynomial Hierarchy: Understanding Alternation as a Source of Hardness

Hubie Chen*

University of the Basque Country (UPV/EHU), San Sebastián, Spain; and IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
hubie.chen@ehu.eus

Abstract

We present and study a framework in which one can present alternation-based lower bounds on proof length in proof systems for quantified Boolean formulas. A key notion in this framework is that of *proof system ensemble*, which is (essentially) a sequence of proof systems where, for each, proof checking can be performed in the polynomial hierarchy. We introduce a proof system ensemble called *relaxing QU-res* which is based on the established proof system *QU-resolution*. Our main results include an exponential separation of the tree-like and general versions of relaxing QU-res, and an exponential lower bound for relaxing QU-res; these are analogs of classical results in propositional proof complexity.

1998 ACM Subject Classification F.1.3 [Computation by Abstract Devices] Complexity measures and classes

Keywords and phrases proof complexity, polynomial hierarchy, quantified propositional logic

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.94

1 Introduction

Background. Traditionally, the area of *propositional proof complexity* studies proof length in propositional proof systems for certifying the unsatisfiability of instances of the *SAT problem*, which instances are quantifier-free propositional formulas [16, 5, 26]. This line of study is supported by multiple motivations; let us highlight a few. First, while satisfiable formulas can be easily certified by a satisfying assignment, it is also natural to desire efficiently verifiable proofs for unsatisfiable formulas (for instance, to check that a SAT algorithm judged unsatisfiability correctly); understanding whether and when proof systems have succinct proofs is a prime concern of this area. Relatedly, *SAT algorithms* for deciding the SAT problem can be typically shown to implicitly generate proofs in a proof system, and thus insight into proof length in the resulting proof system can be used to gain insight into the running-time behavior of SAT algorithms (see for example the discussions in [4, 1]). In addition, the question of whether or not there are proof systems admitting *polynomially bounded proofs* is (when formalized) equivalent to the question of whether or not NP is equal to coNP [16], and one can thus suggest that studying proof length in propositional proof systems sheds light on the relationship between these two complexity classes.

Over recent years, researchers have devoted increasing attention to methods for solving the *QBF problem*, a generalization of the SAT problem and a canonical PSPACE-complete

* This work was supported by the Spanish Project MINECO COMMAS TIN2013-46181-C2-R, Basque Project GIU15/30, and Basque Grant UFI11/45.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 94; pp. 94:1–94:14

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

problem; an instance of this problem is a propositional formula where each variable is either existentially or universally quantified. (*QBF* is short for *quantified Boolean formula*.) It is often suggested that the move to studying this more general problem is based on advances in the efficacy of SAT algorithms (see for example [27]). As reinforces this suggestion, let us point out that one can find QBF solution techniques which use SAT algorithms as black-box, primitive components, and hence which arguably conceive of and treat the SAT problem as feasibly solvable. For instance, sKizzo, a QBF solver dating back to 2005, would convert the QBF being processed to a SAT instance and then call a SAT solver, whenever this was *affordable* [8]. As another example, a different QBF solver which extensively calls a SAT solver during a backtrack-style search was developed and studied [25].

The rise in the study of the QBF problem has resulted in the identification of a number of core algorithmic techniques and corresponding proof systems that aim to capture these (see for example [13, 18, 17, 23, 3, 21, 9, 10] and the references therein). We refer to these proof systems as *QBF proof systems*; they can be used as a basis for certifying a decision for a QBF instance. One can motivate the study of QBF proof systems in much the same way that the study of propositional proof systems has been motivated; hence, these QBF proof systems would seem to suggest a new chapter in the study of proof complexity, and a new domain for the existing lines of inquiry thereof.

However, one is immediately confronted with a dilemma upon inspecting the very basic question of whether or not a typical QBF proof system requires long (exponentially sized) proofs – again, a primary type of question in traditional proof complexity. As an example, let us discuss *Q-resolution* [13], a QBF proof system which is heavily studied and used, in both theory and practice (see for example [2, 19, 18, 22, 23, 3, 9] and the references therein). When applied to SAT instances (viewed as instances of QBF where all variables are existentially quantified), Q-resolution behaves identically to resolution (a heavily studied propositional proof system), and hence the known exponential lower bounds on resolution proof length [20, 7] transfer immediately to Q-resolution. This observation leaves one with a lingering sentiment – which is often expressed by members of the community – that there is something left to be said. After all, Q-resolution is defined on QBF instances, which are substantially more general than SAT instances; the observation does not yield any information about how Q-resolution handles this extra generality, that is, how it copes with alternation of quantifiers. Indeed, there is a sharp disconnect between observing a lower bound for a QBF proof system via a set of SAT instances, and the mentioned treatment of the SAT problem, by QBF algorithms, as a feasibly solvable primitive. These considerations naturally lead to the question of whether or not one can formulate and prove a lower bound which arises from alternation.

Contributions. In this article, we present and study a framework in which it is possible to present such alternation-based lower bounds on proof length in QBF proof systems.

We define a *proof system ensemble* to be an infinite collection of proof systems, where in each proof system, whether or not a given string π constitutes a proof of a given formula Φ can be checked in the polynomial hierarchy (Definition 2). A proof system ensemble is considered to have *polynomially bounded proofs* (for a language) if it contains a proof system which has polynomially bounded proofs in the usual sense (Definition 4). As a result, it is straightforward to define proof system ensembles that have succinct proofs for any set of QBFs with bounded alternation, such as a set of SAT instances (and the proof system ensembles studied herein all have this property); this in turn forces proof length lower bounds, by

nature, to arise from a proof system's inability to cope with quantifier alternation.[1] In terms of complexity classes, the question of whether or not there exists a polynomially bounded proof system ensemble for the QBF problem (or any other PSPACE-complete problem) is equivalent to the question of whether or not PSPACE is contained in PH, the polynomial hierarchy (Proposition 5). Indeed, the relationship that traditional proof complexity bears to the NP equals coNP question is analogous to the relationship between the present framework and the PSPACE equals PH question. (Let us point out that no direct implication is known between these two open questions, and so, in a certain sense, progress in one framework may proceed orthogonally to progress in the other!)

One of our main motivations in pursuing this work was to gain further insight into Q-resolution; here, we focus on a slight extension, *QU-resolution* [18], where from existing clauses one can derive new clauses in two ways: by a rule for eliminating literals on universally quantified variables and by resolving two clauses on any variable (in Q-resolution, one can only resolve on existentially quantified variables). Q-resolution, QU-resolution, and their relatives are typically defined only for clausal QBFs – QBFs that consist of a quantifier prefix followed by a conjunction of clauses. We show how to parameterize and lift QU-resolution to obtain a proof system ensemble which we call *relaxing QU-res* which is in fact defined on arbitrary QBFs (indeed, it is defined on what we call *quantified Boolean circuits*), and not just those in clausal form; relaxing QU-res is the main proof system ensemble that we study. Let us overview how we define it.

- We define an *axiom* of a QBF to be a clause which is, in a certain precise sense, entailed by the QBF (see Section 4.1).
- We then show that, given a QBF Φ and a partial assignment a to some of its variables, one can define a QBF $\Phi[a]$ derived naturally from Φ, where the variables on which a is defined have been instantiated (in a certain precise sense; see Section 4.2). This QBF $\Phi[a]$ has the key property that if it is false, then the clause corresponding to a is an axiom of the QBF Φ (see Proposition 8 for a precise statement). We view the notion of inferring clauses from the falsity of QBFs whose variables are partially instantiated as highly natural; indeed, in the case of SAT, performing such inferences is a basis of modern backtracking SAT solvers that perform *clause learning*.
- Recall that each proof system in our proof system ensemble may use, as an oracle, a level of the PH; in particular, the QBF problem restricted to a constant number of alternations may be used as an oracle. In order to infer clauses from a QBF Φ using the method just described, we need a way of detecting falsity of QBFs having the form $\Phi[a]$. But in general, this is difficult; such a QBF $\Phi[a]$ may have a high number of alternations, and thus might not be immediately decidable using an oracle of the described form. To the end of permitting the falsity detection of QBFs $\Phi[a]$ using such oracles, we define the notion of a *relaxation* of a QBF. A relaxation of a QBF Φ is obtained from Φ by changing the order of the quantifier/variable pairs in the quantifier prefix; roughly speaking, such a pair Qv may be moved to the left if Q is the universal quantifier ($\forall$), and may be moved to the right if Q is the existential quantifier ($\exists$). (See Section 4.2 for the precise definition.) A key property of this notion is that if a relaxation of a QBF Φ is false, then the QBF Φ is false (Proposition 9).

[1] Note that there is, a priori, a difference between allowing proof systems oracle access to the SAT problem – which would be natural for modelling QBF solvers that treat the SAT problem as feasibly solvable – and allowing oracle access to arbitrary levels of the PH. We focus on the latter for various reasons: the proof length lower bounds will arise from alternation; we believe that this results in a more robust model; and, this focus causes the proof length lower bounds, which are here of primary interest, to be stronger.

With this notion of relaxation in hand, we define, for each $k \geqslant 2$, the set $H(\Phi, \Pi_k)$ to contain the axioms that arise from QBFs $\Phi[a]$ having Π_k-relaxations (relaxations with a Π_k prefix) that are false. That is, in this set we collect the axioms obtainable by detecting falsity of QBFs $\Phi[a]$ via the consideration of Π_k-relaxations. (Hence, the detection is sound in that it is always correct, but it is not complete). Note that it holds that

$$H(\Phi, \Pi_2) \subseteq H(\Phi, \Pi_3) \subseteq H(\Phi, \Pi_4) \subseteq \cdots .$$

- This gives us a sequence of versions of QU-resolution: for each k, we obtain a version by defining a proof to be a sequence of clauses derived from the axioms $H(\Phi, \Pi_k)$ and the two aforementioned rules of QU-resolution. This sequence is the proof system ensemble *relaxing QU-res*. Let us remark that each of these versions is sound and complete, in a precise sense (see Definition 2 and Proposition 12).

A couple of remarks are in order. First, note that the empty clause is an axiom in $H(\Phi, \Pi_k)$ whenever Φ is a false QBF whose quantifier prefix is Π_k. Consequently, relaxing QU-res is polynomially bounded on any set of false QBFs having bounded alternation. Let us also note that although here we explicitly lift QU-resolution to a proof system ensemble, the approach that we take here can be applied to analogously lift any proof system which is based on deriving clauses from a set of axiom clauses.

Apart from the formulation of the framework, our main results are as follows. We prove an exponential separation between the tree-like and general versions of relaxing QU-res (Section 6), by exhibiting a set of formulas which have polynomial size QU-resolution proofs, but which require exponential size proofs in tree-like relaxing QU-res; this gives an alternation-based analog of the known separation between tree-like and general resolution [12, 6]. Tree-like QU-resolution proofs can be viewed as the traces of a natural backtrack-style QBF decision procedure (this is evident from the viewpoint in Section 4.3, and is also developed explicitly in [14, Section 4.3]), and so this separation formally differentiates the power of such backtracking and general QU-resolution. The lower bound of this separation is based on a prover-delayer game for tree-like QU-resolution proofs (Section 5), which can be viewed as a generalization of a known prover-delayer game for tree-like resolution [24]; note that independently of our work [15], a game similar to ours was presented for tree-like Q-resolution [11]. We also prove an exponential lower bound for relaxing QU-res (Section 7).

All in all, the ideas and techniques developed in this work draw upon and interface concepts from two-player game interaction, proof complexity, and quantified propositional logic. We believe that further progress could benefit from creative input from each of these areas, and certainly look forward to future research on the presented framework.

2 Preliminaries

For each integer k, we use $[k]$ to denote the set that is equal to $\{1, \ldots, k\}$ when $k \geqslant 1$, and that is equal to the empty set $\varnothing$ when $k < 1$. We use $\mathbb{N}$ to denote the natural numbers $\{0, 1, 2, \ldots\}$.

We use $\mathrm{dom}(f)$ to indicate the domain of a function. A function f is a *restriction* of a function g if $\mathrm{dom}(f) \subseteq \mathrm{dom}(g)$ and, for each $a \in \mathrm{dom}(f)$, it holds that $g(a) = f(a)$; when this holds, we also say that g is an *extension* of f. When f is a function, we use $f[a \to b]$ to denote the function on domain $\mathrm{dom}(f) \cup \{a\}$ that maps a to b, and otherwise behaves like f. We write $f \restriction S$ to denote the restriction of a function f to the set S. We say that two functions f and g *agree* if for each element $a \in \mathrm{dom}(f) \cap \mathrm{dom}(g)$, it holds that $f(a) = g(a)$.

When A and B are sets, we use $[A \to B]$ to denote the set of functions from A to B.

Clauses. In this article, we employ the following terminology to discuss clauses. A *literal* is a propositional variable v or the negation $\overline{v}$ thereof. Two literals are *complementary* if one is a variable v and the other is $\overline{v}$; each is said to be the *complement* of the other. A *clause* is a disjunction of literals that contains, for each variable, at most one literal on the variable. A clause is sometimes viewed as the set of the literals that it contains; two clauses are considered equal if they are equal as sets. A clause is *empty* if it does not contain any literals. The variables of a clause are simply the variables that underlie the clause's literals, and the set of variables of a clause α is denoted by $\mathrm{vars}(\alpha)$. When α is a clause, we use $\mathrm{assign}(\alpha)$ to denote the unique propositional assignment f with $\mathrm{dom}(f) = \mathrm{vars}(\alpha)$ such that α evaluates to false under f. In the other direction, when f is a propositional assignment, we use $\mathrm{clause}(f)$ to denote the unique clause α with $\mathrm{vars}(\alpha) = \mathrm{dom}(f)$ that evaluates to false under f. *We will freely and tacitly interchange between a clause α and its corresponding assignment* $\mathrm{assign}(\alpha)$. A clause γ is a *resolvent* of two propositional clauses α and β on variable v if there exists a literal $L \in \alpha$ such that its complement M is in β, $\gamma = (\alpha \backslash \{L\}) \cup (\beta \backslash \{M\})$, and v is the variable underlying L and M.

Quantified Boolean circuits and formulas. We assume basic familiarity with quantified propositional logic. A *QBC* (short for *quantified Boolean circuit*) consists of a quantifier prefix $\vec{P} = Q_1 v_1 \ldots Q_n v_n$, where each Q_i is a quantifier in $\{\forall, \exists\}$ and each v_i is a propositional variable; and, a Boolean circuit ϕ built from the constants 0 and 1, propositional variables among $\{v_1, \ldots, v_n\}$, and the gates AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). We refer to the computational problem of deciding whether or not a QBC is false as the *QBC problem*. For brevity, we sometimes refer to existentially quantified variables as $\exists$-variables, and universally quantified variables as $\forall$-variables. While it is typical to notate a QBC by simply specifying the prefix $\vec{P}$ immediately followed by the circuit ϕ, we will typically separate these two parts by a colon for the sake of readability, using for example $\vec{P} : \phi$. We assume that each quantifier prefix does not contain repeated variables. When $\Phi = \vec{P} : \phi$ is a QBC, by a *partial assignment of* Φ, we refer to a propositional assignment $f : S \to \{0, 1\}$ defined on a subset S of the variables appearing in $\vec{P}$. A *QBF* is a QBC $\vec{P} : \phi$ where ϕ is a Boolean formula. A *clausal QBF* is a QBF $\vec{P} : \phi$ where ϕ is the conjunction of clauses.

Quantifier prefixes. Let $i \geqslant 1$. A quantifier prefix $\vec{P} = Q_1 v_1 \ldots Q_n v_n$ is Π_i if $Q_1 \ldots Q_n$, viewed as a string over the alphabet $\{\forall, \exists\}$, is contained in the language denoted by the regular expression $\forall^*\exists^*\forall^*\exists^* \ldots$, which contains i starred quantifiers, beginning with $\forall^*$ and alternating; Σ_i is defined similarly, but with respect to the regular expression $\exists^*\forall^*\exists^*\forall^* \ldots$.

The following notation is relative to a quantifier prefix $\vec{P} = Q_1 v_1 \ldots Q_n v_n$; when we use it, the prefix will be clear from context. We write $v_i \preceq v_j$ if $i \leqslant j$ or if $j < i$ and $Q_j = Q_{j+1} = \cdots = Q_i$. We extend this binary relation (and others) to sets in the following natural way: when U and V are sets of variables, we write $U \preceq V$ if for each $u \in U$ and each $v \in V$, it holds that $u \preceq v$. We also write, for example, that $U \preceq v$ for a single variable v when $U \preceq \{v\}$. We write $v_i \equiv v_j$ if $v_i \preceq v_j$ and $v_j \preceq v_i$. It is straightforward to verify that $\equiv$ is an equivalence relation; we refer to each equivalence class of $\equiv$ as a *quantifier block*. We write $v_i \precneqq v_j$ if $v_i \preceq v_j$ and $v_i \not\equiv v_j$. When S is a set of variables, we use $\mathrm{last}(S)$ to denote the variable of S appearing last in the quantifier prefix, that is, the variable v_m, where $m = \max\{i \mid v_i \in S\}$. Typically, when we use the function $\mathrm{last}(S)$, it is in conjunction with the just-defined binary relations, and hence what is most relevant will be the relative location of the quantifier block of $\mathrm{last}(S)$.

Strategies. Let $\Phi = \vec{P} : \phi$ be a QBC; let X denote the $\exists$-variables of Φ, and let Y denote the $\forall$-variables of Φ. When $x \in X$, define $Y_{<x}$ to be the set of variables $\{y \in Y \mid y \precnsim x\}$; dually, when $y \in Y$, define $X_{<y}$ to be the set of variables $\{x \in X \mid x \precnsim y\}$.

An *$\exists$-strategy* is a sequence of mappings $\sigma = (\sigma_x)_{x \in X}$ where each σ_x is a mapping from $[Y_{<x} \to \{0,1\}]$ to $\{0,1\}$. When $\tau : Y \to \{0,1\}$ is an assignment to the universally quantified variables, we use $\langle \sigma, \tau \rangle$ to denote the assignment f defined by $f(y) = \tau(y)$ for each $y \in Y$ and $f(x) = \sigma_x(\tau \restriction Y_{<x})$ for each $x \in X$. We say that $(\sigma_x)_{x \in X}$ is a *winning $\exists$-strategy* if for every assignment $\tau : Y \to \{0,1\}$, it holds that the assignment $\langle \sigma, \tau \rangle$ satisfies ϕ. A *model* of Φ is defined to be a winning $\exists$-strategy of Φ.

Dually, we define a *$\forall$-strategy* to be a sequence of mappings $\tau = (\tau_y)_{y \in Y}$ where each τ_y is a mapping from $[X_{<y} \to \{0,1\}]$ to $\{0,1\}$. When $\sigma : X \to \{0,1\}$ is an assignment to the existentially quantified variables, we use $\langle \tau, \sigma \rangle$ to denote the assignment f defined by $f(x) = \sigma(x)$ for each $x \in X$ and $f(y) = \tau_y(\sigma \restriction X_{<y})$ for each $y \in Y$. We say that $(\sigma_y)_{y \in Y}$ is a *winning $\forall$-strategy* if for every assignment $\sigma : X \to \{0,1\}$, it holds that the assignment $\langle \tau, \sigma \rangle$ falsifies ϕ.

The following are well-known facts that we will treat as basic.

▶ **Proposition 1.** *Let Φ be a QBC.*

- *There exists a winning $\exists$-strategy for Φ (that is, a model of Φ) if and only if Φ is true.*
- *There exists a winning $\forall$-strategy for Φ if and only if Φ is false.*

3 Proof system ensembles

In this section, we formalize the notion of *proof system ensemble* and present some basic associated notions.

For each $m \geqslant 1$, fix $S(m)$ to be the QBC problem restricted to QBCs having a Σ_m prefix, which is a Σ^p_m-complete problem; for $m = 0$, fix $S(m)$ to be a polynomial-time decidable problem.

Let O be a language; when discussing an algorithm A that makes oracle calls, we use A^O to denote the instantiation of A where oracle calls are answered according to O.

▶ **Definition 2.** A *proof system ensemble (A, r) for a language L* consists of an algorithm A which may make oracle calls and receives inputs of the form $(k, (x, \pi))$ where $k \in \mathbb{N}$ and x and π are strings; and, a computable function $r : \mathbb{N} \to \mathbb{N}$ such that:

- For each $k \in \mathbb{N}$, there exists a polynomial p_k such that (for each pair (x, π)) the algorithm $A^{S(r(k))}$ halts on an input $(k, (x, \pi))$ within time $p_k(|(x, \pi)|)$.
- For each $k \in \mathbb{N}$, when L_k is set to $\{(x, \pi) \mid (k, (x, \pi))$ is accepted by $A^{S(r(k))}\}$, it holds that the language $\{x \mid \exists \pi$ such that $(x, \pi) \in L_k\}$ is equal to L.

Let us provide an intuitive explanation of Definition 2. For each fixed value of k, the algorithm A provides a proof system for the language L; on inputs of the form $(k, (x, \pi))$, the algorithm is provided oracle access to $S(r(k))$, and needs to accept or reject within polynomial time (in $|(x, \pi)|$). Acceptance indicates that π is judged to be a proof that $x \in L$. The second condition in the definition states that each such proof system is sound and complete, that is, for each fixed k, an arbitrary string x is in L iff there exists a string π such that $(k, (x, \pi))$ is accepted by A.

We use the following terminology to present lower bounds on proof size in proof system ensembles.

▶ **Definition 3.** Let Z be a set of functions from $\mathbb{N}$ to $\mathbb{N}$. A proof system ensemble (A, r) *requires proofs of size* Z on a sequence $\{\Phi_1, \Phi_2, \ldots\}$ of instances if for each k, there exists $z \in Z$ where (for all $n \geqslant 1$ and all strings π) it holds that $(\Phi_n, \pi) \in L_k$ implies $|\pi| \geqslant z(n)$. Here, $|\pi|$ denotes the size of π. We also apply this terminology to other measures defined on proofs.

We say that a function f mapping strings to strings is a *polynomial-length function* if there exists a polynomial q such that, for each string x, it holds that $|f(x)| \leqslant q(|x|)$.

▶ **Definition 4.** A proof system ensemble (A, r) is *polynomially bounded* on a language L if there exists $k \in \mathbb{N}$ and there exists a polynomial-length function f (mapping strings to strings) such that the following holds: if $x \in L$, then it holds that $(x, f(x)) \in L_k$, where L_k is defined as in Definition 2.

▶ **Proposition 5.** *There exists a polynomially bounded proof system ensemble for a language L if and only if L is in the polynomial hierarchy.*

4 Relaxing QU-resolution

4.1 QU-resolution

Let $\Phi = \vec{P} : \phi$ be a QBC. We define an *axiom set of* Φ to be a set H of clauses on variables of $\vec{P}$ such that, for each $C \in H$, C is an *axiom* of Φ in the following sense: each model of $\vec{P} : \phi$ is a model of $\vec{P} : C$. Let us give examples. First, if the QBC Φ is false, then the empty clause is an axiom of Φ. Second, if C is any clause which is entailed by ϕ, then C is an axiom of Φ. A case of this is when a is an assignment to all variables of Φ that falsifies ϕ; then, clause(a) is entailed by ϕ and is an axiom of Φ.

Relative to a QBC $\Phi = \vec{P} : \phi$, we say that a clause C is obtainable from a second clause D by $\forall$-*elimination* if there exists a literal $L \in D$ such that $C = D \backslash \{L\}$ and the variable y underlying L is a $\forall$-variable and has $\text{vars}(C) \preceq y$.

With these notions, we define QU-resolution for quantified Boolean circuits in the following way.

▶ **Definition 6.** A *QU-resolution proof* of a QBC $\Phi = \vec{P} : \phi$ from an axiom set H (of Φ) is a finite sequence of clauses where each clause is either in H, is obtainable from a previous clause by $\forall$-elimination, or is obtainable from two previous clauses as a resolvent; in the last two cases, we assume that the clause is annotated with the previous clause(s) from which it is derived (this is to provide a clean correspondence between proofs and certain graphs to be defined, see Section 4.3). The *size* of such a proof is defined as the number of clauses. Such a proof is said to be a *falsity proof* if it ends with the empty clause.

It is a folklore and readily verified fact that when one has a clausal QBF $\Phi = \vec{P} : \phi$ with clause set H, and C appears in a QU-resolution proof of Φ from H, then any model of Φ is a model of $\vec{P} : C$. From this fact and the definition of axiom set, we immediately obtain the following proposition.

▶ **Proposition 7.** *Let C be a clause appearing in a QU-resolution proof of a QBC $\Phi = \vec{P} : \phi$ from axiom set H. Each model of $\vec{P} : \phi$ is a model of $\vec{P} : C$. Consequently, if C is the empty clause, then the QBC Φ is false.*

4.2 Relaxing

In order to define a proof system ensemble based on QU-resolution proofs, we now describe how to obtain a sequence of axiom sets for a given QBC. We start by exhibiting a way to infer that a partial assignment is an axiom of a QBC.

Let a be a partial assignment of a QBC $\Phi = \vec{P} : \phi$. Define $\vec{P}[a]$ to be the quantifier prefix which is equal to $\vec{P}$ but where the variables in $\mathrm{dom}(a)$ and their corresponding quantifiers are removed, and where each quantifier of a variable v with $v \preceq \mathrm{last}(a)$ is changed (if necessary) to an existential quantifier. Define $\phi[a]$ to be the circuit obtained from ϕ by replacing each variable $v \in \mathrm{dom}(a)$ with the constant $a(v)$. Define $\Phi[a]$ to be $\vec{P}[a] : \phi[a]$.

▶ **Proposition 8.** *Assume that a is a partial assignment of a QBC $\Phi = \vec{P} : \phi$ such that $\Phi[a]$ is false. Then* $\mathrm{clause}(a)$ *is an axiom of Φ, that is, each model of $\vec{P} : \phi$ is a model of $\vec{P} : \mathrm{clause}(a)$.*

We believe that Proposition 8 provides a natural way to derive axioms from a QBC. Consider the case where Φ is a SAT instance, that is, $\vec{P}$ is purely existential. In this case, if a is a partial assignment such that $\Phi[a]$ is false, then $\mathrm{clause}(a)$ is an axiom of Φ. Indeed, in this case $\Phi[a]$ is simply the QBC instance obtained by instantiating variables according to a, and then removing the instantiated variables from the quantifier prefix. Note that, in the context of backtrack search for SAT, it is typical that, when some variables have been set according to a partial assignment a, a solver attempts to detect falsity of $\Phi[a]$ by heuristics such as unit propagation and generalizations thereof.

In the case of general QBCs, it is natural to ask, when one has a partial assignment a and then instantiates its variables in ϕ to obtain $\phi[a]$, under what conditions $\mathrm{clause}(a)$ can be inferred as an axiom. Proposition 8 provides an answer to this question; let us explain intuitively why the quantifier prefix is adjusted to $\vec{P}[a]$. Consider the case where the first quantifier block of $\vec{P}$ is existential and a is a partial assignment to variables from this first block; then $\vec{P}[a]$ is simply $\vec{P}$ but with the variables of a removed, and so this case of the proposition generalizes the purely existential case just discussed. In the case where a is arbitrary, $\vec{P}[a]$ can be viewed as the prefix where the lowest number of quantifiers have been changed from universal to existential such that the first quantifier block is existential, and all variables of a fall into this first block.

Prima facie, Proposition 8 may appear to be of limited utility; even if one has oracle access to a level of the polynomial hierarchy, it may be that many partial assignments a give rise to a quantifier prefix $\vec{P}[a]$ which has too many alternations to be resolved by the oracle. In order to expand the class of axioms derivable by this proposition (relative to such an oracle), we introduce now the notion of a *relaxation* of a QBC.

A *relaxation* of a quantifier prefix $\vec{P} = Q_1 v_1 \dots Q_n v_n$ is a quantifier prefix which has the form $\vec{P'} = Q_{\pi(1)} v_{\pi(1)} \dots Q_{\pi(n)} v_{\pi(n)}$ where $\pi : [n] \to [n]$ is a permutation and where, for each $\forall$-variable y and for each $\exists$-variable x, it holds that $y \preceq x$ implies $y \preceq' x$; here, $\preceq$ and $\preceq'$ denote the binary relations of $\vec{P}$ and $\vec{P'}$, respectively. As an example, consider the quantifier prefix $\vec{P} = \exists x_1 \exists x_2 \forall y \forall y' \exists x_3$; relaxations thereof include $\forall y \forall y' \exists x_1 \exists x_2 \exists x_3$, $\exists x_1 \forall y' \exists x_2 \forall y \exists x_3$, and $\forall y' \exists x_2 \forall y \exists x_1 \exists x_3$. A *relaxation of a QBC $\vec{P} : \phi$* is a QBC of the form $\vec{P'} : \phi$ where $\vec{P'}$ is a relaxation of $\vec{P}$; such a QBC is said to be a *Π_i-relaxation* if $\vec{P'}$ is Π_i.

The following is straightforward to verify.

▶ **Proposition 9.** *If a relaxation of a QBC Φ is false, then the QBC Φ is false.*

Note that for any quantifier prefix, a relaxation may be obtained by simply placing the universal quantifiers and their variables first, followed by the existential quantifiers and their

variables. Hence, in this sense, each QBC has a canonical Π_2-relaxation, and in the sequel, we focus the discussion on relaxations that are Π_k-relaxations for values of k greater than or equal to 2.

Let Φ be a QBC; for $k \geqslant 2$, we define $H(\Phi, \Pi_k)$ to be the set that contains a clause C if there exists a Π_k-relaxation of $\Phi[\text{assign}(C)]$ that is false. The following fact follows immediately from Propositions 8 and 9.

▶ **Proposition 10.** *When Φ is a QBC and $k \geqslant 2$, it holds that $H(\Phi, \Pi_k)$ is an axiom set of Φ.*

▶ **Definition 11.** *Relaxing QU-res* is defined as the pair (A, r) where r is defined by $r(k) = k + 3$ and A is an algorithm defined to accept an input $(k, (\Phi, \pi))$ if Φ is a QBC and π is a QU-resolution falsity proof of Φ from axioms in $H(\Phi, \Pi_{k+2})$. In particular, the algorithm A examines each clause in π in order; when a clause C is not derived from previous ones by resolution or by $\forall$-elimination, membership of C in $H(\Phi, \Pi_{k+2})$ is checked by the Σ_{k+3} oracle. (Such an oracle can nondeterministically guess a Π_{k+2}-relaxation and then check this relaxation for falsity.)

▶ **Proposition 12.** Relaxing QU-res *is a proof system ensemble for the language of false QBCs.*

Let us now introduce some notions which will be used in our study of *tree-like relaxing QU-res* (defined below). Let f and g be partial assignments of a QBC Φ. We say that g is a *semicompletion* of f if g is an extension of f such that for each universally quantified variable y with $\text{dom}(f) \preceq y$ and $y \notin \text{dom}(f)$, it holds that $\text{dom}(g) \preceq y$ and $y \notin \text{dom}(g)$. A set H of partial assignments of Φ is *semicompletion-closed* if, whenever $f \in H$ and g is a semicompletion of f, it holds that $g \in H$.

4.3 A graph-based view

When $\pi = C_1, \ldots, C_n$ is a QU-resolution proof of a QBC $\vec{P} : \phi$ from axioms H, define $G(\pi)$ to be the directed acyclic graph where there is a vertex for each clause occurrence C_i, which vertex has label $\text{assign}(C_i)$; and, where (for all pairs of clauses C_i, C_j) there is a directed edge from the vertex of C_j to the vertex of C_i if C_j is derived from C_i.

▶ **Proposition 13.** *Let π be a QU-resolution proof of a QBC $\vec{P} : \phi$ from axioms H. The directed acyclic graph $G(\pi)$ has the following properties:*

(α) *If a node with label a has no out-edges, then* $\text{clause}(a)$ *is an element of H.*

(β) *If a node with label a has 1 out-edge to a node with label a', then a' is an extension of a with $\text{dom}(a') = \text{dom}(a) \cup \{y\}$ where y is a universally quantified variable with $\text{dom}(a) \preceq y$.*

(γ) *If a node with label a has 2 out-edges to nodes with labels a_1 and a_2, then there exists a variable v such that a_1 and a_2 are defined on v and $a_1(v) \neq a_2(v)$; $(\text{dom}(a_1) \cup \text{dom}(a_2)) \backslash \{v\} = \text{dom}(a)$; a and a_1 are equal on the variables where they are both defined; and, a and a_2 are equal on the variables where they are both defined.*

Moreover, a labelled graph with these three properties naturally induces a QU-resolution proof: for each node, let a be its label, and associate to it $\text{clause}(a)$. □

▶ **Definition 14.** We say that a QU-resolution proof π is *tree-like* if the graph $G(\pi)$ is a tree. We define *tree-like relaxing QU-res* to be the proof system ensemble (A', r) described as follows. Let (A, r) denote relaxing QU-res. Then, the algorithm A' accepts an input $(k, (x, \pi))$ if A accepts it and π is tree-like.

5 A prover-delayer game for *tree-like relaxing QU-res*

In this section, we present a game that can be used to exhibit lower bounds on the size of tree-like QU-resolution proofs; this game can be viewed as a generalization of a game for studying tree-like resolution, which game was presented by Pudlák and Impagliazzo [24].

We first give an intuitive description of the game. Note, however, that this description is meant only to be suggestive. For a precise description, we urge the reader to consult the formal definition, which follows (Definition 15).

Relative to a QBC Φ and a set H of axioms, the game is played between two players, *Prover* and *Delayer*, which maintain a partial assignment. Prover's goal is to reach a partial assignment in H, while Delayer tries to slow down Prover, scoring points in the process. Prover starts by announcing the empty assignment, and Delayer responds with a semicompletion thereof. After this, the play proceeds in a sequence of rounds. In each round, Prover may perform one of three actions to the current assignment f: select a restriction of f; assign a value to a $\forall$-variable $y \notin \mathrm{dom}(f)$ having $\mathrm{dom}(f) \preceq y$; or, select a variable $v \notin \mathrm{dom}(f)$. In the first two cases, Delayer responds with a semicompletion of the resulting assignment. In the third case, Delayer may give a choice to the Prover. When a choice is given, the Prover sets the value of v, and Delayer may elect to claim a point which is then associated with v. When no choice is given, Delayer sets the value of v. After v is set, Delayer responds (as in the first two cases) with a semicompletion of the resulting assignment. Delayer is said to have a p-point strategy if, he has a strategy where, by the time that Prover achieves her goal, there are p variables on which the final assignment is defined such that Delayer has claimed points on these variables. In what follows, we assume $p \geqslant 1$.

▶ **Definition 15.** Let Φ be a QBC. Relative to a set H of axioms, a *p-point delayer strategy* consists of a set F of partial assignments of Φ and a function $s : F \to \mathbb{N}$ called the *score function* such that the following properties hold:

- *(semicompletion-of-empty)* There exists a semicompletion $g \in F$ of the empty assignment such that $s(g) = 0$.
- *(all-points)* If $f \in F \cap H$, then $s(f) \geqslant p$.
- *(monotonicity)* If $g \in F$, then each restriction of g has a semicompletion $f \in F$ such that $s(f) \leqslant s(g)$.
- *($\forall$-branching)* If $f \in F$ and $y \notin \mathrm{dom}(f)$ is a universally quantified variable with $\mathrm{dom}(f) \preceq y$, then, for each $b \in \{0,1\}$, the assignment $f[y \to b]$ has a semicompletion $g \in F$ with $s(g) = s(f)$.
- *(double-branching)* If $f \in F$ and $v \notin \mathrm{dom}(f)$, there exists a value $b \in \{0,1\}$ such that $f[v \to b]$ has a semicompletion $g \in F$ where (1) $s(g) \leqslant s(f) + 1$ and (2) if $s(g) = s(f) + 1$, the assignment $f[v \to \neg b]$ has a semicompletion $g' \in F$ with $s(g') \leqslant s(f) + 1$.

▶ **Theorem 16.** *Assume that there exists a p-point delayer strategy for a QBC Φ with respect to a semicompletion-closed axiom set H, and that π is a tree-like QU-resolution proof ending with the empty clause, from axioms H. Then, the tree $G(\pi)$ has at least 2^p leaves.*

6 Separation of the tree-like and general versions of *relaxing QU-res*

The family of sentences to be studied in this section is defined as follows. For each $i \in \{0\} \cup [n]$, define X_i to be the variable set $\{x_{i,j,k} \mid j,k \in \{0,1\}\}$, and for each $i \in [n]$, define X'_i analogously to be the variable set $\{x'_{i,j,k} \mid j,k \in \{0,1\}\}$. Define $\vec{P}_n$ to be the prefix $\exists X_0 \exists X'_1 \forall y_1 \exists X_1 \exists X'_2 \forall y_2 \exists X_2 \ldots \exists X'_n \forall y_n \exists X_n$. Note that, for a set of variables X, we use the

notation $\exists X$ to represent the existential quantification of the variables in X, in any order (our discussion will always be independent of any particular order chosen). For $i \in [n]$, we refer to the variables in $X'_i \cup \{y_i\} \cup X_i$ as the *level i variables*.

- Define $B = \{\neg x_{0,j,k} \mid j,k \in \{0,1\}\} \cup \{x_{n,j,0} \vee x_{n,j,1} \mid j \in \{0,1\}\}$.
- For each $i \in [n]$ and each $j \in \{0,1\}$ define $H_{i,j} = \{\neg x'_{i,0,k} \vee \neg x'_{i,1,l} \vee x_{i-1,j,0} \vee x_{i-1,j,1} \mid k,l \in \{0,1\}\}$.
  Observe that the clause $\neg x'_{i,0,k} \vee \neg x'_{i,1,l} \vee x_{i-1,j,0} \vee x_{i-1,j,1}$ is logically equivalent to $(x'_{i,0,k} \wedge x'_{i,1,l}) \to (x_{i-1,j,0} \vee x_{i-1,j,1})$.
- For each $i \in [n]$, define $T_i = \{\neg x_{i,0,k} \vee y_i \vee x'_{i,0,k} \mid k \in \{0,1\}\} \cup \{\neg x_{i,1,k} \vee \neg y_i \vee x'_{i,1,k} \mid k \in \{0,1\}\}$.

Define ϕ_n to be the conjunction of the clauses contained in the just-defined sets. Define Φ_n as $\vec{P}_n : \phi_n$. This definition of this family of sentences was inspired partially by the separating formulas of [12, 6].

Let us explain intuitively what the clauses mandate and why the sentences Φ_n are false. By the clauses in B, all of the variables $x_{0,j,k}$ must be set to 0. By the clauses in the sets $H_{1,j}$, either both variables $x'_{1,0,k}$ or both variables $x'_{1,1,k}$ must be set to 0. Once this occurs, the universal player can set the variable y_1 to 0 or 1 to force either both variables $x_{1,0,k}$ or both variables $x_{1,1,k}$ to 0 (respectively), via the clauses in T_1. This reasoning can then be repeated; for instance, at the next level, either both variables $x'_{2,0,k}$ or both variables $x'_{2,1,k}$ must be set to 0, and then after universal player assigning y_2 appropriately, either both variables $x_{2,0,k}$ or both variables $x_{2,1,k}$ are forced to 0. In the end, the existential player must violate one of the two clauses in B concerning level n.

▶ **Proposition 17.** *The sentences $\{\Phi_n\}_{n \geqslant 1}$ have QU-resolution proofs of size linear in n.*

Let $n \geqslant 1$; we will use the following terminology to discuss Φ_n.

We say that r is a *normal realization* of level $i \in [n]$ if it is an assignment defined on the level i variables such that, when b is set to $r(y_i)$, the following hold:

- $0 = r(x_{i,b,0}) = r(x'_{i,b,0}) = r(x_{i,b,1}) = r(x'_{i,b,1})$
- $r(x_{i,\neg b,0}) = r(x'_{i,\neg b,0}) \neq r(x_{i,\neg b,1}) = r(x'_{i,\neg b,1})$

We say that r is a *funny realization* of level $i \in [n]$ if it is an assignment defined on the level i variables such that, when b is set to $r(y_i)$, the following hold:

- $r(x_{i,b,0}) = r(x'_{i,b,0}) \neq r(x_{i,b,1}) = r(x'_{i,b,1})$
- $0 = r(x'_{i,\neg b,0}) = r(x'_{i,\neg b,1})$
- $r(x_{i,\neg b,0}) \neq r(x_{i,\neg b,1})$

We state two key and straightforwardly verified properties of realizations in the following proposition.

▶ **Proposition 18.** *No assignment defined on the level i variables is both a normal realization and a funny realization. Also, each normal realization and each funny realization (of level i) satisfies all clauses in T_i.*

We define the set of assignments F_n to be the set containing all *normal assignments* and all *funny assignments*, which we now turn to define. Let f be a partial assignment of Φ_n. Let $\ell \geqslant 0$ denote the maximum level ℓ such that f is defined on an $\exists$-variable in level ℓ.

We say that f is a *normal assignment* if the following hold:

- f is defined on the variables in $\{x_{0,j,k} \mid j,k \in \{0,1\}\}$ and equal to 0 on them.

- For each $i \in [\ell - 1]$, the restriction of f to the level i variables is a normal realization of level i.
- If $\ell \geqslant 1$, either the restriction of f to the level ℓ variables is a normal realization of level ℓ; or, f is *half-defined* on level ℓ, by which is meant that f is not defined on any variables in $\{x_{\ell,j,k} \mid j,k \in \{0,1\}\}$, but is defined on all variables in $\{x'_{\ell,j,k} \mid j,k \in \{0,1\}\}$ and has $\sum_{j,k\in\{0,1\}} x'_{\ell,j,k} = 1$.

For each normal assignment f, we define $s_n(f) = \ell$.

We say that f is a *funny assignment* if there exists $m \in [\ell]$ such that the following hold:

- f is defined on the variables in $\{x_{0,j,k} \mid j,k \in \{0,1\}\}$ and equal to 0 on them.
- For each $i \in [m-1]$, the restriction of f to the level i variables is a normal realization of level i.
- The restriction of f to the level m variables is a funny realization of level m.
- For each i with $m < i \leqslant \ell$ and for each $j \in \{0,1\}$, if f is defined on one of the four variables in $\{x_{i,j,k}, x'_{i,j,k} \mid k \in \{0,1\}\}$, then it is defined on all of them and $f(x_{i,j,0}) = f(x'_{i,j,0}) \neq f(x_{i,j,1}) = f(x'_{i,j,1})$.

The following result is obtained by applying the main theorem of the previous section to the strategies (F_n, s_n).

▶ **Theorem 19.** *Tree-like relaxing QU-res requires proofs of size $\Omega(2^n)$ on the sentences $\{\Phi_n\}_{n\geqslant 1}$.*

7 Lower bound for *relaxing QU-res*

We define a family of QBCs, to be studied in this section, as follows. Let $n \geqslant 1$. Define $\vec{P}_n$ to be the quantifier prefix $\exists x_1 \forall y_1 \ldots \exists x_n \forall y_n$. Define $\phi_{n,j}$ to be true if and only if $j + \sum_{i=1}^{n}(x_i + y_i) \not\equiv n \pmod 3$. Define Φ_n to be the sentence $\vec{P}_n : \phi_{n,0}$; these are the sentences that will be used to prove the lower bound. It is straightforward to verify that ϕ_n can be represented as a circuit of size polynomial in n, and we assume that ϕ_n is so represented.

▶ **Proposition 20.** *For each $n \geqslant 1$, the sentence Φ_n is false.*

To obtain the lower bound, we show that for any proof π, the graph $G(\pi)$ must have exponentially many sinks. We begin by showing that any assignment to an initial segment of the $\exists$-variables can be mapped naturally to a sink.

▶ **Lemma 21.** *Let π be a relaxing QU-res proof of Φ_n from an axiom set, and suppose $t \geqslant 1$. Let $f : \{x_1, \ldots, x_{n-\lceil t/2\rceil}\} \to \{0,1\}$ be an assignment. There exists a sink of $G(\pi)$ whose label agrees with f.*

We next show that each sink must be defined on a variable that occurs *towards the end* of the quantifier prefix, made precise as follows.

▶ **Lemma 22.** *Let π be a relaxing QU-res proof of Φ_n from axiom set $H(\Phi, \Pi_t)$, where $t \geqslant 2$ and $n \geqslant \lceil t/2\rceil$. Each sink of $G(\pi)$ has a label a that is defined on one of the following variables:*

$$x_{n-(\lceil t/2\rceil-1)}, y_{n-(\lceil t/2\rceil-1)}, \ldots, x_{n-1}, y_{n-1}, x_n, y_n\,.$$

When f is a partial assignment of Φ_n, we refer to the elements of $\{v \mid v \preceq \mathrm{last}(f)\}\backslash\mathrm{dom}(f)$ as *holes*.

▶ **Lemma 23.** *Let π be a relaxing QU-res proof of Φ_n from an axiom set of the form $H(\Phi_n, \Pi_t)$. Each sink of $G(\pi)$ has a label f having at most one hole.*

▶ **Theorem 24.** *Suppose that $t \geqslant 2$ and that $n \geqslant \lceil t/2 \rceil$. Let π be a QU resolution proof of Φ_n from the axiom set $H(\Phi_n, \Pi_t)$. The graph $G(\pi)$ has at least $2^{n-\lceil t/2 \rceil - 1}$ sinks.*

From the previous theorem, we immediately obtain the following.

▶ **Theorem 25.** *Relaxing QU-res requires proofs of size $\Omega(2^n)$ on the sentences $\{\Phi_n\}_{n \geqslant 1}$.*

Acknowledgements. The author thanks Stefan Mengel and Moritz Müller for useful comments on a draft of this article.

References

1. Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *J. Artif. Intell. Res. (JAIR)*, 40:353–373, 2011.
2. Valeriy Balabanov and Jie-Hong R. Jiang. Resolution proofs and skolem functions in QBF evaluation and applications. In *Computer Aided Verification – 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, pages 149–164, 2011.
3. Valeriy Balabanov, Magdalena Widl, and Jie-Hong R. Jiang. QBF resolution systems and their proof complexities. In *Theory and Applications of Satisfiability Testing – SAT 2014 – 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, pages 154–169, 2014.
4. Paul Beame, Henry A. Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res. (JAIR)*, 22:319–351, 2004.
5. Paul Beame and Toniann Pitassi. Propositional proof complexity: Past, present and future. *Bulletin of the EATCS*, 65:66–89, 1998.
6. Eli Ben-Sasson, Russell Impagliazzo, and Avi Wigderson. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, 2004.
7. Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. *J. ACM*, 48(2):149–169, 2001.
8. Marco Benedetti. skizzo: A suite to evaluate and certify QBFs. In *Automated Deduction – CADE-20, 20th International Conference on Automated Deduction, Tallinn, Estonia, July 22-27, 2005, Proceedings*, pages 369–376, 2005.
9. Olaf Beyersdorff, Leroy Chew, and Mikolas Janota. On unification of QBF resolution-based calculi. In *Mathematical Foundations of Computer Science 2014 – 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 81–93, 2014.
10. Olaf Beyersdorff, Leroy Chew, and Mikolás Janota. Proof complexity of resolution-based QBF calculi. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 76–89, 2015.
11. Olaf Beyersdorff, Leroy Chew, and Karteek Sreenivasaiah. A game characterisation of tree-like q-resolution size. *Electronic Colloquium on Computational Complexity (ECCC)*, 2014.
12. Maria Luisa Bonet, Juan Luis Esteban, Nicola Galesi, and Jan Johannsen. On the relative complexity of resolution refinements and cutting planes proof systems. *SIAM J. Comput.*, 30(5):1462–1484, 2000.
13. Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Information and Computation*, 117(1):12–18, 1995.

14 Hubie Chen. Beyond q-resolution and prenex form: A proof system for quantified constraint satisfaction. *CoRR*, abs/1403.0222, 2014.

15 Hubie Chen. Proof complexity modulo the polynomial hierarchy: Understanding alternation as a source of hardness. *CoRR*, abs/1410.5369, 2014.

16 Stephen A. Cook and Robert A. Reckhow. On the lengths of proofs in the propositional calculus (preliminary version). In *Proceedings of the 6th Annual ACM Symposium on Theory of Computing, April 30 – May 2, 1974, Seattle, Washington, USA*, pages 135–148, 1974.

17 Uwe Egly, Florian Lonsing, and Magdalena Widl. Long-distance resolution: Proof generation and strategy extraction in search-based QBF solving. In *Logic for Programming, Artificial Intelligence, and Reasoning – 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, pages 291–308, 2013.

18 Allen Van Gelder. Contributions to the theory of practical quantified boolean formula solving. In *Principles and Practice of Constraint Programming – 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings*, pages 647–663, 2012.

19 Alexandra Goultiaeva, Allen Van Gelder, and Fahiem Bacchus. A uniform approach for generating proofs and strategies for both true and false QBF formulas. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 546–553, 2011.

20 Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.

21 Marijn Heule, Martina Seidl, and Armin Biere. A unified proof system for QBF preprocessing. In *Automated Reasoning – 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, pages 91–106, 2014.

22 Mikolás Janota, Radu Grigore, and João Marques-Silva. On QBF proofs and preprocessing. In *Logic for Programming, Artificial Intelligence, and Reasoning – 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, pages 473–489, 2013.

23 Mikolás Janota and Joao Marques-Silva. On propositional QBF expansions and Q-resolution. In *SAT*, pages 67–82, 2013.

24 Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-sat (preliminary version). In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 128–136, 2000.

25 Horst Samulowitz and Fahiem Bacchus. Using SAT in QBF. In *Principles and Practice of Constraint Programming – CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, pages 578–592, 2005.

26 N. Segerlind. The complexity of propositional proofs. *Bull. Symbolic Logic*, 13:417–626, 2007.

27 Yinlei Yu and Sharad Malik. Validating the result of a quantified boolean formula (QBF) solver: theory and practice. In *Proceedings of the 2005 Conference on Asia South Pacific Design Automation, ASP-DAC 2005, Shanghai, China, January 18-21, 2005*, pages 1047–1051, 2005.

Past, Present, and Infinite Future

Thomas Wilke

Kiel University, Kiel, Germany
thomas.wilke@email.uni-kiel.de

Abstract

I was supposed to deliver one of the speeches at Wolfgang Thomas's retirement ceremony. Wolfgang had called me on the phone earlier and posed some questions about temporal logic, but I hadn't had good answers at the time. What I decided to do at the ceremony was to take up the conversation again and show how it could have evolved if only I had put more effort into answering his questions. Here is the imaginary conversation with Wolfgang.

The contributions are (1) the first direct translation from counter-free ω-automata into future temporal formulas, (2) a definition of bimachines for ω-words, (3) a translation from arbitrary temporal formulas (including both, future and past operators) into counter-free ω-bimachines, and (4) an automata-based proof of separation: every arbitrary temporal formula is equivalent to a boolean combination of pure future, present, and pure past formulas when interpreted in ω-words.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.2 Modes of Computation, F.4.1 Mathematical Logic, F.4.3 Formal Languages

Keywords and phrases linear-time temporal logic, separation, backward deterministic ω-automata, counter freeness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.95

1 Act I

Wolfgang is sitting at his desk. Thomas is standing in his office, looking over the bay. They are talking to each other on the phone.

Wolfgang. I am teaching a course on applied automata theory this semester, and I would like to explain to my students how one can translate a counter-free ω-automaton into a temporal formula. I took a look at your STACS paper from 1999 [17], but the translation you give there is only for finite words. How does the whole story go for infinite words?

Thomas. I don't know of any published translation which comes close to what I present in the STACS paper. There is Volker and Paul's comprehensive contribution to the volume that celebrated your 60th birthday [4], but the construction presented therein is probably not what you are looking for, given its algebraic nature.

Wolfgang. I know what Volker and Paul did. Indeed, I am looking for something which is more automata-theoretic.

Thomas. I may have a suggestion for you.

Wolfgang. So?

Thomas. First of all, we need to choose the right ω-automaton model. Imagine you wanted to translate a future temporal formula over finite words into a finite-state automaton. Which model of automaton would you use?

Wolfgang. When I use ordinary automata, which read a word from left to right, I end up with a nondeterministic automaton, because the automaton can only guess what will happen in the future. When I use backward automata, which read a word from right to left,

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 95; pp. 95:1–95:14


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

I end up with a deterministic automaton right away, simply because what happens in the future can easily be determined when coming back from it.

Thomas. That's the point. When translating counter-free ω-automata into temporal formulas, we start out best from backward deterministic ω-automata, the model introduced by Olivier and Max Michel in their 1999 paper [3]. Do you recall how these automata are defined?

Wolfgang. Sure, I do. I think Jean-Éric and Dominique call them "prophetic" automata in their book [11], but backward deterministic ω-automata is also a good term. Anyway, such an automaton is given by a finite state set Q, an initial recurrence condition $\mathcal{I}$, a backward transition function $\rho\colon \Sigma \times Q \to Q$, and a set $F \subseteq Q$ of final states. It is required that for every ω-word over Σ there is exactly one initial run.

Thomas. Which recurrence conditions do you have in mind?

Wolfgang. Nothing in particular. How about you?

Thomas. Generalized transition Büchi conditions will come in handy. With such a condition, $\mathcal{I}$ is a set of sets $T \subseteq \Sigma \times Q$, each of them referred to as a transition recurrence set. A run $q_0q_1q_2\ldots$ on a word $w \in \Sigma^\omega$ needs to satisfy $q_i = \rho(w(i), q_{i+1})$ for every $i < \omega$. For such a run to be initial it is required that for every transition recurrence set $T \in \mathcal{I}$ there are infinitely many i with $\langle w(i), q_{i+1}\rangle \in T$. It is final if $q_0 \in F$, and it is accepting if it is initial and final, just as usual, only that we are going the other direction.

Wolfgang. OK. So we agree on the automaton model to be used.

Thomas. The next thing we need to agree on is what "counter freeness" should mean for such automata.

Wolfgang. To me, there seems to be a straightforward definition. For every finite word $w \in \Sigma^*$ we consider the function ${}^\rho w\colon Q \to Q$ induced by w on the state space. This is defined as usual, that is, ${}^\rho w(q) = {}^*\rho(w, q)$ for every $q \in Q$, where ${}^*\rho$ is the backward transition function extended from Σ to Σ^*. We say the automaton has a counter if there exists some word $w \in \Sigma^+$ and some nonempty subset $Q' \subseteq Q$ such that ${}^\rho w$ operates as a non-trivial permutation on Q'. More precisely, ${}^\rho w|_{Q'}$ is a bijection and ${}^\rho w|_{Q'} \neq \mathrm{id}_{Q'}$. If there is no counter, the automaton is counter-free.

Thomas. You are perfectly right. There is no difference to what we know from finite words, and the definition coincides with Volker and Paul's definition for Büchi automata in general.

Wolfgang. What I don't see right away is that the definition really captures the essence of counter freeness in general. I would like to understand this, but, maybe, it is better to return to this later.

Thomas. Promised. – Let's start to work on a translation from counter-free backward deterministic ω-automata to future temporal formulas. I suggest we work with the usual vocabulary for temporal logic. For every symbol a in the alphabet, we have an atomic formula a, which is true in a word if the word starts with a. We may use the temporal operator "next", which we write as X, and the stutter-free "until", which we write as U. We may also use derived operators such as "eventually" and "always", which we write as F and G, respectively. Finally, boolean constants and operators are allowed as well.

Wolfgang. I recall that your translation from counter-free automata to temporal logic is defined by induction, on the number of states of the given automaton in the first place and the size of the alphabet in the second place. Do we proceed in the same fashion for ω-words?

Thomas. Yes, exactly. Recall that for every ω-word there is exactly one initial run of a given backward deterministic ω-automaton on this word. Let's call the first state of this

run the halting state of the word. The inductive claim is that for every $q \in Q$, there exists a temporal formula defining the set of words with halting state q. – Wolfgang, things are getting more technical now. Let's use a desktop sharing software.

Thomas walks to his laptop. Wolfgang, already sitting in front of his screen, initiates a session between the two of them.

Thomas. I was saying that for every $q \in Q$, we construct a temporal formula $\alpha[q, \rho, \mathcal{I}]$, which defines the set of ω-words with halting state q. For convenience, let's write $L_\omega(q, \rho, \mathcal{I})$ for this language.

Wolfgang. I also recall you proceed by a simple case distinction. The almost trivial case is that all symbols induce the identity, which would mean ${}^\rho a = \mathrm{id}_Q$ for every $a \in \Sigma$.

Thomas. We proceed by the same case distinction here. If all symbols induce the identity function, the translation is simple and does not need the inductive assumption, but it is slightly more complicated than for finite words, because we are using generalized transition Büchi conditions.

Wolfgang. What exactly do you mean?

Thomas. When every symbol induces the identity, then every run is of the form q^ω for some $q \in Q$. Whether or not such a run is initial for a given word depends on the symbols occurring infinitely often in the word.

Wolfgang. That's funny. My first thought was that this case is really trivial.

Thomas. It is almost trivial, because for every set $\Sigma' \subseteq \Sigma$ the formula

$$\bigwedge_{a \in \Sigma'} \mathrm{GF}a \wedge \bigwedge_{a \in \Sigma \setminus \Sigma'} \mathrm{FG}\neg a$$

specifies exactly the ω-words where Σ' is the set of symbols occurring infinitely often.

Wolfgang. That's indeed almost trivial. *(Smiling.)* Let me understand what is going on in the more complicated case, when there is some symbol $c \in \Sigma$ such that ${}^\rho c$ is not the identity, that is, the image of ${}^\rho c$ is a strict subset of Q.

Thomas. Let's say this image is Q', and let's refer to $\Sigma \setminus \{c\}$ by Γ.

Wolfgang. In the proof for finite words, you split up each word in the positions where c occurs. If we do this here as well, there are three cases to distinguish: *i.* words which belong to Γ^ω, *ii.* words which belong to $\Sigma^* c \Gamma^\omega$, and *iii.* words which belong to $(\Gamma^* c)^\omega$.

Thomas. That's exactly right. We can deal with these three cases separately, because if we have a formula for each of these cases, their disjunction is the formula we are looking for. Case i is almost straightforward. We restrict the given backward transition function ρ to the smaller alphabet Γ, say the result is ρ', and the induction hypothesis applies right away. We obtain a formula $\alpha[q, \rho', \mathcal{I}]$ for $L_\omega(q, \rho', \mathcal{I})$, and we can set

$$\alpha[q, \rho, \mathcal{I}] = \mathrm{G}\neg c \wedge \alpha[q, \rho', \mathcal{I}] \ .$$

In other words, we take what we get from the induction hypothesis and rule out the symbol c.

Wolfgang. Why do we need to rule out c?

Thomas. Here is a simple example. Assume our alphabet was $\{a, b, c\}$ and the formula we obtained by induction was the formula a. Then ac^ω would be a model of the formula a, but this word does not belong to Γ^ω.

Wolfgang. Interesting, but I also have another question. Isn't it true that when we restrict ρ to ρ' as above we also need to restrict $\mathcal{I}$ appropriately, because in $\mathcal{I}$ there might be sets T with elements $\langle c, q \rangle$, but c does not belong to the underlying alphabet?

Thomas. Strictly speaking, you are right. That's why we should agree, once and for all, that we implicitly restrict recurrence conditions to the symbols occurring in the respective transition function.

Wolfgang. OK. – I think I know how to proceed in Case *ii*. Every word in $\Sigma^* c \Gamma^\omega$ can be broken up in a unique fashion. It can be written as ucv where $v \in \Gamma^\omega$. For the first part, u, we use what we know from finite words, and for the second part, v, we use the induction hypothesis.

Thomas. That's exactly right, but let's make it more precise. Let's write $L_*(q, \rho, q')$ for the language recognized by the backward deterministic automaton on finite words with initial state q', backward transition function ρ, and final state q. Then the language we are interested in, $L_\omega(q, \rho, \mathcal{I}) \cap \Sigma^* c \Gamma^\omega$, is the finite union of all languages

$$L_*(q, \rho, \rho(c, q'))\, c\, L_\omega(q', \rho', \mathcal{I}) \ , \tag{1}$$

for q' ranging over Q.

Wolfgang. Because of what we know about finite words, we have a temporal formula $\beta[q, \rho, q']$ for each language $L_*(q, \rho, q')$. And because of the induction hypothesis, we have a temporal formulas $\alpha[q', \rho', \mathcal{I}]$ for each language $L_\omega(q', \rho', \mathcal{I})$.

Thomas. Right! – A formula for a language as above, as given in (1), is therefore given by

$$\mathrm{xt}(\beta[q, \rho, \rho(c, q')]) \wedge \mathrm{F}(c \wedge \mathrm{XG}\neg c \wedge \mathrm{X}\alpha[q', \rho', \mathcal{I}]) \ ,$$

where $\mathrm{xt}(\beta)$ extends β to ω-words.

Wolfgang. I think I know what you mean by "extending" β to ω-words. You mean that for each $u \in \Sigma^*$ and $v \in \Gamma^\omega$, the word ucv is a model of $\mathrm{xt}(\beta)$ if, and only if, u is a model of β.

Thomas. You are perfectly right. – It is easy to obtain $\mathrm{xt}(\beta)$ from β by an inductive construction. The only interesting parts are the base case for a symbol and the induction steps involving temporal operators, because then the formula may "look" beyond the last occurrence of c, what is to be avoided:

$$\begin{aligned} \mathrm{xt}(a) &= a \wedge \mathrm{XF}c \ , \\ \mathrm{xt}(\mathrm{X}\psi) &= \mathrm{X}(\mathrm{xt}(\psi) \wedge \mathrm{F}c) \ , \\ \mathrm{xt}(\psi_0 \mathrm{U} \psi_1) &= \mathrm{xt}(\psi_0)\mathrm{U}(\mathrm{xt}(\psi_1) \wedge \mathrm{F}c) \ . \end{aligned}$$

Wolfgang. So the really interesting case is the last one, Case *iii*, when we consider words which belong to $(\Gamma^* c)^\omega$. Do you use an encoding trick like the one you use for finite words?

Thomas. Yes, the construction is very similar to the one in the finite-word setting, but considerably trickier. Let w be any word in $(\Gamma^* c)^\omega$. We can write w as $v_0 c v_1 c v_2 c \ldots$ where $v_i \in \Gamma^*$ for every $i < \omega$. There is exactly one initial run for w, say $q_0 q_1 q_2 \ldots$. Consider the subsequence $q_{i_0} q_{i_1} q_{i_2} \ldots$ which collects the states the automaton assumes just left of any c.

Wolfgang. Because we assume the image of ${}^\rho c$ is Q', these states are special in the sense that each of them belongs to Q'. So in some sense, each of the cv_i's transforms some state from Q' to some state from Q'.

Thomas. That's it! – Basically, we classify each word in $c\Gamma^*$ according to how it operates on Q', more precisely, to every such word w we assign the function $\tilde{w} \colon Q' \to Q'$ defined by $\tilde{w}(q) = {}^*\rho(w, q)$ for every $q \in Q'$.

Wolfgang. I understand that, but what I am concerned about is that we lose information about the recurrence condition!

Thomas. That's why I said "basically". We do a little bit more. We not only assign $\tilde{w}$ to w but also $\dot{w}$, a function $Q' \to 2^{\mathcal{I}}$, which collects the information about the recurrence condition we need.

Wolfgang. I can guess how $\dot{w}$ is defined. For every state $q \in Q'$ we consider the backward run of A – viewed as a backward automaton on finite words – on w starting in q and collect in $\dot{w}(q)$ all transition recurrence sets it passes through.

Thomas. Exactly, that's how we do it. – The code alphabet, let's call it Δ, is large. For each finite word $w \in c\Gamma^*$ it contains the symbol $\langle \tilde{w}, \dot{w} \rangle$.

Wolfgang. How do we construct the backward deterministic ω-automaton, let's call it C, which uses this alphabet?

Thomas. Its state space is Q', which is smaller than the state space of A. Its backward transition function – let's denote it τ – is defined by $\tau(\langle f, g \rangle, q') = f(q')$ for every $q' \in Q'$. For each transition Büchi set $T \in \mathcal{I}$, the transition Büchi set $\mathcal{J}$ of C has a Büchi set T', which is given by $T' = \{\langle \langle f, g \rangle, q' \rangle \mid T \in g(q')\}$. Most importantly, C is a counter-free backward deterministic ω-automaton as defined above.

Wolfgang. I immediately see how C mimics A. Let's define a function $h_0 \colon c\Gamma^* \to \Delta$ by $h_0(w) = \langle \tilde{w}, \dot{w} \rangle$ and use this to define a function $h \colon (c\Gamma^*)^\omega \to \Delta^\omega$ by setting $h(cw_0cw_1 \dots) = h_0(cw_0)h_0(cw_1) \dots$ for any choice of $w_i \in \Gamma^*$. Then, for every word $w \in (c\Gamma^*)^\omega$, the image $h(w)$ has halting state q' in C if, and only if, w has halting state q' in A. Or, formally, $L_\omega(q, \rho, \mathcal{I}) \cap (\Gamma^* c)^\omega$ is the finite union of all languages

$$L_*(q, \rho', q')\, h^{-1}(L_\omega(q', \tau, \mathcal{J})) \ , \tag{2}$$

where q' ranges over all states in Q'.

Thomas. That's right! – Did you observe where the generalized transition Büchi condition came in handy?

Wolfgang. Yes, I did. If we had used some other condition, transferring it from A to C could have easily blown up the state space of C.

Thomas. So you see how we can use the induction hypothesis!?

Wolfgang. Sure. We can apply it to C, because C has a smaller state space than A, and obtain, for every $q' \in Q'$, a temporal formula $\alpha[q', \tau, \mathcal{J}]$ for $L_\omega(q', \tau, \mathcal{J})$, the alphabet being Δ.

Thomas. And do you see where the results from the finite-word setting come into the picture?

Wolfgang. Sure. They are used in two different places. First, for every $q \in Q$ and every $q' \in Q'$ there is a temporal formula $\beta[q, \rho, q']$ that defines $L_*(q, \rho', q')$. Second, for every symbol $\langle f, g \rangle \in \Delta$ there is a temporal formula $\chi_{f,g}$ such that a word $w \in \Gamma^*$ is a model of $\chi_{f,g}$ if, and only if, $h_0(cw) = \langle f, g \rangle$.

Thomas. Precisely! – What is left to be done is to assemble all these formulas in the right fashion.

Wolfgang. I can take over the first programming task. We need to transform every formula φ over Γ into a formula $\mathrm{xt}'(\varphi)$ over Σ such that for all $u \in \Gamma^*$ and $v \in \Sigma^\omega$, the formula φ is a model of u if, and only if, the formula $\mathrm{xt}'(\varphi)$ is a model of ucv. The crucial

definitions are:

$$\begin{aligned} \mathrm{xt}'(a) &= a \ , \\ \mathrm{xt}'(\mathrm{X}\psi) &= \neg c \wedge \mathrm{X}\,\mathrm{xt}'(\psi) \ , \\ \mathrm{xt}'(\psi_0 \mathrm{U} \psi_1) &= (\mathrm{xt}'(\varphi) \wedge \neg c)\mathrm{U}\,\mathrm{xt}'(\psi) \ . \end{aligned}$$

Thomas. Second, we need to transform every formula φ over Δ into a corresponding formula $\mathrm{lft}(\varphi)$, which "lifts" the formula from the alphabet Δ to the alphabet Σ, more precisely, for each $u \in (c\Sigma^*)^\omega$, the word u is a model of $\mathrm{lft}(\varphi)$ if, and only if, $h(u)$ is a model of φ. Here, we can set:

$$\begin{aligned} \mathrm{lft}(\langle f, g \rangle) &= \mathrm{xt}'(\chi_{f,g}) \ , \\ \mathrm{lft}(\mathrm{X}\varphi) &= \mathrm{X}(\neg c \mathrm{U}(c \wedge \mathrm{X}\,\mathrm{lft}(\varphi))) \ , \\ \mathrm{lft}(\varphi \mathrm{U} \psi) &= (c \to \mathrm{X}\,\mathrm{lft}(\varphi))\mathrm{U}(c \wedge \mathrm{X}\,\mathrm{lft}(\psi)) \ . \end{aligned}$$

Wolfgang. We are done. The overall formula for a language as in (2) is

$$\mathrm{GF}c \wedge \mathrm{xt}'(\beta[q, \rho', q']) \wedge \neg c \mathrm{U}(c \wedge \mathrm{lft}(\alpha)[q', \tau, \mathcal{J}]) \ .$$

Thomas, I need to run. There's a meeting I have to attend ...

Wolfgang grabs a pile of documents from a bookshelf and leaves his room in a hurry.

2 Act II

Wolfgang, wearing a headset, and Thomas are sitting at their desks. On Thomas's screen, a notification from the desktop sharing software pops up. Wolfgang is trying to connect. Thomas puts on his headset.

Thomas. Wolfgang?

Wolfgang. Thomas! Can I come back to the discussion on temporal logic and counter-free automata we has the other day?

Thomas. Sure. What is it that you would like to talk about?

Wolfgang. You said that your definition of "counter-free ω-automaton" is a good one; you even said it would be the right one. Can you explain that to me?

Thomas. From finite words we know that there are many equivalent definitions of what it means for a formal language to be star-free: recognizable by a counter-free automaton [13], definable in first-order logic [10], definable in temporal logic (with future and past operators) [8], definable in future temporal logic [7], ... For ω-words, the same equivalences hold, in particular, first-order logic, the two variants of temporal logic, and star-free expressions are equally expressive [9, 15, 8, 7]. Do you recall this? *(Smiling.)* Anyway, any notion of "counter freeness" that is equivalent to one of these formalisms should be ok.

Wolfgang. I agree! – What we already know is that every counter-free automaton according to your definition is equivalent to a future temporal logic formula. So if you can also prove the converse to me, I will be happy.

Thomas. To tell you the truth, parts of this were already proved a long time ago, but went unnoticed. What I mean is that the straightforward translation of a future temporal formula into a generalized Büchi automaton yields a counter-free backward deterministic

ω-automaton. In fact, for all I know, this observation was the motivation for defining backward deterministic ω-automata [1] and you can find the translation, for instance, in a paper by Olivier [2]. – The problem is to show that this translation yields a counter-free automaton.

Wolfgang. *(Indignant, but smiling.)* Thomas! – That we obtain backward deterministic Büchi automata when translating future temporal formulas into automata is something I have known for years, in fact, I explained this to one of the anonymous referees of this paper already 15 years ago. But I didn't look at counter freeness at the time I have to admit. So let's see what happens when we follow a construction like the one Pierre and Moshe suggested [16]?

Thomas. A typical automaton for a temporal formula guesses, for each point in time, which subformulas are true at that point and which are not.

Wolfgang. So we model a state as a function $f\colon \mathrm{sbf}(\varphi) \to \{0,1\}$, where $\mathrm{sbf}(\varphi)$ stands for the set of subformulas of φ. The functions considered are required to satisfy the following straightforward conditions: $f(\top) = 1$; $f(\neg\psi) = 1$ if, and only if, $f(\psi) = 0$; $f(\psi_0 \vee \psi_1) = 1$ if, and only if, $f(\psi_0) = 1$ or $f(\psi_1) = 1$. Here, ψ and $\psi_0 \vee \psi_1$ stand for elements of $\mathrm{sbf}(\varphi)$. The transition relation, let's denote it by Δ, is defined according to the semantics of temporal logic, more precisely, $(f, a, g) \in \Delta$ if, and only if, the following conditions are satisfied:

- $f(a) = 1$,
- $f(b) = 0$ for every symbol $b \in \Sigma$ with $b \neq a$,
- $f(\mathrm{X}\psi) = g(\psi)$,
- $f(\psi_0 \mathrm{U} \psi_1) = 1$ if, and only if, $f(\psi_1) = 1$ or, $f(\psi_0) = 1$ and $g(\psi_0 \mathrm{U} \psi_1) = 1$.

Just as above, $\mathrm{X}\psi$ and $\psi_0 \mathrm{U} \psi_1$ stand for elements of $\mathrm{sbf}(\varphi)$. And, of course, the transition relation Δ is backward deterministic, that is, $\rho(a, g) = f$ for $(f, a, g) \in \Delta$ is well-defined.

Thomas. What about the recurrence condition?

Wolfgang. We need to make sure that when a U-formula is guessed to be true it becomes true eventually. To this end, we use a generalized state Büchi recurrence condition, which can easily be transformed into a generalized transition Büchi condition. For every U-subformula, say $\psi_0 \mathrm{U} \psi_1$, we have the set

$$\{f \mid f(\psi_0 \mathrm{U} \psi_1) = 0 \text{ or } f(\psi_1) = 1\}$$

as an element of the recurrence condition. – So how do we know this automaton is counter-free?

Thomas. This needs a proof indeed. – Suppose $Q' \subseteq Q$ is a nonempty subset of the state space and ${}^{\rho}w$ restricted to Q' is a permutation for some word $w \in \Sigma^+$. We want to show ${}^{\rho}w(f) = f$ for every state $f \in Q'$. One way to do this is to fix an element $f_0 \in Q'$ and consider the orbit of f_0.

Wolfgang. What do you mean by "orbit"?

Thomas. Define f_{i+1} by $f_{i+1} = {}^{\rho}w(f_i)$ for $i < \omega$. Then, because we assume ${}^{\rho}w$ restricted to Q' is a permutation, $f_m = f_0$ for some $m > 0$. The set $\{f_m \mid i < m\}$ is what we call the orbit of f_0.

Wolfgang. I see. If we can prove $f_1 = f_0$, or, equivalently, $f_i = f_{i'}$ for all $i, i' < m$, we are done, because this means ${}^{\rho}w(f_0) = f_0$.

Thomas. Exactly. By induction, we show $f_i(\psi) = f_{i'}(\psi)$ for every $\psi \in \mathrm{sbf}(\varphi)$ and all $i, i' < m$, which is sufficient. In fact, we prove something stronger.

Wolfgang. I have no idea what that could be.

Thomas. We refine the picture in an adequate fashion by introducing more states and making it cyclic. Let $n = |w|$. First, we extend w in both directions by repeating it, that is,

we consider the sequence $(a_i)_{i\in\mathbf{Z}}$ defined by $a_{i+kn} = w(i)$ for all $i < n$ and $k \in \mathbf{Z}$ with $k \neq 0$. Second, we define, for every $i \in \mathbf{Z}$, a state g_i by

$$\begin{aligned} g_0 &= f_0 \ , \\ g_i &= {}^{\rho}a_i(g_{i+1}) && \text{for } i \text{ with } -mn < i < 0 \ , \\ g_{i+kmn} &= g_i && \text{for } i \text{ with } -mn < i \leq 0 \text{ and } k \neq 0 \ . \end{aligned}$$

Then $g_{-ni} = f_i$ for every $i \in \mathbf{N}$ and $g_i = {}^{\rho}a_i(g_{i+1})$ for all $i \in \mathbf{Z}$. Furthermore, $g_i(\psi) = g_{i'}(\psi)$ for all $i, i' \in \mathbf{N}$ with $i \equiv i' \,(mn)$ and $\psi \in \mathrm{sbf}(\varphi)$. I write $i \equiv i' \,(l)$ to denote the fact that i and i' are identical after reduction modulo l. The stronger claim is that $g_i(\psi) = g_{i'}(\psi)$ holds for all $i, i' \in \mathbf{Z}$ with $i \equiv i' \,(n)$ and $\psi \in \mathrm{sbf}(\varphi)$.

Wolfgang. I see, this contains the original claim as a special case. – I am curious to see how the inductive proof goes.

Thomas. For $\psi = \top$, the claim is trivial. For the other base case, assume $a \in \Sigma$ and $\psi = a$. We have $g_i(a) = 1$ if, and only if, $a = a_i$, which proves the claim, because if $i \equiv i' \,(n)$, then $a_i = a_{i'}$, by definition of $(a_i)_{i\in\mathbf{Z}}$.

Wolfgang. Now it's my turn. First, the claim is trivial for the boolean connectives $\neg$ and $\vee$. Assume $\mathrm{X}\psi \in \mathrm{sbf}(\varphi)$. Then $g_i(\mathrm{X}\psi) = g_{i+1}(\psi)$ for every $i \in \mathbf{Z}$, because of the definition of Δ. By induction hypothesis, we have $g_{i+1}(\psi) = g_{i'+1}(\psi)$ for all i and i' with $i \equiv i' \,(n)$, which then implies the claim.

Thomas. Let me conclude the proof by looking at the case where $\psi_0 \mathrm{U} \psi_1 \in \mathrm{sbf}(\varphi)$. It is good to proceed by a case distinction.

Wolfgang. I can imagine what the cases are.

Thomas. The first case is when $g_i(\psi_0) = 1$ holds for all $i \in \mathbf{Z}$. If $g_{i_0}(\psi_0 \mathrm{U} \psi_1) = 1$ for some i_0, then, by definition of Δ, $g_i(\psi_0 \mathrm{U} \psi_1) = 1$ for all $i \leq i_0$. Since $g_{i_0}(\psi_0 \mathrm{U} \psi_1) = 1$ means $g_{i_0+kmn}(\psi_0 \mathrm{U} \psi_1) = 1$ for all $k \in \mathbf{N}$, we even have $g_i(\psi_0 \mathrm{U} \psi_1) = 1$ for all $i \in \mathbf{Z}$.

The second case is when $g_{i_0}(\psi_0) = 0$ holds for some $i_0 \in \mathbf{Z}$. In this case, $g_{i_0+kmn}(\psi_0) = 0$ for all $k \in \mathbf{Z}$. So if $g_i(\psi_0 \mathrm{U} \psi_1) = 1$ for some $i \in \mathbf{Z}$, then, because of the definition of Δ, there exists some $l \in \mathbf{N}$ with $g_{i+l}(\psi_1) = 1$ and $g_{i+k}(\psi_0) = 1$ for all k with $k < l$ – a simple proof by induction shows that. Let i' be such that $i \equiv i' \,(n)$. Then $i + j \equiv i' + j \,(n)$ for every $j \in \mathbf{Z}$. So from the induction hypothesis, we can conclude $g_{i'+l}(\psi_1) = 1$ and $g_{i'+k}(\psi_0) = 1$ for all k with $k < l$. Hence, $g_{i'}(\psi_0 \mathrm{U} \psi_1) = 1$.

Wolfgang. We are done. Perfect!

Thomas. Wolfgang, my theory lecture starts in a few minutes. I hope you don't mind me hanging up now. I am calling back later.

Wolfgang. I am sorry. Please, go ahead.

Thomas grabs his tablet, a copy of Sipser's book [14], and leaves his room; Wolfgang takes a notebook and starts scribbling.

3 Act III

Wolfgang is still sitting at his desk, contemplating. Thomas just entered his office, carrying his tablet and Sipser's book, and went straight to his laptop. He is initiating another session with Wolfgang.

Wolfgang. Thomas, thanks for calling back. I hope your lecture went well. Here is what I thought about in the meantime. Now that we know how to deal with counter-free ω-automata and future temporal logic, can we also say something about temporal logic in general? When future *and* past operators are allowed?

Thomas. Why are you asking? – We somehow know what happens because of the result by Gabbay, Pnueli, Shelah, and Stavi [7].

Wolfgang. I know this result settles the question in the sense that every arbitrary temporal formula is equivalent to a future temporal formula when interpreted in the first position of ω-words. I also know that by a result by Gabbay [6] every arbitrary temporal formula is equivalent to a boolean combination of pure past, present, and pure future formulas. But proofs of these results are technically involved and I am missing automata theory in this picture!

Thomas. Oh, that's an interesting thought.

Thomas thinking ...

Thomas. I think we can say something really nice here. Again, we need to agree on the right automaton model.

Wolfgang. What are you thinking of?

Thomas. Well, arbitrary temporal logic formulas are interpreted in some position in an ω-word. In some sense, the semantics of such a formula is a function $[\![\varphi]\!]$ which maps ω-words over Σ to ω-words over $\{0,1\}$, where the bit at position i of the image of a given word is the truth value of the formula when interpreted at position i in the given word. So what I think we should do is to come up with a slick automaton model for describing such functions.

Wolfgang. There are many ways to define functions from ω-words to ω-words and even more ways to define functions from finite words to finite words. A very general notion is that of a rational function and it has been studied in detail. For instance, there is a result by Elgot and Mezei which states that a rational function of finite words is the composition of a left-sequential and a right-sequential function [5], and I believe I have come across a similar result for ω-words.

Thomas. That is true. Elgot and Mezei's result was generalized to ω-words by Olivier [2]. I am thinking of extending Schützenberger's bimachines [12] to ω-words in a way adequate for dealing with temporal logic.

Wolfgang. I remember Schützenberger's bimachines. How do we extend them to ω-words?

Thomas. Suppose we want to realize a function $\nu\colon \Sigma^\omega \to \Gamma^\omega$. Suppose we are given a word $w \in \Sigma^\omega$. And suppose we want to know the symbol at position i in $\nu(w)$, that is, we want to know $\nu(w)(i)$. What we could do first is to split w at position i, that is, write w as uav where the length of u is i. Then we could run a forward deterministic automaton on u, a backward deterministic ω-automaton on v, and output $\nu(w)(i)$ depending on the two halting states and the symbol a. – Here is a picture. *(Drawing on the screen.)*

Wolfgang. That is very close to what Schützenberger suggested for finite words, only that he allowed an arbitrary word to be output, rather than a single symbol. – Let me try to make your picture formal and define what we may call ω-bimachines. Such a machine, let's call it A, consists of

- a forward deterministic automaton without final condition, say with finite state set Q, initial state q_I, and transition function $\delta\colon Q \times \Sigma \to Q$,
- a backward deterministic ω-automaton without final condition, say with finite state set S, initial recurrence condition $\mathcal{I}$, and a backward transition function $\rho\colon \Sigma \times S \to S$, and
- an output function $o\colon Q \times \Sigma \times S \to \Gamma$.

Thomas. That's it. The semantics is a function $\Sigma^\omega \to \Gamma^\omega$, which we denote by ν. Assume $w \in \Sigma^\omega$ and we want to define $\nu(w)(i)$ for some $i \in \omega$. We take the halting state of the forward automaton for the word $w(0) \dots w(i-1)$, say this is q. We take the halting state of the backward automaton for the word $w(i+1)w(i+2)\dots$, say this is s. Then $\nu(w)(i) = o(q, w(i), s)$.

Wolfgang. That is indeed a very simple and intuitive definition. It also connects nicely with rational functions, because from Olivier's results it should follow that the class of functions computed by ω-bimachines is the same as the class of total letter-to-letter rational functions. – What we want to do is to transform every temporal formula into an equivalent ω-bimachine.

Thomas. We should even strive for a counter-free ω-bimachine, where this simply means that the forward and the backward automaton are counter-free. Because if we manage to achieve that, we also have a proof of the result by Gabbay, saying that every arbitrary temporal formula is equivalent to a boolean combination of pure future, present, and pure past formulas – completely in automata-theoretic terms.

Wolfgang. That sounds like an interesting plan and I would like to give it a shot. In the future-only setting, it was easy to describe the state space in one go – it was a function $\mathrm{sbf}(\varphi) \to \{0,1\}$. I believe it is going to be more complicated here, which is the reason I suggest we try an inductive definition.

Thomas. I will be happy with an inductive definition!

Wolfgang. There are two base cases: $\psi = \top$ and $\psi = a$ for some $a \in \Sigma$. In both cases, we can choose the forward and the backward automaton to be a 1-state automaton. In the first case, we can set $o(q, a, s) = 1$ for every $a \in \Sigma$; in the second case, we can set $o(q, a, s) = 1$ and $o(q, b, s) = 0$ for all $b \in \Sigma \setminus \{a\}$. – Easy!

Thomas. Clearly, these automata are counter-free.

Wolfgang. In the inductive step, we have to take care of boolean operators and temporal operators.

Thomas. Boolean operators can be dealt with easily, only the temporal operators are interesting.

Wolfgang. We have four temporal operators when we admit future as well as past operators and follow standard syntax and semantics: X – next, P – previously, U – until, and S – since. I suggest we consider "previously" and "until", the two other can be dealt with in a similar fashion.

Thomas. What we need for each of the two operators are two things. First, we need a construction. Second, we need a proof that it preserves counter freeness.

Wolfgang. So let's turn to "previously" and assume we are given a counter-free ω-bimachine which computes a function $\mu\colon \Sigma^\omega \to \{0,1\}^\omega$. We want to construct a new counter-free ω-bimachine which computes the function ν defined by $\nu(w)(i+1) = \mu(w)(i)$ and $\nu(w)(0) = 0$ for every $i < \omega$. Apparently, we don't have to change the backward automaton; we only need to adapt the forward automaton and the output function.

Thomas. I agree. This seems to be a simple construction.

Wolfgang. We make the forward automaton lag behind in the sense that it keeps track of its previous state and the symbol just read. In the beginning, we start from a new state. So

the state space is $Q \times \Sigma \cup \{\bot\}$ and the transition function is given by $\delta'(\langle q, a\rangle, b) = \langle \delta(q, a), b\rangle$ and $\delta'(\bot, a) = \langle q_I, a\rangle$. The output function, o', is given by $o'(\langle q, a\rangle, b, s) = o(q, a, \rho(b, s))$ and $o'(\bot, a, s) = 0$.

Thomas. This is quite convincing. In fact, the construction does not only seem to be correct to me. It also preserves counter freeness, as far as I can see.

Wolfgang. So let's turn to "until".

Thomas. We assume we are given two counter-free ω-bimachines computing functions $\mu_0 \colon \Sigma^\omega \to \{0,1\}^\omega$ and $\mu_1 \colon \Sigma^\omega \to \{0,1\}^\omega$, respectively, and we want to construct a counter-free ω-bimachine computing the function $\nu \colon \Sigma^\omega \to \{0,1\}^\omega$ given by $\nu(w)(i) = 1$ if, and only if, there exists some $k \geq i$ such that $\mu_1(w)(k) = 1$ and $\mu_0(w)(j) = 1$ for all j with $i \leq j < k$. – By taking a product of the two forward automata and the two backward automata, we can simplify the situation in the sense that we can think of only one counter-free ω-bimachine but with two output functions, o_0 and o_1, where the first one is for μ_0 and the second one is for μ_1. – Do you see what we need to do?

Wolfgang. Yes, right away. Let's call the given automaton A and the one to be constructed A'. It is important that the backward automaton of A' knows, at any point, those states from the forward automaton of A from which the U-formula can be satisfied. – I am aware of the fact that this is a vague description, but it should become clear when we work out the details.

Thomas. Let me see if I understand what you mean. The state space of the backward automaton of A' is $S \times 2^Q$. Its backward transition function is defined by

$$\rho'(a, \langle s, P\rangle) = \langle \rho(a, s), P'\rangle \ ,$$

where P stands for a subset of Q and P' is defined by

$$P' = \{q \in Q \mid o_1(q, a, s) = 1\} \cup \{q \in Q \mid \delta(q, a) \in P \text{ and } o_0(q, a, s) = 1\} \ .$$

The forward automaton of A' is the same as the one of A. The output function of A', let's denote it o', is given by $o'(q, a, \langle s, P\rangle) = 1$ if, and only if, $o_1(q, a, s) = 1$, or $o_0(q, a, s) = 1$ and $\delta(q, a) \in P$.

Wolfgang. This is what I was thinking of. The transition function ρ' reflects the semantics of the until operator in a particular sense, which I would like to make precise. For every state $q \in Q$, let A_q denote the ω-bimachine which is obtained from A by changing the initial state of the forward automaton to q. Further, let ν_0^q and ν_1^q denote the corresponding functions. Now suppose $w \in \Sigma^\omega$ and $\langle s_0, P_0\rangle\langle s_1, P_1\rangle \dots$ is a run of the backward automaton of A' on w such that $s_0 s_1 \dots$ is an initial run of the backward automaton of A and write P for P_0. Then the following is true for every state $q \in Q$ and can be proved by a straightforward induction:

1. If there is some j such that $\nu_1^q(j) = 1$ and $\nu_0^q(i) = 1$ for every $i < j$, then $q \in P$.
2. If $q \in P$, then either
   a. there is some j such that $\nu_1^q(j) = 1$ and $\nu_0^q(i) = 1$ for every $i < j$, or
   b. $\nu_0^q(i) = 1$ for all $i < \omega$.

The problem I see is that we really want 2.a for each $q \in P$. For a U-formula to be true, it is not enough to have 2.b only.

Thomas. Your concern is completely valid. Without any further measure, the construction may "overapproximate". The problem is similar to the fairness problem we had when we looked at the backward deterministic ω-automaton for a given future temporal formula and introduced a recurrence set for every U-subformula.

Wolfgang. What do you suggest? Are we going to do the same here?

THOMAS. The situation is more complicated. Here is the basic idea. For every q satisfying 2.a there is a smallest j with the specified property. Let's denote this by j_q. We assign a natural number $p(q)$ to every state $q \in P$ such that $p(q)$ reflects the order of the j_q's, that is, $p(q) < p(q')$ if, and only if, $j_q < j_{q'}$. We then make sure by appropriate recurrence conditions that the numbers that are assigned to the successors of q decrease over time until j_q is finally reached.

WOLFGANG. That sounds interesting. Can you make this precise?

THOMAS. Let $m = |Q|$ and set $M = \{0, \ldots, m-1, \infty\}$. A state of the backward automaton of A' is then a pair $\langle s, p \colon Q \to M \rangle$ where P from above is now given by $\{q \in Q \mid p(q) \neq \infty\}$.

WOLFGANG. Given this, I think I can describe how the correct backward deterministic transition function works. In a first step, we set

- $p_0(q) = -1$ for every q with $o_1(q, a, s) = 1$,
- $p_0(q) = p(\delta(q, a))$ for every q with $o_0(q, a, s) = 1$ and $o_1(q, a, s) = 0$, and
- $p_0(q) = \infty$ for all other $q \in Q$.

This is consistent with the definition of P' from above in the sense that $P' = \{q \in Q \mid p(q) \neq \infty\}$, but there are values out of range – we may have -1 as a value. We adjust p_0 to obtain p' by increasing some of its values as follows, where t is defined by $t = \min\{j \in \{-1, \ldots, m-1\} \mid p_0^{-1}(j) = \emptyset\}$.

- $p'(q) = p_0(q) + 1$ for all q with $p(q) < t$.
- $p'(q) = p_0(q)$ for all other $q \in Q$.

In some sense, we are adjusting as little as is necessary.

THOMAS. This is absolutely correct. What we still have to define is an appropriate recurrence condition. This will be the union of two recurrence sets $\mathcal{I}_0$ and $\mathcal{I}_1$, where $\mathcal{I}_0$ simply extends $\mathcal{I}$ to the new state space in a straightforward way. More precisely, $\mathcal{I}_0 = \{S' \times M^Q \mid S' \in \mathcal{I}\}$.

The set $\mathcal{I}_1$ contains a transition recurrence set T_i for every $i \in \{0, \ldots, m-1\}$. This set contains a pair $\langle a, \langle s, p \rangle \rangle$ if, and only if, $t = i$ for the value t as defined above or $p'(q) \in \{0, \ldots, i-1, \infty\}$ for every $q \in Q$ and p' as defined above.

WOLFGANG. Thomas, I understand what you are saying, but I think a rigorous correctness proof is needed here.

THOMAS. I agree. As the construction is inspired by Oliver and Max Michel's work, we can also borrow some of their ideas for the correctness proof.

WOLFGANG. Anyway, we also need to prove that our construction really yields a counter-free automaton.

THOMAS. I agree again. The proof of this is tedious, but doable. One can use what we know from the proof that the backward deterministic ω-automaton for a given future temporal formula is counter-free.

WOLFGANG. OK. There is still some work to be done before I can present the material to the students. I am calling it a day. Thanks a lot, and see you soon, Thomas!

THOMAS. Bye, bye. And, please, say "hello" to Renate.

Wolfgang and Thomas close their desktop sharing applications.

4 Epilogue

A few weeks later, Thomas receives the following email.

Hi, Thomas!

You probably want to know how it went with my lectures. First of all, it didn't take me much time to work out all the details of what we talked about. Then everything went fine, only the students came up with a fair number of questions I couldn't answer right away. Here are the most interesting ones, maybe:

- What is the complexity of the translation from counter-free backward deterministic ω-automata to future temporal formulas?
- What is the complexity of the translation from arbitrary temporal formulas to ω-bimachines? Non-elementary?
- Say "prop" is some interesting class of deterministic automata. What happens when we consider "prop" ω-bimachines instead of counter-free ω-bimachines?

Any ideas?

All the best, Wolfgang

References

1 Olivier Carton. Personal communication.

2 Olivier Carton. Right-sequential functions on infinite words. In Farid M. Ablayev and Ernst W. Mayr, editors, *Computer Science – Theory and Applications, 5th International Computer Science Symposium in Russia, CSR 2010, Kazan, Russia, June 16-20, 2010. Proceedings*, volume 6072 of *Lecture Notes in Computer Science*, pages 96–106. Springer, 2010. doi:10.1007/978-3-642-13182-0_9.

3 Olivier Carton and Max Michel. Unambiguous Büchi automata. In Gaston H. Gonnet, Daniel Panario, and Alfredo Viola, editors, *LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings*, volume 1776 of *Lecture Notes in Computer Science*, pages 407–416. Springer, 2000. doi:10.1007/10719839_40.

4 Volker Diekert and Paul Gastin. First-order definable languages. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

5 Calvin C. Elgot and Jorge E. Mezei. On relations defined by generalized finite automata. *IBM Journal of Research and Development*, 9(1):47–68, Jan 1965. doi:10.1147/rd.91.0047.

6 Dov M. Gabbay. The declarative past and imperative future: Executable temporal logic for interactive systems. In Behnam Banieqbal, Howard Barringer, and Amir Pnueli, editors, *Temporal Logic in Specification, Altrincham, UK, April 8-10, 1987, Proceedings*, volume 398 of *Lecture Notes in Computer Science*, pages 409–448. Springer, 1987.

7 Dov M. Gabbay, Amir Pnueli, Saharon Shelah, and Jonathan Stavi. On the temporal basis of fairness. In Paul W. Abrahams, Richard J. Lipton, and Stephen R. Bourne, editors, *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA, January 1980*, pages 163–173. ACM Press, 1980. URL: http://dl.acm.org/citation.cfm?id=567446, doi:10.1145/567446.567462.

8 Johan Anthony Willem Kamp. *Tense logic and the theory of linear order*. PhD thesis, University of California, Los Angeles, 1968.

9 Richard E. Ladner. Application of model theoretic games to discrete linear orders and finite automata. *Information and Control*, 33(4):281–303, 1977. doi:10.1016/S0019-9958(77)90443-0.

10 Robert McNaughton and Seymour Papert. *Counter-free automata*. M.I.T. Press research monographs. M.I.T. Press, 1971.

11 Dominique Perrin and Jean-Éric Pin. *Infinite Words: Automata, Semigroups, Logic, and Games*, volume 141 of *Pure and Applied Mathematics*. Elsevier, Amsterdam, 2004.

12 Marcel Paul Schützenberger. A remark on finite transducers. *Information and Control*, 4(2-3):185–196, 1961. doi:10.1016/S0019-9958(61)80006-5.

13 Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965. doi:10.1016/S0019-9958(65)90108-7.

14 Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, Boston, Mass., 3rd edition, 2013.

15 Wolfgang Thomas. Star-free regular sets of omega-sequences. *Information and Control*, 42(2):148–156, 1979. doi:10.1016/S0019-9958(79)90629-6.

16 Moshe Y. Vardi and Pierre Wolper. Reasoning about infinite computations. *Inf. Comput.*, 115(1):1–37, 1994. doi:10.1006/inco.1994.1092.

17 Thomas Wilke. Classifying discrete temporal properties. In Christoph Meinel and Sophie Tison, editors, *STACS 99, 16th Annual Symposium on Theoretical Aspects of Computer Science, Trier, Germany, March 4-6, 1999, Proceedings*, volume 1563 of *Lecture Notes in Computer Science*, pages 32–46. Springer, 1999. doi:10.1007/3-540-49116-3_3.

Thin MSO with a Probabilistic Path Quantifier

Mikołaj Bojańczyk

University of Warsaw, Warsaw, Poland

Abstract

This paper is about a variant of MSO on infinite trees where:

- there is a quantifier "zero probability of choosing a path $\pi \in 2^\omega$ which makes $\varphi(\pi)$ true";
- the monadic quantifiers range over sets with countable topological closure.

We introduce an automaton model, and show that it captures the logic.

1998 ACM Subject Classification F.4.1 Mathematical Logic

Keywords and phrases Automata, MSO, infinite trees, probabilistic temporal logics

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.96

1 Introduction

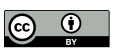
The ambient topic of this paper is MSO on infinite binary trees, extended by a quantifier $\mathsf{zero}\pi\ \varphi(\pi)$ which says that there is zero probability of choosing a path π in the tree so that $\varphi(\pi)$ is true. Here we assume that each bit (i.e. turn) in the path is chosen independently at random. This logic was introduced by Michalewski and Mio in [10], where the decidability of satisfiability was left open.

That satisfiability question is not solved here, but we make a small step in its direction. We consider a fragment of the logic, called TMSO+zero, standing for *thin* MSO+zero. In this fragment, the monadic set quantifiers are restricted to sets which are thin in the following sense: a set of nodes is *thin* if there are countably many paths which visit it infinitely often. For example, every path (when seen as a set of nodes) is thin, and every finite set is thin. In the logic TMSO+zero, one has existential and universal quantification over nodes and thin sets of nodes, as well as the probabilistic path quantifier zero. Being thin is definable in MSO, and therefore without the zero quantifier, the logic would be a special case of MSO, and with the zero quantifier it is a special case of the logic from [10].

The contribution of this paper is the definition of an automaton model, called zero automata, and a proof that every formula of TMSO+zero can be effectively translated to an equivalent zero automaton.

Motivation

The first source of motivation for this paper is the study of probabilistic temporal logics [1, 8, 13, 4]. An important example is the logic PCTL. It is an open problem whether this logic has decidable satisfiability. Much of the difficulty stems from the ability of talking about probabilities like 1/2 or 1/3. If one can only compare probabilities to 0 or 1, which is in the spirit of our logic TMSO+zero, then we get *qualitative* PCTL, whose satisfiability was shown decidable by Brázdil, Forejt, Kretínský and Kucera in [4]. Actually, the qualitative fragment of PCTL, as well as stronger qualitative logics like PCTL*, can be straightforwardly formalised in TMSO+zero, and therefore, by the main result of this paper, translated into zero automata. Another example that we discuss later in the paper is the probabilistic version of tree automata by Carayol, Haddad and Serre [6]; these are also a special case of zero automata.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 96; pp. 96:1–96:13

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

The second source of motivation is trying to find a robust classes of languages of infinite words or trees which remains decidable (e.g. with respect to satisfiability). The point of departure is MSO, with its famous decidability results by Büchi [5] and Rabin [11]. One way of departing from that point is to add unary predicates, e.g. extending MSO over ω-words by a predicate "x is a position of the form $n!$", see [12] for a survey. Another way is to add new quantifiers. Due to the strength of MSO, it is not so easy to come up with a quantifier extending MSO that is not obviously undecidable, and yet not already definable in MSO. For example, a nice quantifier is "there exist uncountably many sets with a given property" – but as shown in [2], this quantifier does not add to the expressive power of MSO. A logic that *does* properly extend MSO is MSO+U, which is an extension of MSO by a quantifier which can say that a given property is true for finite sets of unbounded size. The logic is itself undecidable, but has many decidable fragments, typically variants of weak MSO. See [3] for a survey of MSO+U and related logics, including the cost logics of Colcombet [7]. The logic studied in this paper, TMSO+zero, is another example of a logic that is not contained in MSO (and even contains MSO, if the logic is extended by allowing an outermost layer of non-weak existential set quantifiers, which does not affect decidability of satisfiability).

2 The logic and the automaton

This section describes the two main models used in the paper: the logic TMSO+zero and zero automata. The following sections discuss how the logic is translated into the automaton.

Tree notation

The logics and automata of this paper describe properties of possibly infinite binary labelled trees. We treat a node in a tree as a sequence in 2^*, with 2 denoting the set of directions $\{0, 1\}$. Define a *tree* over an alphabet Σ to be a partial function $t : 2^* \to \Sigma$ whose domain is closed under prefixes. The special case when function is total is called a *complete tree*, but we do allow incomplete trees, e.g. trees with finite domains. We use standard terminology for trees: node, root, left child, right child, leaf, ancestor and descendant. In our definition, a node might have a right child but not a left child. We write $\mathsf{trees}\Sigma$ for the set of trees over Σ.

Probability measure over paths

A *path* is defined to be a sequence in 2^ω, which is viewed as an infinite sequence of left or right turns. An equivalent definition is that a path is an ancestor-closed set of nodes that is totally ordered by the ancestor relation. When saying that a path is contained in a set of nodes, or contains a node, the second definition is used. When talking about the probability of a subset of 2^ω we use the *coin-flipping* measure, i.e. we assume that each bit is chosen independently at random, with 0 and 1 having equal probability. The probability is defined at least for all Borel subsets of 2^ω.

▶ **Definition 1.** We say a set $\Pi \subseteq 2^\omega$ has *zero probability* if it is contained in a Borel set with coin-flipping measure zero.

The sets of paths that will appear in the logic TMSO+zero will always be Borel, so the closure under subsets in the above definition will not play much of a role.

2.1 The logic

Before defining the logic TMSO+zero, we discuss the probability-free fragment TMSO.

Thin MSO without the zero quantifier

A set of nodes $X \subseteq 2^*$ is called *thin* if its *closure* defined by

$$\bar{X} \stackrel{\text{def}}{=} \{\pi \in 2^\omega : \pi \text{ passes through infinitely many nodes from } X\}$$

is countable. For example, every finite set is thin, because it has empty closure, and every path is thin, when viewed as a set of nodes, because its closure has one path. Thin sets are closed under arbitrary intersections and finite unions, but not under countable unions, because the countable set of all nodes has all paths in its closure, and is therefore not thin.

The logic *thin* MSO, denoted by TMSO, is the variant of MSO as in Rabin's theorem, except that set quantifiers range only over thin sets. The syntax of the logic is the same as for MSO from Rabin's theorem: there are two types of variable in the logic: *node variables*, which range over nodes in the domain of the input tree, and *(thin) set variables*, which range over thin subsets of the domain of the input tree. There are binary predicates for the left and right child relations, and there is a unary predicate for every label in the input alphabet. By the Cantor-Bendixson theorem, a set of nodes X is thin if and only if one cannot find a subset $Y \subseteq X$ such that Y, when ordered by the descendant relation, is a complete binary tree. Since this alternative characterisation can be formalised in MSO, it follows that TMSO is a fragment of MSO in terms of expressive power. On the other hand, TMSO is at least as expressive as WMSO with path quantifiers.

As far as the author knows, the logic TMSO was not considered explicitly in the literature so far, and it might be interesting to examine its expressive power, e.g. prove that it is strictly weaker than MSO and maybe, in the long run, find an algorithm which inputs a formula of MSO and decides if the formula is equivalent to some formula in TMSO. This investigation, however, is not the topic of the present paper. The present paper is about extending TMSO with a quantifier for zero probability.

Thin MSO with the zero quantifier

We now define the main topic of this paper, i.e. the logic TMSO+zero. First we explain why our point of departure for adding the zero quantifier is TMSO and not some other fragment of MSO. The reason is that TMSO is the strongest logic we could find such that the set quantifiers commute with the probabilistic quantifier in a way which will be made more precise in Section 6. The key observation reason is this: if the domain of the input tree is thin, then it has countably many paths, and therefore the zero quantifier can be eliminated because it always says "yes".

A parameter in the definition of TMSO+zero is a family zero of subsets of 2^ω. The example we have in mind is that zero is the sets with zero probability according to Definition 1, but the results will also work for other choices of zero. The logic TMSO+zero is the extension of the logic TMSO defined above, by adding a quantifier, called zero, which binds a thin set variable π, and such that

$$\mathsf{zero}\pi\ \varphi(\pi)$$

is true if zero contains the set of paths π which are contained in the domain of the input tree and make $\varphi(\pi)$ true, assuming that a path is treated as a set of nodes. (Formally speaking, the path π is seen as a set of nodes when evaluating $\varphi(\pi)$, and as an element of 2^ω when measuring how many paths π make $\varphi(\pi)$ true.)

▶ **Example 2.** Consider an alphabet $\{a, b\}$. The following formula says that zero contains the set of paths that visit at least one a:

$$\mathsf{zero}\pi\ \exists x\ (x \in \pi \wedge a(x)).$$

If the parameter zero is prefix independent (see Definition 7 for a more precise treatment) and does not contain the set 2^ω of all paths, then the above formula is equivalent to $\forall x\ \neg a(x)$, and therefore the zero quantifier can be avoided.

▶ **Example 3.** Consider an alphabet $\{a, b\}$. The following formula says that zero contains the set of paths which visit b finitely often:

$$\mathsf{zero}\pi\ \exists x\big(x \in \pi \wedge \forall y\ (y \geq x \wedge y \in \pi \Rightarrow b(x))\big)$$

If zero is our guiding example of zero probability, the negation of the above formula says that the Büchi condition is satisfied with probability one. As shown in [6], Theorem 21, the property above is not definable in MSO.

▶ **Example 4.** The reduction from qualitative PCTL* in Theorem 5 from [10] produces formulas where set quantification is only used for paths. Therefore, qualitative PCTL* is a special case of TMSO+zero.

Beyond Thin MSO with the zero quantifier

In the logic TMSO+zero, the set variables are restricted to thin sets. The obvious question is about the more general case, where set variables range over arbitrary sets of nodes, not necessarily thin ones. As mentioned in the introduction, the more general logic was introduced in [10], under the name MSO$+\forall_\pi^{=1}$, and the authors asked about decidability of its satisfiability problem. A long term project for this research is to find out if the satisfiability problem for the more general logic is decidable – or not. In this paper we only begin the project, by studying the thin variant. One scenario is that the thin variant is decidable, but the non-thin variant is undecidable, which would be similar to the situation for MSO+U, where weak variants are decidable, but the full logic is undecidable. However, one should not take the analogy with MSO+U too far: e.g. the thin variant of MSO+U would already be undecidable, because MSO+U is undecidable already for ω-words.

Another natural version of MSO with probability would be to choose a subset of 2^* at random, with each node chosen independently, and then have a quantifier that says there is zero probability of finding a subset with a given property. This logic was proved undecidable in [10], already for ω-words (which can be seen as a special case of TMSO), and the undecidability proof works also for formulas of the form

$$\text{there is zero probability of choosing a set } X \subseteq \mathbb{N} \text{ which makes } \varphi(X) \text{ true,}$$

where $\varphi(X)$ is a formula of first-order logic that defines a set of ω-words over alphabet 2. Therefore, it seems that this kind of probabilistic quantifier is doomed to undecidability.

2.2 The automaton

Having defined the logic TMSO+zero, we define our main automaton model, which is called a zero *automaton*. Like in the logic TMSO+zero, a parameter of the semantics for the automaton is a family zero of subsets of 2^ω. The idea is that the automaton extends a nondeterministic parity automaton with the ability to say that the set of paths satisfying the parity condition belongs, or does not belong, to zero.

▶ **Definition 5.** The syntax of a zero automaton is a tuple

$$\underbrace{Q}_{\text{states}} \qquad \underbrace{\Sigma}_{\text{input alphabet}} \qquad \underbrace{I \subseteq Q}_{\text{initial states}} \qquad \underbrace{\bigcup_{C \subseteq 2} \delta_C \subseteq Q \times \Sigma \times Q^{|C|}}_{\text{transitions}},$$

with all components finite, together with a total order on Q and four subsets

$$Q_{\text{all}}, Q_{\text{zero}}, Q_{\text{nonzero}}, Q_{\text{seed}} \subseteq Q.$$

The idea behind the transitions is that $\delta_{\{0,1\}}$ is used for those nodes which have both children defined, but e.g. $\delta_{\{1\}}$ is used for nodes where only the right child is defined, and $\delta_\emptyset$ is used for leaves.

The semantics are defined as follows. The automaton is run on a tree over the input alphabet, which might not necessarily be complete. A *run* of the automaton is a tree labelled by states with the same domain as the input tree, which is consistent with the transition relation in the following sense: if a node x is in the domain, and we define

$$C \stackrel{\text{def}}{=} \{i \in 2 : xi \text{ is in the domain}\}$$

then there must be a transition in δ_C which relates the state in x, the label of x in the input tree, and the states in the children of x that are in the domain. A tree is accepted if it admits a run which has the initial state in the root and is accepting in the following sense. Define the *maxinf state* on a path in a run to be the maximal state that appears infinitely often on the path. When talking about a maximal state, we refer to the total order on states that is given in the syntax of the automaton. A run ρ is accepting if all of the following conditions hold, assuming that $\mathsf{paths}\,\rho \subseteq 2^\omega$ denotes the set of paths contained in ρ:

1. **all paths acceptance condition:** every path from $\mathsf{paths}\,\rho$ has maxinf in Q_{all}; and
2. **zero acceptance condition:** zero contains the set of paths from $\mathsf{paths}\,\rho$ which have maxinf state in Q_{zero}; and
3. **nonzero acceptance condition:** for every node x in the run with state $q \in Q_{\text{seed}}$:

$$\mathsf{zero} \not\ni \{\pi \in \mathsf{paths}\,\rho : \begin{cases} \pi \text{ passes through } x, \text{ and} \\ \pi \text{ sees only states } < q \text{ after } x, \text{ and} \\ \pi \text{ has maxinf state in } Q_{\text{nonzero}} \end{cases} \}$$

An automaton is called *zeroless* if Q_{zero} is empty (which makes the zero condition vacuously true) and *seedless* if there are no seed states, i.e. Q_{seed} is empty (which makes the nonzero condition vacuously true). In particular, a zeroless and seedless automaton is the same thing as a parity automaton, which proves the zero automata are at least as powerful as MSO.

▶ **Example 6.** Assume that zero is probability zero as in Definition 1. Consider the special case of a zero automaton where Q_{all} is all states and Q_{seed} is empty. A run is accepting if and only if there is zero probability of having maxinf state in Q_{zero}. Equivalently, the probability of having maxinf state outside Q_{zero} is one. Languages recognised by such automata are the *qualitative tree languages* from [6]. The class of *positive tree languages* from [6] is obtained when Q_{all} and Q_{zero} are empty, and the initial state is used only once in the root, is maximal in the total order, and is the unique seed state.

3 Fat Cantor

In this section, we illustrate the logic and automaton with an extended example. Let us fix zero to be probability zero according to Definition 1. Define the *fat Cantor language* to be the set of complete trees over the alphabet $\{a, b\}$ which satisfy the following property:

$$\underbrace{\neg \mathsf{zero}\pi\big(\forall x\ x \in \pi \wedge b(x)\big)}_{\text{nonzero probability of avoiding } a} \quad \wedge \quad \underbrace{\forall x \exists y\ y \geq x \wedge a(y)}_{a\text{'s are dense}}$$

Note that "avoiding a" is a Borel property of paths, and therefore "nonzero probability of avoiding a" means that the sets of paths avoiding a have defined positive probability. This argument will be true in general for our logic – for every fixed input tree, any property of paths definable in the logic will be Borel, and therefore not belonging to zero will mean that it there is defined and positive probability.

The fat Cantor language is nonempty. To construct a tree in the fat Cantor language, choose a fast growing sequence of natural numbers

$$n_1 < n_2 < n_3 < \cdots$$

and then choose a tree (which is unique up to reordering siblings) where a labels are found only at depths from the sequence above, and every node at depth n_i has a unique descendant at depth n_{i+1} with label a. If the sequence (n_i) grows fast enough, then there is nonzero probability of avoiding a. Let us now argue that the fat Cantor language contains no regular tree, i.e. no tree with finitely many nonisomorphic subtrees. Suppose then that t is a regular tree, with n distinct subtrees. If a's are dense in this tree, it follows from regularity that every node has a descendant at distance at most n that has label a. This means there is some constant $\epsilon > 0$ such that for every interval $I \subseteq \mathbb{N}$ of n consecutive positions, the probability of a path visiting a at depth from I is at least ϵ. These events are independent for disjoint intervals, and therefore the probability of seeing a at least once, and even infinitely often, is 1. Summing up: the fat Cantor language is nonempty but contains no regular trees.

Fat Cantor automaton

We now show a zero automaton which recognises the fat Cantor language described above. To simplify notation, we define an automaton which works only on complete trees, i.e. it recognises the intersection of the fat Cantor language with the set of complete trees. In particular, when talking about transitions, we only consider transitions δ_C for $C = \{0, 1\}$.

The input alphabet is $\{a, b\}$. The automaton has four states, totally ordered as follows:

$$\underbrace{q_a}_{\text{already saw } a} < \underbrace{q_1}_{\text{searching for } a} < \underbrace{q_2}_{\text{not searching for } a} < \underbrace{q_0}_{\text{initial state}}$$

The automaton begins in state q_0 in the root, this state will not be visited again during the run. When the automaton is in state q_i with $i \in \{0, 1, 2\}$ and it reads a node with label b, then it sends q_1 to some child and q_2 to the other child, as witnessed by the following transitions:

$$(q_i, b, q_j, q_k) \quad \text{for } i \in \{0, 1, 2\} \text{ and } \{j, k\} = \{1, 2\}.$$

Choosing which child gets q_1 and which child gets q_2 is the only source of nondeterminism in this automaton. When the automaton sees letter a, it sends q_a to both children regardless of

its current state, and q_a is a sink state that cannot be left, as witnessed by the following transitions:

$$(q, a, q_a, q_a) \quad \text{for all } q \in Q \qquad (q_a, a, q_a, q_a) \qquad (q_a, b, q_a, q_a)$$

Since q_0 is used only once in the root, and q_a is a sink state, it follows that on every path either q_a is seen from some point on, or q_a is never seen and the maxinf state is one of q_1, q_2. The acceptance condition is defined by the following sets:

$$Q_{\text{all}} = \{q_a, q_2\} \qquad Q_{\text{zero}} = \emptyset \qquad Q_{\text{nonzero}} = \{q_1, q_2\} \qquad Q_{\text{seed}} = \{q_0\}$$

Because Q_{zero} is empty, every run satisfies the zero acceptance condition. The state q_0 appears only once in the root, and therefore it is never used as a maxinf state. By choice of Q_{all}, the state q_1 is forbidden as a maxinf state, which means that in an accepting run, every path eventually stabilises on either q_a or q_2. Since the only way of leaving q_1 is by seeing an a letter, it follows that a's must be dense. The only seed state is the initial state, which is used only once in the root, and is also the most important state. Therefore, a run satisfies the nonzero acceptance condition if and only if its there is nonzero probability of having maxinf state in $\{q_1, q_2\}$, which means there is nonzero probability of avoiding a.

4 From logic to automata

The main technical result of this paper is that every formula of TMSO+zero can be effectively translated to an equivalent zero automaton. The result works not just for zero probability, but for other choices of zero, as described in the following definition.

▶ **Definition 7.** For a family zero of subsets of 2^ω, consider the following properties:

1. **σ-ideal:** zero is closed under subsets and countable union;
2. **atomless:** zero contains all singletons;
3. **prefix independence:** every set $\Pi \subseteq 2^\omega$ satisfies

$$\Pi \in \mathsf{zero} \Leftrightarrow i\Pi \in \mathsf{zero} \qquad \text{for every } i \in 2$$

4. **recurrent nonzero:** there is a zero automaton which recognises the language

$$\{t \in \mathsf{trees}\{1,2,3\} : \text{for every subtree, the set of paths with maxinf 2 is } \notin \mathsf{zero}\}$$

In the recurrent nonzero condition, it is important that the trees are not necessarily complete. For such a tree, a subtree is obtained by shifting the root to some node in the domain. In particular, if a tree belongs to the language from the recurrent nonzero condition, then it cannot have any leaves.

Here is the main result of this paper.

▶ **Theorem 8.** *Let* zero *be a family of subsets of* 2^ω *satisfying conditions 1-4 in Definition 7. Then for every formula of* TMSO+zero *one can compute an equivalent* zero *automaton.*

The proof has three steps. In Section 5, we show closure properties of languages recognised by zero automata, of which the most interesting is closure under intersection. In Section 6, we show that the logic TMSO+zero has the same expressive power as a certain transducer model. In Section 7, we complete the proof of the theorem, by translating transducers into zero automata. The results in Section 5 and 6 only use properties 1-3 in Definition 7, while Section 7 uses also property 4.

The following corollary shows the main application of Theorem 8.

▶ **Corollary 9.** *Let* zero *be the subsets of* 2^ω *that have zero probability in the sense of Definition 1. Then for every formula of* TMSO+zero *one can compute an equivalent* zero *automaton.*

Other examples of zero which can be shown to satisfy the assumptions of Theorem 8 include "countable sets of paths [2]" and "meagre sets of paths [9]". These other examples are less interesting because they can already be formalised in MSO alone, i.e. parity automata are sufficient. Theorem 8 can be seen as an alternative way of recovering the results from [2, 9]: one only needs to check that the assumptions of Theorem 8 are satisfied for a particular choice of zero, and that zero automata can be captured by MSO. In view of the results from [2, 9], we have only one example of zero that satisfies the assumptions of Theorem 8, and which strictly extends MSO, namely probability zero.

5 Closure properties of zero automata

This section is about closure properties of the class of languages recognised by zero automata. We show that this class is closed under positive Boolean operations – with intersection being by far the more interesting case. We do not know if languages recognised by zero automata are closed under complementation. If they would be, then zero automata would have exactly the same expressive power as full MSO+zero.

Define a *Mealy machine* to be a deterministic finite automaton on words over some input alphabet Σ, where every transition is labelled by a letter from some output alphabet Γ. Such a machine can be run on a finite word, yielding a length preserving function $\Sigma^* \to \Gamma^*$, it can also be run on an ω-word, yielding a function $\Sigma^\omega \to \Gamma^\omega$, or finally it can be run on all paths in a tree, yielding a function $\mathsf{trees}\Sigma \to \mathsf{trees}\Gamma$ which does not change the domain of the tree. The last case will be called a *tree transducer recognised by a Mealy machine.*

▶ **Lemma 10.** *Languages recognised by* zero *automata are closed under union, as well as images and inverse images under tree transducers recognised by Mealy machines.*

Proof sketch. The lemma does not require any closure properties from the set zero. For union, we use disjoint union of automata (and gluing the initial state). For images use nondeterminism, and for both images and inverse images use a Cartesian product construction to simulate the Mealy machine in the state space of the zero automaton. Note that state spaces in zero automata are ordered. Therefore we impose some random total order on a Mealy machine, and in the Cartesian product we use a lexicographic ordering, with the order on the original zero automaton being more important. ◀

We now show another closure property, which is closure under factorisations, as described below. Define a *factor* to be a set of nodes that is connected with respect to the child relation. In particular, a factor has a unique *root*, i.e. a unique node which is least with respect to the descendant ordering. If X is a factor, then define the *restriction to* X of a tree t to be the tree obtained from t by keeping only the nodes from X. We now show that if L is a language recognised by a zero automaton, then there is a zero automaton which inputs a tree together with a decomposition into disjoint factors, and checks that L contains every tree obtained by restricting the input tree to one of the factors in the partition.

We begin by describing how a decomposition into factors is given on the input. If X is a set of nodes, then define an X-factor to be a set of nodes obtained by taking some $x \in X$ and adding all descendants y such that $(x..y]$ is disjoint with X, where $(x..y]$ denotes proper descendants of x that are (not necessarily proper) ancestors of y. By abuse of notation, we

define an X-factor of a tree t to be any tree obtained from t by restricting it to some X-factor. Finally, if X is a set of nodes in a tree $t \in \mathsf{trees}\Sigma$, then define $t \otimes X \in \mathsf{trees}(\Sigma \times 2)$ to be the tree obtained from t by extending the label of each node by a bit indicating membership in X.

▶ **Lemma 11** (Factorisation Lemma). *Assume that* zero *satisfies conditions 1–3 in Definition 7. If $L \subseteq \mathsf{trees}\Sigma$ is recognised by a* zero *automaton, then so is*

$$\{t \otimes X : t \in \mathsf{trees}\Sigma \text{ and } X \text{ is a set of nodes in } t \text{ such that } L \text{ contains every } X\text{-factor of } t\}.$$

The main idea in the proof is that to use the "nested" character of the nonzero acceptance condition; here by nesting we mean that the paths contributing to the nonzero condition are cut off whenever a more important state is seen.

We finish this section by stating the most challenging result, which is closure under intersection, as stated in the following lemma.

▶ **Lemma 12** (Intersection Lemma). *Assume that* zero *satisfies conditions 1-3 in Definition 7. Then languages recognised by* zero *automata are closed under intersection.*

The proof has several steps. One of these steps, namely the first step, is showing that languages recognised by zero automata are closed under intersection with languages recognised by zero automata which do not use the nonzero acceptance condition. The first step uses McNaughton's Latest Appearance Record construction.

6 Transducers

To prove Theorem 8, we use a transducer characterisation of the logic TMSO+zero. The transducer characterisation is an "if and only if" characterisation, unlike the translation in the main Theorem 8.

Transducers

Define a *tree transducer* to be any function $\mathsf{trees}\Sigma \to \mathsf{trees}\Gamma$ which does not change the domain of the input tree. Our goal is to show each language definable MSO+zero can be described by composing transducers of certain basic types. To model a language as a transducer, we use the following definition.

▶ **Definition 13.** For a tree language $L \subseteq \mathsf{trees}\Sigma$, define

$$\mathsf{trans}L : \mathsf{trees}\Sigma \to \mathsf{trees}2,$$

called the *characteristic transducer of* L, to be the transducer which labels each node of the input tree by a bit saying whether or not the subtree rooted in that node belongs to L.

We define the *combination* $t_0 \otimes t_1$ of two trees t_0, t_1 over possibly different alphabets Σ_0, Σ_1 but with equal domains, to be the unique tree over $\Sigma_0 \times \Sigma_1$ which projects to each t_i on the i-th coordinate. In the following theorem, composition of transducers is defined as for functions, while the combination of two transducers f_1, f_2 with the same input alphabet but possibly different output alphabets is the transducer $t \mapsto f_1(t) \otimes f_2(t)$.

▶ **Theorem 14.** *Assume that* zero *has the closure properties 1-3 from Theorem 8. Then a tree language is definable in* TMSO+zero *if and only if its characteristic transducer belongs to the smallest class of transducers which is closed under composition and combination, and which contains the following transducers:*

1. **Zero base.** *The characteristic transducers of all languages of the form:*

$$\mathsf{Z}_n \stackrel{def}{=} \{t \in \mathsf{trees}\{1,\dots,n,\bot\} : \mathsf{zero} \ni \{\pi \in \mathsf{paths}\, t : \begin{cases} \pi \text{ does not visit } \bot, \text{ and} \\ \pi \text{ has even maxinf} \end{cases} \}\}$$

2. **Zeroless base.** *The characteristic transducers of all languages definable in* TMSO.
3. **Child number transducer.** *Transducers of the form* $\mathsf{trees}\Sigma \to \mathsf{trees}2$ *which map each node to its child number, with the convention that the root gets label* 0.
4. **Mealy machine on trees.** *Transducers recognised by Mealy machines.*

The difficult implication is from logic to transducers; here we use the composition method. Intuitively speaking, the above theorem shows that formula of TMSO+zero can be decomposed into parts that do not talk about zero at all, and into the very basic property Z_n.

7 From transducers to zero automata

In this section we complete the proof of Theorem 8, by showing that the transducers from the previous section can be compiled into zero automata. We say that a tree transducer f is *recognised* by a zero automaton if there is a zero automaton recognising the set of trees $t \otimes f(t)$ where t ranges over all input trees for the tree transducer.

▶ **Lemma 15.** *Transducers recognised by* zero *automata are closed under composition, combination and include the child number transducers, transducers induced by Mealy machines, and the characteristic transducers of all languages definable in* TMSO.

Proof sketch. For composition, the automaton guesses the intermediate result, and checks both underlying transducers in parallel, using the Intersection Lemma. Combination also uses intersection. For the child-number transducers, Mealy machines and characteristic transducers of languages definable in TMSO, one observes that their corresponding languages are definable in MSO, and zero automata generalise nondeterministic parity tree automata. ◀

By Theorem 14 and the above lemma, in order to prove Theorem 8 it suffices to show that zero automata recognise the characteristic transducers of the languages of the form Z_n as used in Theorem 14. By unraveling the definitions, we need to show the following lemma.

▶ **Lemma 16.** *For every* $n \in \mathbb{N}$ *there is a* zero *automaton recognising the set of trees*

$$t \otimes s \qquad \text{with } t \in \mathsf{trees}\{1,\dots,n,\bot\},\ s \in \mathsf{trees}2$$

such that for every node x*, its label in* s *is* 1 *iff* Z_n *contains the subtree of* t *rooted in* x*.*

Proof. Let L be the language in the statement of the lemma. For a tree $t \otimes s$, define a $\bot$-factor to be a maximal factor contained in the domain of the tree that does not use label $\bot$ in t. It is not difficult to see that $t \otimes s$ belongs to L if and only if: (a) every node with label $\bot$ in t has label 1 in s; and (b) every $\bot$-factor belongs to L. Condition (a) can be easily checked by a parity automaton, so thanks to the Intersection Lemma it suffices to produce a zero automaton which checks (b). By the Factorisation Lemma, it suffices to find a zero automaton which tests memberhip in L for individual $\bot$-factors.

Summing up, we can assume without loss of generality that t does not use label $\bot$ at all. Therefore, in the rest of the proof, we show a zero automaton which recognises the language L restricted to the case where $t \in \{1,\dots,n\}$.

For $i \in \{1, \ldots, n\}$, consider the function

$$f_i : \mathsf{trees}\{1, \ldots, n\} \to \mathsf{trees}\{1, 2, 3\} \qquad \text{label of } x \text{ in } f_i(t) = \begin{cases} 1 & \text{if label of } x \text{ in } t \text{ is } < i \\ 2 & \text{if label of } x \text{ in } t \text{ is } = i \\ 3 & \text{if label of } x \text{ in } t \text{ is } > i \end{cases}.$$

We will only use this function for even i. For $t \in \mathsf{trees}\{1, \ldots, n\}$, define nonzero$(t)$ to be the set of nodes in t whose subtree does not belong to Z_n. In terms of this definition, a tree $t \otimes s$ belongs to L if and only if nonzero(t) is exactly the nodes that have label 0 in s. Also, condition 4 from Definition 7 says that there is a zero automaton recognising the language

$$\mathsf{N} \stackrel{\text{def}}{=} \{t \in \mathsf{trees}\{1, 2, 3\} : \text{nonzero}(t) \text{ is all nodes of } t\}$$

▶ **Claim 17.** *Let $t \in \mathsf{trees}\{1, \ldots, n\}$ and $s \in \mathsf{trees2}$ be trees with the same domain. Then $t \otimes s \in L$ if and only if one can find an ancestor closed set of nodes $\{X_i\}_i$, with i ranging over even numbers in $\{1, \ldots, n\}$, such that the following conditions hold:*

1. *a node has label 0 in s if and only if it belongs to some X_i;*
2. *for every even $i \in \{1, \ldots, n\}$, restricting $f_i(t)$ to the nodes from X_i yields a tree in N;*
3. $\mathsf{zero} \ni \{\pi \in \mathsf{paths}\, t : \pi$ *has even t-maxinf and sees 0 finitely often in s*$\}$

Before proving the claim, let us observe how it implies the lemma. Since a zero automaton can nondeterministically guess the sets X_i, it suffices to show that there is a zero automaton which checks conditions 1, 2, 3 in the claim. By the Intersection Lemma, it suffices to check each condition individually. Condition 1 is definable in MSO. Condition 2, for any fixed i, follows from the assumption that N is recognised by a zero automaton and the Factorisation Lemma. For condition 3, it is straightforward to construct a zero automaton – it essentially copies the labels from t into its states, except that nodes with label 0 in s trigger a state which is maximal in the total order. It remains to prove the claim.

Proof. We begin with the following observation, which follows from the assumption that zero satisfies conditions 1-3 in Definition 7. For every $t \in \mathsf{trees}\{1, \ldots, n\}$, the set nonzero$(t)$ is closed under ancestors and a node x belongs to nonzero(t) if and only if

$$\mathsf{zero} \not\ni \{\pi \in \mathsf{paths}\, t : \begin{cases} \pi \text{ is contained in nonzero}(t), \text{ and} \\ \pi \text{ passes through } x, \text{ and} \\ \pi \text{ has even } t\text{-maxinf} \end{cases} \} \tag{1}$$

By definition of the tree transducers f_i, a path has even t-maxinf if and only if it has even $f_i(t)$-maxinf for some even i. Therefore, by closure of zero under countable – and therefore also finite – unions, we see that

$$\text{nonzero}(t) = \bigcup_i \text{nonzero}(f_i(t)), \tag{2}$$

where i ranges over even numbers in $\{1, \ldots, n\}$.

Let us now prove the claim.

Let us begin with the bottom-up implication. From condition 2 it follows that every node in X_i belongs to nonzero(t). From condition 1 it follows that all nodes with label 0 are in nonzero(t). From condition 1, it follows that the set of nodes with label 0 in s is closed under ancestors. Therefore, condition 3 implies that for every node with label 1 in s is outside

nonzero(t). Thus nonzero(t) is exactly the nodes which have label 0 in s, which means that $t \otimes s \in L$.

Consider the top-down implication. Our assumption is that nonzero(t) is exactly the nodes which have label 0 in s. Define X_i to be nonzero($f_i(t)$). By (2), we see that condition 1 in the statement of the claim holds. From (1) applied to the trees $f_i(t)$, we get condition 2. To prove condition 3, by definition of nonzero(t) and prefix independence of zero, we know that every node $x \notin$ nonzero(t) satisfies

$$\mathsf{zero} \ni \{\pi \in \mathsf{paths}\, t : \pi \text{ passes through } x \text{ and has even } t\text{-maxinf}\}$$

Since nonzero(t) is ancestor closed, it follows that a path passes through some $x \notin$ nonzero(t) if and only if it sees 0 in s finitely often. Therefore, by closure of zero under countable unions, we get condition 3 in the statement of the claim. ◀
◀

8 Conclusion

We have proved that, under certain conditions on zero, every formula of the logic TMSO+zero is recognised by a zero automaton. Therefore, in order to decide satisfiability of TMSO+zero, it suffices to decide emptiness for zero automata. Unlike the logic, zero automata involve no nesting, which makes the emptiness check easier. A planned followup paper will show that emptiness is indeed decidable for zero automata, assuming that zero is the sets of probability zero.

Apart from the emptiness question for zero automata, the main open problem is decidability for the full logic MSO+zero, and not just the thin variant considered in this paper. It is not at all clear if zero automata are closed under complement, and therefore it is quite possible that zero automata are not the right model for MSO+zero. There is another logic, which sits between TMSO+zero and MSO+zero, and which might still admit a translation to zero automata. In this intermediate logic, the condition on sets $X \subseteq 2^*$ is relaxed: instead of thin sets, we consider sets which satisfy $\bar{X} \in$ zero. We leave open the question whether this intermediate logic admits a translation to zero automata.

Acknowledgment. I would like to thank Henryk Michalewski and Matteo Mio for introducing me into this area, and for their many valuable comments and suggestions.

References

1 Christel Baier, Marcus Größer, and Nathalie Bertrand. Probabilistic ω-automata. *J. ACM*, 59(1):1, 2012. doi:10.1145/2108242.2108243.

2 Vince Bárány, Łukasz Kaiser, and Alexander Rabinovich. Cardinality quantifiers in MLO over trees. In *Proc. of CSL*, 2009.

3 Mikolaj Bojanczyk. U. *ACM SIGLOG News*, 2(4):2–15, 2015.

4 Tomás; Brázdil, Vojtech Forejt, Jan Kretínský, and Antonín Kucera. The satisfiability problem for probabilistic CTL. In *Proc. of LICS*, pages 391–402, 2008.

5 Julius R. Büchi. On a decision method in restricted second-order arithmetic. In *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science*, pages 1–11, 1962.

6 Arnaud Carayol, Axel Haddad, and Olivier Serre. Randomization in automata on infinite trees. *ACM Trans. Comput. Log.*, 15(3):24:1–24:33, 2014. doi:10.1145/2629336.

7 Thomas Colcombet. Regular cost functions, part I: logic and algebra over words. *Logical Methods in Computer Science*, 9(3), 2013. doi:10.2168/LMCS-9(3:3)2013.

8 Daniel Lehmann and Saharon Shelah. Reasoning with time and chance. *Information and Control*, 53(3):165–1983, 1982.

9 Henryk Michalewski and Matteo Mio. Baire category quantifier in monadic second order logic. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, pages 362–374, 2015. doi:10.1007/978-3-662-47666-6_29.

10 Henryk Michalewski and Matteo Mio. Measure quantifier in monadic second order logic. In *Logical Foundations of Computer Science – International Symposium, LFCS 2016, Deerfield Beach, FL, USA, January 4-7, 2016. Proceedings*, pages 267–282, 2016. doi:10.1007/978-3-319-27683-0_19.

11 Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of American Mathematical Society*, 141:1–35, 1969.

12 Alexander Rabinovich. On decidability of monadic logic of order over the naturals extended by monadic predicates. *Inf. Comput.*, 205(6):870–889, 2007. doi:10.1016/j.ic.2006.12.004.

13 Sergiu Hart Micha Sharir. Probabilistic propositional temporal logics. *Information and Control*, 70(2–3):97–155, 1986.

Deciding Piecewise Testable Separability for Regular Tree Languages*

Jean Goubault-Larrecq[1] and Sylvain Schmitz[2]

1 **LSV, ENS Cachan & CNRS & Inria, Université Paris-Saclay, Paris, France**
goubault@lsv.fr
2 **LSV, ENS Cachan & CNRS & Inria, Université Paris-Saclay, Paris, France**
schmitz@lsv.fr

Abstract

The piecewise testable separability problem asks, given two input languages, whether there exists a piecewise testable language that contains the first input language and is disjoint from the second. We prove a general characterisation of piecewise testable separability on languages in a well-quasi-order, in terms of ideals of the ordering. This subsumes the known characterisations in the case of finite words. In the case of finite ranked trees ordered by homeomorphic embedding, we show using effective representations for tree ideals that it entails the decidability of piecewise testable separability when the input languages are regular. A final byproduct is a new proof of the decidability of whether an input regular language of ranked trees is piecewise testable, which was first shown in the unranked case by Bojańczyk, Segoufin, and Straubing [Log. Meth. in Comput. Sci., 8(3:26), 2012].

1998 ACM Subject Classification F.4.3 Formal Languages, F.4.1 Mathematical Logic

Keywords and phrases Well-quasi-order, ideal, tree languages, first-order logic

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.97

1 Introduction

The *separability* problem for a class $\mathcal{C}$ of input languages and a class $\mathcal{S}$ of separators, asks given two input languages L and L' from $\mathcal{C}$ whether there exists a language S from $\mathcal{S}$ such that $L \subseteq S$ and $S \cap L' = \emptyset$. This classical problem has been studied in formal language theory since the 70's [27], but has sparked renewed interest due in particular to its connection with the *definability* problem: given L from $\mathcal{C}$, does L belong to $\mathcal{S}$? When $\mathcal{C}$ is effectively closed under complement, this last question reduces to the separability of L and of its complement. Separability thus generalises definability, and has been instrumental in the recent advances on the definability problem for the alternation hierarchy over finite words [25].

We focus in this paper on the class of *piecewise testable* languages as class $\mathcal{S}$ of separators. Over finite words, this coincides with the languages defined by $\mathcal{B}\Sigma_1(<)$, the Boolean combinations of existential first-order sentences with order, and is one of the oldest and best-known classes of languages with decidable definability: Simon [26] showed that a language is piecewise-testable if and only if its syntactic monoid is $\mathcal{J}$*-trivial*. Lately, this has been extended in two different directions:

- Over finite unranked ordered trees, Bojańczyk, Segoufin, and Straubing [6] generalise Simon's algebraic approach and characterise the syntactic forest algebræ of piecewise

* A full version of this work is available online at https://hal.inria.fr/hal-01276119/.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 97; pp. 97:1–97:15


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

testable tree languages defined by the most common signatures. This yields the decidability of the piecewise testable definability problem for regular unranked tree languages.

- Over finite words, Almeida and Zeitoun [2, 3] first proved the decidability of the PTL separability problem for regular input languages, based on a topological characterisation of separability. This abstract characterisation has been turned into a much more efficient characterisation in terms of patterns found simultaneously in L and L' [24, 11], which culminated in the work of Czerwiński, Martens, van Rooijen, Zeitoun, and Zetzsche [12] with a proof of the decidability of piecewise separability for many classes $\mathcal{C}$ of input languages, which even includes higher-order languages [17, 8, 4]. The crux of this proof is a reduction to the computation of representations for downward-closures [30] ordered by scattered subword embedding.

Separability for Tree Languages. Considering these two lines of research side by side begs the question whether piecewise testable separability might be decidable over richer structures than words, and in particular over trees. In this paper, we answer positively by proving:

▶ **Theorem 1.** *Piecewise testable separability is decidable for regular languages of ranked trees ordered by homeomorphic embedding.*

Since the class of regular tree languages is effectively closed under complement, Theorem 1 entails as a sub-problem the decidability of the corresponding definability problem:

▶ **Corollary 2.** *Piecewise testable definability is decidable for regular languages of ranked trees ordered by homeomorphic embedding.*

Thus, apart from the restriction to ranked trees, Theorem 1 also yields as a byproduct a new proof of decidability for the piecewise testable definability problem studied by Bojańczyk et al. [6], in the most challenging case of homeomorphic embeddings.

Order-Theoretic Framework. We employ vastly different techniques from Bojańczyk et al.'s, and Theorem 1 should be thought of as a proof of concept for our main contribution, which is a very general order-theoretic framework for piecewise testable separability:

- We characterise in Section 3 separability by piecewise testable languages over any combinatorial *well-quasi-order*; this encompasses for instance words ordered by scattered subword embedding and trees ordered by homeomorphic embedding or minor ordering. Our characterisation proceeds similarly to the topological characterisation of Almeida [2] by comparing the closures of L and L', but is stated in purely order-theoretic terms and compares the *ideals* – i.e. the irreducible downwards-closed sets – adherent to L and L'. The advantage of this approach is that it scales to more complex structures than words – we have a toolkit of effective ideal representations at our disposal [13, 15] – and leads to a very simple and general proof, that delegates syntactic manipulations to the ideal representation. Our techniques encompass the word case: e.g. the patterns derived by Czerwiński et al. [12] turn out to be representations for word ideals [19, 1], and we eschew the technical work on patterns that absorbed a large part of Czerwiński et al.'s proof.
- We then show in Section 4 how to turn our characterisation into a generic decision procedure, by reduction to the *adherence membership* problem for the class $\mathcal{C}$. Provided we have effective representations for ideals, this yields directly the decidability for regular tree languages. We also derive the reduction of Czerwiński et al. to the computation of downward-closures in full generality.

- In order to instantiate this framework for tree languages, we provide in Section 5 our last ingredient: an effective representation for ideals of ranked trees ordered by homeomorphic embedding. This is a contribution of independent interest, which fixes an issue with the representation proposed in [13].

We start in Section 2 by defining piecewise testable languages in the general setting of combinatorial quasi-orders, and by recalling their logical characterisation. We assume the reader is already familiar with tree languages; see [9] for a general reference.

2 Piecewise Testable Languages

We work in the first part of this paper with *combinatorial* classes X of elements, where X is a countable set equipped with a *size* function $|.|: X \to \mathbb{N}$ such that $X_{\leq n} \stackrel{\text{def}}{=} \{x \in X \mid |x| \leq n\}$ is finite for every n. As soon as we equip X with a quasi-ordering $\leq$, we can define a notion of piecewise-testable languages (see Section 2.1). The usual setting for piecewise testable languages is however that of classes of finite structures along with the embedding relation, in which case those sets also enjoy a logical characterisation (see Section 2.2).

2.1 Pieces

Two elements x and y are *n-piecewise equivalent*, noted $x \equiv_n y$, if for every $z \in X_{\leq n}$, $z \leq x$ if and only if $z \leq y$. A subset S of X is called *n-piecewise testable* (an n-PTL) over $(X, \leq)$ if S is a union of n-piecewise equivalence classes. A subset S of X is *piecewise testable* (a PTL) if it is an n-PTL for some $n \geq 0$. This is well-known to be equivalent to the following formulation:

▶ **Fact 3.** *S is an n-PTL over $(X, \leq)$ if and only if S is a finite Boolean combination of* principal filters $\uparrow x \stackrel{\text{def}}{=} \{x' \in X \mid x \leq x'\}$ *where $x \in X_{\leq n}$.*

2.2 Logical Characterisation

Although the choice of the particular quasi-order $(X, \leq)$ is irrelevant to Fact 3 and the treatment in Section 3, one normally chooses X to be a class $\mathcal{K}$ of finite structures over some signature σ and $\leq$ to be the *induced substructure* ordering, which we denote by '$\sqsubseteq$'. When $(\mathcal{K}, \sqsubseteq)$ is a *well-quasi-order*, this leads to a well-known logical characterisation in terms of the Boolean closure of existential first-order formulæ– one of the chief motivations for studying piecewise testable languages. Other orderings may lead to different logical characterisations, or have no logical characterisation.

2.2.1 Finite Structures

We consider finite relational structures $\mathfrak{M} = \langle M, (R_i)_{i \in I} \rangle$ over a finite signature $\sigma = (R_i)_{i \in I}$ (we conflate the names of relations in σ with their interpretations in $\mathfrak{M}$). The finite set M is the (potentially empty) domain of $\mathfrak{M}$ and each relation $R_i \subseteq M^{r_i}$ has a prescribed arity $r_i > 0$. The size $|\mathfrak{M}|$ of $\mathfrak{M}$ is the cardinality of its domain. A class $\mathcal{K}$ of structures is a set of such finite structures sharing the same signature σ and closed under isomorphism.

▶ **Example 4** (Finite Words)**.** Let Σ be a finite alphabet; we write Σ^* for the set of finite words over Σ. Such a word w can be seen as a relational structure over $\sigma_s \stackrel{\text{def}}{=} ((P_a)_{a \in \Sigma}, <)$ with its set of positions $\{1, \ldots, |w|\}$ as domain (where $|w|$ denotes the length of w, and thus coincides with its size), and $P_a(x)$ for $x \in \{1, \ldots, |w|\}$ holds if and only if the xth symbol of w is $a \in \Sigma$, while $x < y$ has the obvious interpretation.

▶ **Example 5** (Ranked Trees). Fix a finite ranked alphabet $\mathcal{F}$, where each symbol f is mapped to a single rank $r(f) \geq 0$, and write $\mathcal{F}_r$ for the subset of $\mathcal{F}$ with rank r. A (finite, ranked, ordered) tree is defined inductively as a term $f(t_1, \ldots, t_r)$ where $f \in \mathcal{F}_r$ and $t_1, \ldots, t_r$ are trees (see e.g. [9]). The notions of node, child, parent, descendant, and ancestor relations between nodes of a tree are defined as usual. We denote by $T(\mathcal{F})$ the set of trees over $\mathcal{F}$.

A tree can be seen as a structure with its set of nodes as domain – note that this domain is never empty by definition –; several signatures are then pertinent (see [6] and the full paper). We shall focus on the signature $\sigma_T \stackrel{\text{def}}{=} ((P_f)_{f \in \mathcal{F}}, <, <_{\text{dfs}}, \sqcap)$ where, for all nodes x, y,

- $P_f(x)$ holds whenever x is labelled by f,
- $x < y$ whenever x is an ancestor of y,
- $x <_{\text{dfs}} y$ whenever x is visited before y in a depth-first, left-first traversal of the tree – this is known as the *document order* –, and
- $z = x \sqcap y$ whenever z is the *least* common ancestor of x and y, i.e. z is the unique descendant of all the nodes which are ancestors of both x and y.

Embeddings. Recall that $\mathfrak{A} \sqsubseteq \mathfrak{B}$ holds between two structures $\mathfrak{A}$ and $\mathfrak{B}$ if and only if there exists an *embedding* from $\mathfrak{A}$ to $\mathfrak{B}$, i.e. an injective mapping e from A to B that preserves the relations: for all $i \in I$ and $a_1, \ldots, a_{r_i} \in A$, $(a_1, \ldots, a_{r_i}) \in R_i$ in $\mathfrak{A}$ if and only if $(e(a_1), \ldots, e(a_{r_i})) \in R_i$ in $\mathfrak{B}$. As $\sqsubseteq$ is transitive and reflexive, $(\mathcal{K}, \sqsubseteq)$ is a *quasi-order* (a qo).

Well-Quasi-Orders. Given a qo $(X, \leq)$ and a subset $S \subseteq X$, the *downward-closure* of S is $\downarrow S \stackrel{\text{def}}{=} \{x \in X \mid \exists s \in S \,.\, x \leq s\}$. A subset $D \subseteq X$ is *downwards-closed* if $D = \downarrow D$; when D is a singleton $\{x\}$ we write more succinctly $\downarrow x$. The notions of *upward-closure* and *upwards-closed* subsets are defined similarly.

A qo $(X, \leq)$ is a *well-quasi-order* (wqo) if in every infinite sequence $x_0, x_1, \ldots$ of elements of X, one can find an infinite sequence of indices $i_0 < i_1 < \cdots$ such that $x_{i_0} \leq x_{i_1} \leq \cdots$ [18, 20]. Equivalently, $(X, \leq)$ is well-founded (there are no infinite descending sequences $x_0 > x_1 > \cdots$) and satisfies the *finite antichain condition* (FAC), i.e. all its antichains are finite. Still equivalently, $(X, \leq)$ has the *descending chain condition*: there are no infinite descending sequences $D_0 \supsetneq D_1 \supsetneq \cdots$ of downwards-closed subsets $D_i \subseteq X$. For instance, any finite set ordered by equality is a wqo by the Pigeonhole Principle. As another example, if $(X, \leq_X)$ and $(Y, \leq_Y)$ are two wqos, then by Dickson's Lemma, their Cartesian product $X \times Y$ is well-quasi-ordered by the *product ordering* defined by $(x, y) \leq (x', y')$ if and only if $x \leq_X x'$ and $y \leq_Y y'$.

▶ **Example 6** (Scattered Subword Embedding). The signature σ_s of Example 4 for finite words in Σ^* gives rise to an embedding relation $\sqsubseteq_s$ known as the (scattered) *subword embedding*: $u \sqsubseteq_s v$ if and only if $u = a_1 \cdots a_m$ and $v = v_0 a_1 v_1 \cdots v_{m-1} a_m v_m$ for some $a_1, \ldots, a_m \in \Sigma$ and $v_0, \ldots, v_m \in \Sigma^*$.

By Higman's Lemma [18], the subword embedding relation $\sqsubseteq_s$ well-quasi-orders Σ^*.

▶ **Example 7** (Homeomorphic Tree Embedding). Using the signature σ_T of Example 5 for trees in $T(\mathcal{F})$, the ordering $t \sqsubseteq_T t'$ is better known as the *homeomorphic tree embedding* relation, and holds for $t = f(t_1, \ldots, t_r)$ and $t' = g(t'_1, \ldots, t'_s)$ if and only if

- there exists $1 \leq i \leq s$ such that $t \sqsubseteq_T t'_i$, or
- $f = g$ (and thus $r = s$) and for every $1 \leq i \leq r$, $t_i \sqsubseteq_T t'_i$;

see the full paper for a proof of this folklore result. Note that $(\Sigma^*, \sqsubseteq_s)$ is isomorphic to $(T(\mathcal{F}_\Sigma), \sqsubseteq_T)$ where $\mathcal{F}_\Sigma \stackrel{\text{def}}{=} \Sigma \uplus \{\$\}$ assigns rank 1 to letters in Σ and rank 0 to \$, so our results on ranked trees properly generalise the case of finite words.

As first shown by Higman [18], the homeomorphic tree embedding relation $\sqsubseteq_T$ similarly well-quasi-orders $T(\mathcal{F})$, and Kruskal's Tree Theorem [20] shows that this generalises to unranked trees.

Existential First-Order Logic. Consider first-order logic over the signature σ, and a class of structures $\mathcal{K}$ over σ. Given a sentence φ, the set of structures $\mathfrak{M} \in \mathcal{K}$ such that $\mathfrak{M} \models \varphi$ is called the (first-order) *language* of φ. It is well-known that *existential sentences* in $\Sigma_1(\sigma)$ define *upwards-closed* first-order languages S with respect to embeddings, i.e. such that $S = \uparrow S \stackrel{\text{def}}{=} \{\mathfrak{M}' \in \mathcal{K} \mid \exists \mathfrak{M} \in S . \mathfrak{M} \sqsubseteq \mathfrak{M}'\}$; however the converse does not necessarily hold [28].

We are interested here in the Boolean closure $\mathcal{B}\Sigma_1(\sigma)$ of $\Sigma_1(\sigma)$:

▶ **Fact 8.** *If a set S is an n-PTL over $(\mathcal{K}, \sqsubseteq)$, then it is definable by a sentence in $\mathcal{B}\Sigma_1(\sigma)$ with at most n variables. Conversely, assuming additionally that $(\mathcal{K}, \sqsubseteq)$ is a wqo, if S is definable by a sentence in $\mathcal{B}\Sigma_1(\sigma)$, then it is a PTL over $(\mathcal{K}, \sqsubseteq)$.*

3 Ideal Characterisation of PTL Separability

A set $S \subseteq X$ *separates* $L \subseteq X$ from $L' \subseteq X$ if $L \subseteq S$ and $S \cap L' = \emptyset$. This can be turned into a decision problem when restricting L and L' to a class $\mathcal{C} \subseteq 2^X$ of finitely representable sets:

▶ **Problem** (PTL separability for $\mathcal{C}$ over $(X, \leq)$).
Input: *Representations of L and L' from $\mathcal{C}$*
Question: *Are L and L'* PTL separable*, i.e. does there exist S a PTL over $(X, \leq)$ that separates L from L'?*

Throughout this section, we assume that $(X, \leq)$ is a well-quasi-order. Under this assumption, we establish in Section 3.2 a characterisation of PTL separability in terms of *ideals* of $(X, \leq)$ (whose definition we recall in Section 3.1). This yields a generic framework in which the decidability of PTL separability can be tackled, which we present in Sections 4 and 5 and instantiate for the case of $(T(\mathcal{F}), \sqsubseteq_T)$ the set of ranked trees together with homeomorphic embeddings against the class $\mathcal{C} = \mathrm{Reg}(T(\mathcal{F}))$ of regular tree languages.

3.1 Ideals

An *ideal* of a qo $(X, \leq)$ is a downwards-closed and (up-)*directed* subset $I \subseteq X$, where this last condition ensures that I is non-empty and that, given any $x \in I$ and $y \in I$, there exists $z \in I$ with $x \leq z$ and $y \leq z$. A related notion is the following. A downwards-closed subset D of $(X, \leq)$ is *irreducible* if and only if it is non-empty, and for any two downwards-closed subsets D_1, D_2 such that $D \subseteq D_1 \cup D_2$, D is contained in D_1 or in D_2 already; equivalently, D is non-empty and cannot be written as the union of two proper, downwards-closed subsets.

▶ **Fact 9** (cf. Lemma 1 in [7]). *The following are equivalent for a downwards-closed subset D of a quasi-ordered set: (a) D is an ideal; (b) D is directed; (c) D is irreducible.*

We write $\mathrm{Idl}(X)$ for the set of ideals of X. Ideals are especially useful when $(X, \leq)$ is a wqo: for one thing, when X is countable, $\mathrm{Idl}(X)$ is then also countable [7, Theorem 1], and furthermore any downwards-closed subset has a decomposition as a finite union of ideals:

▶ **Fact 10** (cf. Lemma 2 in [7]). *A qo is FAC if and only if every downwards-closed subset is a finite union of ideals.*

3.2 Characterising Separability

Over any qo $(X, \leq)$, the case where L and L' are not PTL separable has a well-known characterisation in terms of sequences of indistinguishable witnesses, which is a straightforward consequence of the definitions:

▶ **Lemma 11.** *L and L' are* not *PTL separable over a qo $(X, \leq)$ if and only if there exist two sequences of elements $(x_n)_{n\in\mathbb{N}}$ in L and $(x'_n)_{n\in\mathbb{N}}$ in L' such that for every n, $x_n \equiv_n x'_n$.*

Proof Idea. It suffices to observe that L and L' are not n-PTL separable if and only if there exist $x_n \in L$ and $x'_n \in L'$ such that $x_n \equiv_n x'_n$. See the full paper for a detailed proof. ◀

When $(X, \leq)$ is a wqo, we have another characterisation in terms of directed subsets of L and L' defining the same ideal. The fact that $(X, \leq)$ is wqo is needed in order for Lemma 12 to hold: otherwise, the full paper presents a counter-example for the *subtree ordering* over $T(\mathcal{F})$.

▶ **Lemma 12** (Key Lemma)**.** *L and L' are not PTL separable over a wqo $(X, \leq)$ if and only if there exist two directed sets $\Delta \subseteq L$ and $\Delta' \subseteq L'$ such that $\downarrow\Delta = \downarrow\Delta'$.*

Proof of 'if'. Let $\Delta \subseteq L$ and $\Delta' \subseteq L'$ be two directed sets with $\downarrow\Delta = \downarrow\Delta'$. Let us show that for every $n \in \mathbb{N}$, there exist $x_n \in L$ and $x'_n \in L'$ such that $x_n \equiv_n x'_n$, from which Lemma 11 yields the result. Write I for the ideal $\downarrow\Delta = \downarrow\Delta'$. Observe that, for every $n \in \mathbb{N}$, $I \cap X_{\leq n}$ is finite since X is a combinatorial class. Furthermore, for every $z \in I \cap X_{\leq n}$, by definition of I there exist $x_z \in \Delta$ with $z \leq x_z$ and $x'_z \in \Delta'$ with $z \leq x'_z$. Since Δ and Δ' are directed, we can find $x_n \in \Delta$ and $x'_n \in \Delta'$ greater or equal to all those finitely many x_z and x'_z when z ranges over $I \cap X_{\leq n}$ (if $I \cap X_{\leq n} = \emptyset$ then any $x_n \in \Delta$ and $x'_n \in \Delta'$ fit). Then $x_n \equiv_n x'_n$, since for any $z \in X_{\leq n}$, either $z \in I$ and then both $z \leq x_z \leq x_n$ and $z \leq x'_z \leq x'_n$, or $z \notin I$ and then both $z \not\leq x_n$ and $z \not\leq x'_n$ since I is downwards-closed. ◀

Proof of 'only if'. Assume L and L' are not PTL separable, hence by Lemma 11 there exist two infinite sequences $(x_n)_n$ in L and $(x'_n)_n$ in L' with $x_n \equiv_n x'_n$ for every n.

Let us consider the infinite sequence of *pairs* $(x_n, x'_n)_{n\in\mathbb{N}}$. By Dickson's Lemma, $X \times X$ is a wqo for the product ordering, hence there exists an infinite sequence of indices $i_0 < i_1 < \cdots$ such that $x_{i_j} \leq x_{i_{j+1}}$ and $x'_{i_j} \leq x'_{i_{j+1}}$ for every $j \in \mathbb{N}$. Define $\Delta \stackrel{\text{def}}{=} \{x_{i_j} \mid j \in \mathbb{N}\}$ and $\Delta' \stackrel{\text{def}}{=} \{x'_{i_j} \mid j \in \mathbb{N}\}$. These two sets are directed; they are actually infinite chains $x_{i_0} \leq x_{i_1} \leq \cdots$ and $x'_{i_0} \leq x'_{i_1} \leq \cdots$.

It remains to show that $\downarrow\Delta = \downarrow\Delta'$. By symmetry, it suffices to show $\Delta \subseteq \downarrow\Delta'$. Consider some $x_{i_j} \in \Delta$; then there exists some index $i_k > \max(i_j, |x_{i_j}|)$. Hence $x_{i_j} \leq x_{i_k}$, and since $x_{i_k} \equiv_{i_k} x'_{i_k}$, $x_{i_j} \leq x'_{i_k}$ and thus $x_{i_j} \in \downarrow\Delta'$. ◀

Related Work over Finite Words. Let S be a subset of X. We define the *adherence* of S as the set of ideals defined by the directed subsets of S:

$$\mathrm{Adh}(S) \stackrel{\text{def}}{=} \{\downarrow\Delta \in \mathrm{Idl}(X) \mid \Delta \subseteq S \text{ is directed } \}\,. \tag{1}$$

Lemma 12 can be restated as saying that L and L' are PTL separable if and only if their adherences are disjoint, i.e. $\mathrm{Adh}(L) \cap \mathrm{Adh}(L') = \emptyset$. This makes Lemma 12 quite reminiscent of several results on separability for word languages over Σ^*. Almeida [2] showed that two regular languages over Σ are PTL separable over $(\Sigma^*, \sqsubseteq_s)$ if and only if their topological closures inside a specific profinite semigroup do not intersect. This lead to a first decision procedure [3] by explicitly constructing representations for these topological closures and

testing their intersection for emptiness – the counterpart in terms of Lemma 12 would be to compute the adherences of L and L'.

Major improvements were brought by Place et al. [24] and Czerwiński et al. [11] and culminate in Theorem 2.1 of [12], by reducing this non-empty intersection check to identifying a common pattern 'densely' matched by both L and L'. It turns out that these patterns are essentially finite representations for ideals of $(\Sigma^*, \sqsubseteq_s)$, so Lemma 12 subsumes Theorem 2.1 of Czerwiński et al. [12] – and has a considerably simpler proof.

4 Deciding PTL Separability

While Lemma 12 provides a general characterisation for PTL separability, turning it into a decision procedure requires finite representations and effectiveness assumptions on its various ingredients. We define a set of such assumptions in Section 4.1, which is sufficient to derive a generic algorithm. We describe the latter in two steps: we first show in Section 4.2 a reduction to the *adherence membership problem*. The final step in Section 4.3 is to show that this last problem is decidable for the set of regular tree languages over $(T(\mathcal{F}), \sqsubseteq_T)$.

4.1 Effectiveness Assumptions

In order to put Lemma 12 into practice, we need to consider in more details how we are going to represent PTLs over $(X, \leq)$, ideals in $\mathrm{Idl}(X)$, and languages in $\mathcal{C}$.

PTLs. Fact 3 provides a natural representation for PTLs as finite Boolean combinations of principal filters, i.e. more concretely as terms of the free Boolean algebra with elements of X as atoms.

Ideals. Recall that over a countable wqo $(X, \leq)$, $\mathrm{Idl}(X)$ is also countable [7]. In Section 4.2, we only need to have explicit *ideal representations* as a means of enumerating ideals, while Corollary 15 further needs a means of computing representations as regular tree languages. Section 5 fulfils both requirements by providing ideal representations for $(T(\mathcal{F}), \sqsubseteq_T)$ as regular tree expressions.

Languages. Our last effectiveness assumptions regard the class of languages $\mathcal{C}$. We call $\mathcal{C}$ *PTL-effective* over a qo $(X, \leq)$ if $\mathcal{C}$ has *decidable emptiness*: given a representation for $L \in \mathcal{C}$, there is an algorithm that decides whether $L = \emptyset$, and if $\mathcal{C}$ is *effectively closed under intersection with PTLs*: given a representation for $L \in \mathcal{C}$ and one for S a PTL over $(X, \leq)$, there is an algorithm that constructs a representation for $L \cap S$ in $\mathcal{C}$.

For instance, both over $(\Sigma^*, \sqsubseteq_s)$ and over $(T(\mathcal{F}), \sqsubseteq_T)$, a principal filter $\uparrow x$ is a regular language, and since regular languages are closed under Boolean operations by Fact 3 any PTL is regular. Thus, PTL-effective classes of word and tree languages abound, starting with regular languages themselves, but also context-free, etc.

4.2 Reducing to Adherence Membership

We describe our decision procedure in two steps. The first one reduces the PTL separability problem for a PTL-effective class $\mathcal{C}$ to the following problem:

▶ **Problem** (Adherence Membership for $\mathcal{C}$ over $(X, \leq)$).
Input: *A representation for* $L \in \mathcal{C}$ *and one for* $I \in \mathrm{Idl}(X)$.
Question: *Is* $I \in \mathrm{Adh}(L)$*?*

▶ **Proposition 13.** *Let $(X, \leq)$ be a wqo with ideal representations and $\mathcal{C} \subseteq 2^X$ be PTL-effective over $(X, \leq)$. Then there is a Turing reduction from the PTL separability problem to the adherence membership problem.*

Proof. Given an oracle for the adherence membership problem for $\mathcal{C}$ over $(X, \leq)$, our algorithm consists of two semi-decision procedures that take representations for $L \in \mathcal{C}$ and $L' \in \mathcal{C}$ as input. The first attempts to show that L and L' are PTL separable, and enumerates representations of PTLs S until $L \cap (X \setminus S) = \emptyset$ and $L' \cap S = \emptyset$. This is possible because $\mathcal{C}$ is PTL-effective and complementing a PTL representation is trivial. The second relies on Lemma 12 and attempts to show that L and L' are not PTL separable by enumerating representations of ideals I until $I \in \mathrm{Adh}(L)$ and $I \in \mathrm{Adh}(L')$, using the oracle for adherence membership for the tests. ◀

4.3 Deciding Adherence Membership

Regular Languages. In the case of regular tree languages over $T(\mathcal{F})$, the adherence membership problem is decidable thanks to the following lemma:

▶ **Lemma 14.** *Let $(X, \leq)$ be a qo and $L \subseteq X$. Then $I \in \mathrm{Adh}(L)$ if and only if $I \subseteq {\downarrow}(I \cap L)$.*

Proof. The 'only if' part is immediate or the 'if' part we show that $I \cap L$ is directed. Since I is non-empty and included in ${\downarrow}(I \cap L)$, $I \cap L$ is also non-empty. Furthermore, if x, y are in $I \cap L$, then since I is directed there exists $z \in I$ such that $x \leq z$ and $y \leq z$, and since $I \subseteq {\downarrow}(I \cap L)$ there exists $z' \in I \cap L$ such that $z \leq z'$. ◀

Now, if L is a regular tree language, I is also effectively regular using the ideal representations from Section 5, and so are $I \cap L$ and ${\downarrow}(I \cap L)$, hence $I \subseteq {\downarrow}(I \cap L)$ is decidable since inclusion of regular tree languages is decidable, proving Theorem 1:

▶ **Corollary 15.** *Adherence membership is decidable for regular tree languages over $(T(\mathcal{F}), \sqsubseteq_T)$.*

Generic Approach. Let us finally generalise the previous idea. Our issue is that, for instance when $\mathcal{C}$ is the class of context-free languages over Σ^*, while I is a regular language and ${\downarrow}(I \cap L)$ is a computable context-free language, the inclusion test between a regular language and a context-free one is in general undecidable. Thankfully, ${\downarrow}(I \cap L)$ is regular for arbitrary languages L by Fact 10, since it is a finite union of ideals and ideals are regular. However, the issue is then to compute a representation of ${\downarrow}(I \cap L)$ *as a regular language*, or more generally to solve the following problem:

▶ **Problem** (Ideal Decomposition for $\mathcal{C}$ over $(X, \leq)$)**.**
Input: *A representation for $L \in \mathcal{C}$*
Output: ${\downarrow}L$ *as a finite union of representations of ideals in* $\mathrm{Idl}(X)$.

The following result requires effective operations on ideal representations. We refer the reader to [13, 15] for a systematic study of algorithms on finite ideal representations for wqos; here we use the notion of *effective* ideal representations as defined by Goubault-Larrecq et al. [15] to prove (see the full paper):

▶ **Proposition 16.** *Let $(X, \leq)$ be a wqo with effective ideal representations and $\mathcal{C} \subseteq 2^X$ be PTL-effective over $(X, \leq)$. Then the adherence membership problem and the ideal decomposition problem are Turing-equivalent.*

The seemingly innocuous hypothesis that $\mathcal{C}$ is PTL-effective is actually crucial: Theorem V.2 and Theorem VIII.1 in [22] provide an instance of undecidable adherence membership but computable ideal decompositions over a wqo with effective ideal representations; in these results, $\mathcal{C}$ is the set of run structures between two configurations of a vector addition system, and has decidable emptiness but lacks effective closure under intersection with PTLs.

Related Work over Finite Words. Propositions 13 and 16 together show that PTL separability reduces to the computation of ideal decompositions for downward-closures over any wqo with effective ideal representations. This includes as a special case $(\Sigma^*, \sqsubseteq_s)$ using the representations in [19, 1], and since ideal decompositions of downward-closures are computable for many classes of PTL-effective word languages [30] – including context-free languages [29, 10], reachability languages of vector addition systems [16], matrix languages [30], and higher-order languages [17, 8, 4] –, PTL separability is decidable for them.

In fact, propositions 13 and 16 subsume most of Theorem 2.5 of Czerwiński et al. [12], which is stated for *full trios* $\mathcal{C}$ over Σ^*, i.e. classes of languages closed under rational transductions – which are thus effectively closed under intersection with PTLs. There is a small price to pay for our level of generality, which is that their Theorem 2.5 also shows a converse reduction over $(\Sigma^*, \sqsubseteq_s)$ from the ideal decomposition problem back to the PTL separability problem. In our general setting, the best we know is that $L \stackrel{\text{def}}{=} I$ and $L' \stackrel{\text{def}}{=} X \setminus {\downarrow}(I \cap S)$ are PTL-separable if and only if $I \in \mathrm{Adh}(S)$ by Lemma 14, but this either uses the closure of $\mathcal{C}$ under complementation and downward-closure, or already uses computable ideal decompositions. The former however holds for regular languages:

▶ **Proposition 17.** *There is a many-one reduction from the adherence membership problem to the PTL separability problem for regular tree languages over* $(T(\mathcal{F}), \sqsubseteq_T)$.

5 Ideals for Ranked Trees with Homeomorphic Embedding

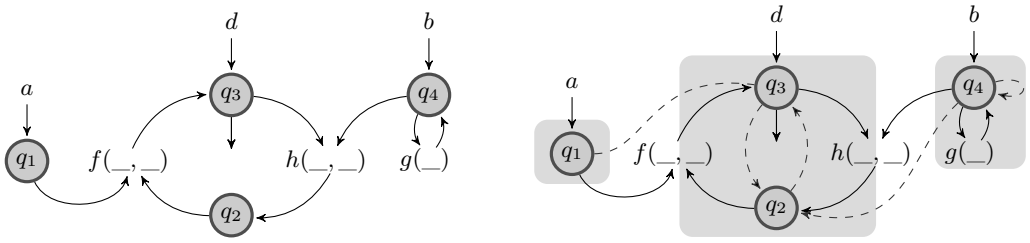
In this section, we provide finite representations for ideals of ranked trees ordered by homeomorphic embedding. These representations are expressed as tree regular expressions [9, Section 2.2]. We show in Section 5.1 that any downwards-closed subset of $T(\mathcal{F})$ can be represented as a *simple tree regular expression* (STRE), which we construct from its tree automaton. In Section 5.2, we then characterise ideals in this syntax as *tree products*, obtained as the summands of the normal form according to a rewrite system. Thanks to this particular proof strategy, the entire construction also solves the ideal decomposition problem: given a regular tree language, first build the STRE for its downward-closure, then normalise this STRE into a union of tree products representing its ideals.

Note that a concrete syntax for ranked tree ideals was proposed in [13]. However, no proof was given, and indeed the proposed tree regular expressions failed to be downwards-closed.

5.1 Simple Tree Regular Expressions

Let $L \subseteq T(\mathcal{F})$ be a downwards-closed language. Its complement is upwards-closed and has finitely many minimal elements, i.e. $T(\mathcal{F}) \setminus L = {\uparrow}\{t_1, \ldots, t_n\} = \bigcup_{1 \leq i \leq n} {\uparrow} t_i$, hence this is a regular tree language and L is also regular. Thus L is recognised by a finite (bottom-up) tree automaton $\mathcal{A}$, or equivalently by a tree regular expression [9, Section 2.2].

We shall see that $\mathcal{A}$ is equivalent to an expression of a specific shape, called a *simple tree regular expression* (STRE). We describe next a general procedure that converts any (ε-free) finite tree automaton $\mathcal{A}$ to an STRE S whose language is the downward-closure ${\downarrow} L(\mathcal{A})$ of the



(a) Initial tree automaton. **(b)** ε-strongly connected components.

Figure 1 Converting tree automata to STREs.

language recognised by $\mathcal{A}$. This procedure is best explained on an example: see Figure 1a, where there is one 0-ary transition a (from no state) to state q_1, one binary transition f from the pair of states q_1, q_2 to q_3, and so on. In textual form, we write these transitions as rewrite rules [9]: $a \to q_1$, $f(q_1, q_2) \to q_3$, $h(q_3, q_3) \to q_2$, $c \to q_3$, $g(q_4) \to q_4$, $b \to q_4$.

We first extend our automaton with ε-transitions. An ε-transition from s to s' will be drawn as a dashed arrow (see Figure 1b), and is just a rewrite rule of the new form $s \to s'$. This implies that every tree recognised at s is also recognised at s'. For each transition, say $f(s_1, s_2, \cdots, s_n) \to s$, of $\mathcal{A}$, we add n ε-transitions $s_1 \to s$, $s_2 \to s$, ..., $s_n \to s$. Call the resulting automaton $\downarrow\mathcal{A}$. It is an easy exercise to show that $L(\downarrow\mathcal{A}) = \downarrow L(\mathcal{A})$.

There is a graph underlying $\downarrow\mathcal{A}$, whose vertices are the states of $\downarrow\mathcal{A}$, and whose edges are the ε-transitions. Consider its strongly connected components, shown against a grey background in Figure 1b. Any two states in the same strongly connected component C recognise exactly the same trees, so it makes sense to talk of the language $L_C(\downarrow\mathcal{A})$ of those trees recognised at any state of C. Let $C \to C'$ if and only if $s \to s'$ for some $s \in C$, $s' \in C'$. The strict ordering $\prec \stackrel{\text{def}}{=} \to^+$ is well-founded, and we shall build an expression S_C whose language is $L_C(\downarrow\mathcal{A})$ by induction along $\prec$.

Trivial Components. If C is a trivial strongly connected component (one state s, no self-edge), then enumerate its incoming non-ε transitions $f_i(s_{i1}, s_{i2}, \cdots, s_{in_i}) \to s$, $1 \leq i \leq m$. Let S_{ij} be an expression whose language is the set of trees recognised at s_{ij}, which is given by induction hypothesis. Then $S_C \stackrel{\text{def}}{=} P_1 + \cdots + P_m$ with $P_i \stackrel{\text{def}}{=} f_i^?(S_{i1}, S_{i2}, \cdots, S_{in_i})$ is our desired expression. For instance, the set of trees recognised at the leftmost state q_1 of Figure 1b is the language of $S_1 \stackrel{\text{def}}{=} a^?$.

Here, a tree t is in the language of an expression $P = f^?(S_1, \ldots, S_n)$ (written '$t \in P$') for $f \in \mathcal{F}_n$ and expressions $S_1, \ldots, S_n$ if and only if either t is of the form $f(t_1, \cdots, t_n)$ with $t_i \in S_i$ for every i, $1 \leq i \leq n$, *or* if $t \in \bigcup_{i=1}^n S_i$; as the notation suggests, for $S = P_1 + \cdots + P_m$, $t \in S$ if and only if $t \in P_j$ for some j, $1 \leq j \leq m$.

Iterators. If C is a non-trivial strongly connected component, then enumerate the non-ε transitions $f_i(s_{i1}, s_{i2}, \cdots, s_{in_i}) \to s_i$, $1 \leq i \leq m$, whose end state s_i is in C. For each pair i, j, if s_{ij} is in C, then $S_{\Box ij} \stackrel{\text{def}}{=} \Box$ is a placeholder; otherwise, let $S_{\Box ij}$ be an expression whose language is the set of trees recognised at s_{ij}, which we obtain by induction hypothesis. Then $S_C \stackrel{\text{def}}{=} (A_1 + \cdots + A_m)^*.0$ for $A_i \stackrel{\text{def}}{=} f_i(S_{\Box i1}, S_{\Box i2}, \cdots, S_{\Box in_i})$ is a suitable expression. For example, the rightmost strongly connected component $\{q_4\}$ of Figure 1b yields the expression $S_4 \stackrel{\text{def}}{=} (b + g(\Box))^*.0$. One might have expected an expression of the more intuitive form $(g(\Box))^*.b^?$, however, as we are going to see, they define exactly the same language.

Here, 0 denotes the empty sum with empty language, and intuitively the language of an *iterator* $C^*.S$ for $C = A_1 + \cdots + A_m$ should consist of all trees obtained by repeatedly applying *contexts* from $A_1, \ldots, A_m$ – i.e. trees over the extended alphabet $\mathcal{F} \cup \{\Box\}$ where $\Box$ has arity 0 –, until one reaches a tree in S. More precisely, the language of an *atom* $A = f(S_{\Box 1}, \cdots, S_{\Box n})$ consists of those contexts in $T(\mathcal{F} \cup \{\Box\})$ of the form $f(c_1, \cdots, c_n)$ where $c_i \in S_{\Box i}$ for every i. In turn, $c \in S_\Box$ if and only if either $S_\Box = \Box$ and c is the trivial context $\Box$, or $S_\Box$ is an expression S, c is a tree t in $T(\mathcal{F})$, and $t \in S$. The language of $C = A_1 + \cdots + A_m$ is the union of the languages of A_j, $1 \leq j \leq m$. For example, $(f(\Box))^*.a^?$ will recognise all trees of the form $f^n(a)$, $n \in \mathbb{N}$. There are however two catches:

1. The first one has to do with atoms A where the placeholder $\Box$ occurs more than once: as usual with tree regular expressions, when replacing $\Box$ by a tree from S, several occurrences of $\Box$ can be replaced by *different* trees from S. Hence $(f(\Box, \Box))^*.(a^? + b^?)$ consists of all binary-branching trees with inner nodes labelled f and leaves labelled a or b, including $f(f(a, a), a)$ and $f(f(b, b), b)$ but also $f(f(a, b), a))$ or $f(f(b, b), a)$ among others. For a context c, and a set of trees S, we shall write $c[S]$ for the set of trees obtained from c by replacing each occurrence of $\Box$ by a (possibly different) tree in S.
2. The second catch has to do with downward-closure. It is tempting to define the trees of $C^*.S$ as those in $c_1[\cdots[c_k[S]]\cdots]$, for some $k \in \mathbb{N}$ and some $c_1, \cdots, c_k \in C$. However, there are cases where that language would fail to be downwards-closed, e.g., $(f(a^?, \Box))^*.b^?$ would contain $f(a, b)$ but not a, according to that semantics.
   We repair that as follows. For $A = f(S_{\Box 1}, \cdots, S_{\Box n})$, define $\operatorname{supp} A$ as follows: if those $S_{\Box i}$, $1 \leq i \leq n$, that are different from $\Box$ define non-empty languages, then $\operatorname{supp} A$ is the union of those languages; if some $S_{\Box i} \neq \Box$ has an empty language, then $\operatorname{supp} A = \emptyset$. Hence, for example, $\operatorname{supp} f(\Box, \Box) = \emptyset$, $\operatorname{supp} f(a^?, \Box) = a^? = \{a\}$, and $\operatorname{supp} f'(a^?, \Box, 0) = \emptyset$. For $C = A_1 + \cdots + A_m$, let $\operatorname{supp} C = \bigcup_{j=1}^m \operatorname{supp} A_j$.

We are now ready to define the language of $C^*.S$, as the language of trees in $c_1[\cdots[c_k[S \cup \operatorname{supp} C]]\cdots]$, for some $k \in \mathbb{N}$ and some $c_1, \cdots, c_k \in C$.

Finally, $\downarrow L(\mathcal{A})$ is the union of the languages of the strongly connected components containing a final state of $\mathcal{A}$; in our example the strongly connected component in the middle yields the final expression $(d + f(S_1, \Box) + h(\Box, S_4))^*.0$.

General Syntax. Summing up, we define *simple tree regular expressions* (STREs) by the following abstract syntax:

$$S ::= P_1 + \cdots + P_m \qquad P ::= f^?(S, \cdots, S) \mid C^*.S$$
$$C ::= A + \cdots + A \qquad A ::= f(S_\Box, \cdots, S_\Box) \qquad S_\Box ::= S \mid \Box$$

where $f \in \mathcal{F}_r$ in $f^?(S_1, \cdots, S_r)$ and in $f(S_{\Box 1}, \cdots, S_{\Box r})$, the sum operation $+$ is associative and commutative (we shall sometimes write $\sum_{i=1}^m P_i$ for $P_1 + \cdots + P_m$) with 0 denoting the empty sum, and $\Box \notin \mathcal{F}$ is a placeholder. Note that $\Box$ is not meant to denote a family of placeholders, rather a single one. The STREs of the form P are called *tree pre-products*. Among them, the *tree products* will be our notations for ideals, see Section 5.2.

▶ **Proposition 18.** *Every STRE defines a downwards-closed language of $(T(\mathcal{F}), \sqsubseteq_T)$. Every downwards-closed language of $(T(\mathcal{F}), \sqsubseteq_T)$ is the language of some STRE.*

Proof. The first part can be shown by structural induction on STREs; see the full paper for details. The second part was sketched above. ◀

$$
\begin{array}{rcll}
P + P' & \to_1 & P' & \text{if } P \subseteq P' \\
0 + P & \to_1 & P & \\
0^*.S & \to_1 & S & \\
f^?(\vec{S}_1, 0, \vec{S}_2) & \to_1 & 0 & \\
f(\vec{S}_{\Box 1}, 0, \vec{S}_2) & \to_1 & 0 & \\
C^*.0 & \to_1 & 0 & \\
C^*.(S + S') & \to_1 & C^*.S + C^*.S' & \text{if } C \text{ is } \Box\text{-linear}
\end{array}
\qquad
\begin{array}{rcll}
A + A' & \to_1 & A' & \text{if } A \subseteq A' \\
0 + A & \to_1 & A & \\
(C + f(S_1, \cdots, S_n))^*.S & \to_1 & C^*.(S + f^?(S_1, \cdots, S_n)) & \\
f^?(\vec{S}_1, S + S', \vec{S}_2) & \to_1 & f^?(\vec{S}_1, S, \vec{S}_2) + f^?(\vec{S}_1, S', \vec{S}_2) & \\
f(\vec{S}_{\Box 1}, S_\Box + S'_\Box, \vec{S}_{\Box 2}) & \to_1 & f(\vec{S}_{\Box 1}, S_\Box, \vec{S}_{\Box 2}) + f(\vec{S}_{\Box 1}, S'_\Box, \vec{S}_{\Box 2}) & \\
\multicolumn{4}{c}{\text{if } C = \sum_{i=1}^m f_i(\Box, \cdots, \Box) \text{ and no } f_i \text{ is 0-ary}}
\end{array}
$$

Figure 2 The rewrite relation $\to_1$.

5.2 Tree Products

We now characterise the STREs that define ideals of $(T(\mathcal{F}), \sqsubseteq_T)$. We define a rewrite relation $\to_1$ on STREs that moves all $+$ signs to the outside: for a $\to_1$-normal STRE $S = P_1 + \cdots + P_m$, each P_i will be irreducible, hence S will be an ideal if and only if $m = 1$ by Fact 9.

The rewrite relation $\to_1$ is defined in Figure 2. Recall that $+$ is understood modulo associativity and commutativity. Letters matter, too: S, S', S_1, …, S_n are STREs, while P, P' are those special STREs of the form $f^?(S_1, \cdots, S_n)$ or $C^*.S$, etc. In particular, the third rule of the second column applies provided the pattern $f(S_1, \cdots, S_n)$ does not contain $\Box$ at all. In the first rules, note that inclusion of STREs is decidable. For the two bottom rules, we need some auxiliary definitions. Say that a pattern $A = f(S_{\Box 1}, \cdots, S_{\Box n})$ is $\Box$-*linear* if and only if at most one $S_{\Box i}$ is the placeholder $\Box$. Writing C as $A_1 + \cdots + A_m$, we say that C is $\Box$-linear if and only if every non-empty A_i is $\Box$-linear. The $\Box$-linearity restriction imposed on the last two rules is needed for the following to hold.

▶ **Fact 19.** *If $S \to_1^* S'$ then S and S' define the same language.*

▶ **Lemma 20.** *Every STRE S has a normal form with respect to $\to_1$.*

Proof. Using Bachmair and Plaisted's associative path ordering $>_{apo}$ [5] on a precedence where $+$ is minimal, $f > f^?$ for each symbol f, and the $(\_)^*.\_$ operator has lexicographic status, we see that $\to_1$ is even a terminating relation. ◀

▶ **Definition 21.** A *tree product* is any $\to_1$-normal tree pre-product P.

▶ **Lemma 22.** *Every ideal of $T(\mathcal{F})$ is the language of some tree product.*

Proof. By Proposition 18, an ideal I is the language of some STRE S, which has a $\to_1$-normal form by Lemma 20; write it $P_1 + \cdots + P_m$. Since I is non-empty, $m \geq 1$, and since I is irreducible (Fact 9), it is included in, hence equal to, the language of some P_i. ◀

Conversely, the language of any tree product is directed (see the full paper for a proof), thus:

▶ **Theorem 23.** *The ideals of $T(\mathcal{F})$ are exactly the languages of tree products.*

6 Concluding Remarks

We have presented a general order-theoretic characterisation of PTL separability, and shown that it could be applied to decide PTL separability of languages beyond finite words, namely of ranked regular tree languages ordered by homeomorphic embedding. Our work further adds

to the growing body of algorithmic applications of downwards-closed sets and ideals of well-quasi-orders in logic and verification, e.g. in forward analysis [13, 14], backward analysis [21], inference of inductive invariants [23], and reachability in vector addition systems [22].

We are confident our techniques apply to unranked trees, by defining suitable ideal representations. In the same vein, it would be interesting to develop ideal representations for the tree minor ordering, and to try to decide PTL separability for context-free tree languages.

An open-ended question is how to finely relate our order-theoretic characterisation with the algebraic and topological characterisations typically employed for the definability and separability problems. For instance, can one derive the characteristic equations of Bojańczyk et al. [6] for piecewise-testable tree languages from Lemma 12?

Acknowledgements. The authors thank Wojciech Czerwiński, Luc Segoufin, and Georg Zetzsche for their insightful comments, and an anonymous reviewer for correcting Fact 8.

References

1 Parosh Aziz Abdulla, Aurore Collomb-Annichini, Ahmed Bouajjani, and Bengt Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004. doi:10.1023/B:FORM.0000033962.51898.1a.

2 Jorge Almeida. Some algorithmic problems for pseudovarieties. *Publicationes Mathematicae Debrecen*, 54(suppl.):531–552, 1999.

3 Jorge Almeida and Marc Zeitoun. The pseudovariety J is hyperdecidable. *RAIRO Theoretical Informatics and Applications*, 31:457–482, 1997.

4 Kazuyuki Asada and Naoki Kobayashi. On word and frontier languages of unsafe higher-order grammars. In *ICALP 2016*, Leibniz International Proceedings in Informatics, 2016. To appear. URL: https://arxiv.org/abs/1604.01595.

5 Leo Bachmair and David A. Plaisted. Termination orderings for associative-commutative rewriting systems. *Journal of Logic and Computation*, 1(4):329–349, 1985. doi:10.1016/S0747-7171(85)80019-5.

6 Mikołaj Bojańczyk, Luc Segoufin, and Howard Straubing. Piecewise testable tree languages. *Logical Methods in Computer Science*, 8(3), 2012. doi:10.2168/LMCS-8(3:26)2012.

7 Robert Bonnet. On the cardinality of the set of initial intervals of a partially ordered set. In *Infinite and finite sets: to Paul Erdős on his 60th birthday, Vol. 1*, Coll. Math. Soc. János Bolyai, pages 189–198. North-Holland, 1975.

8 Lorenzo Clemente, Paweł Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recursion schemes is decidable. In *LICS 2016*. ACM, 2016. To appear.

9 H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. Inria, 2007. URL: http://tata.gforge.inria.fr/.

10 Bruno Courcelle. On constructing obstruction sets of words. *Bulletin of the EATCS*, 44:178–186, 1991.

11 Wojciech Czerwiński, Wim Martens, and Tomáš Masopust. Efficient separability of regular languages by subsequences and suffixes. In *ICALP 2013*, volume 7966 of *Lecture Notes in Computer Science*, pages 150–161. Springer, 2013. doi:10.1007/978-3-642-39212-2_16.

12 Wojciech Czerwiński, Wim Martens, Lorijn van Rooijen, Marc Zeitoun, and Georg Zetzsche. A characterization for decidable separability by piecewise testable languages. Preprint, 2015. An extended abstract appeared as:

W. Czerwiński, W. Martens, L. van Rooijen, and M. Zeitoun. A note on decidable separability by piecewise testable languages. In *FCT 2015*, volume 9210 of *LNCS*, pages 173–185. Springer, 2015. URL: http://arxiv.org/abs/1410.1042v2.

13 Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, part I: Completions. In *STACS 2009*, volume 3 of *Leibniz International Proceedings in Informatics*, pages 433–444. LZI, 2009. doi:10.4230/LIPIcs.STACS.2009.1844.

14 Alain Finkel and Jean Goubault-Larrecq. Forward analysis for WSTS, part II: Complete WSTS. *Logical Methods in Computer Science*, 8(3), 2012. doi:10.2168/LMCS-8(3:28)2012.

15 Jean Goubault-Larrecq, Prateek Karandikar, K. Narayan Kumar, and Philippe Schnoebelen. The ideal approach to computing closed subsets in well-quasi-orderings. In preparation, 2016. See also an earlier version in: J. Goubault-Larrecq. On a generalization of a result by Valk and Jantzen. Research Report LSV-09-09, LSV, ENS Cachan, 2009. URL: http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/PDF/rr-lsv-2009-09.pdf.

16 Peter Habermehl, Roland Meyer, and Harro Wimmel. The downward-closure of Petri net languages. In *ICALP 2010*, volume 6199 of *Lecture Notes in Computer Science*, pages 466–477. Springer, 2010. doi:10.1007/978-3-642-14162-1_39.

17 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In *POPL 2016*, pages 151–163. ACM, 2016. doi:10.1145/2837614.2837627.

18 Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(2):326–336, 1952. doi:10.1112/plms/s3-2.1.326.

19 Pierre Jullien. *Contribution à l'étude des types d'ordres dispersés*. Thèse de doctorat, Université de Marseille, 1969.

20 Joseph B. Kruskal. Well-quasi-ordering, the Tree Theorem, and Vazsonyi's Conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225, 1960. doi:10.2307/1993287.

21 Ranko Lazić and Sylvain Schmitz. The ideal view on Rackoff's coverability technique. In *RP 2015*, volume 9328 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2015. doi:10.1007/978-3-319-24537-9_8.

22 Jérôme Leroux and Sylvain Schmitz. Demystifying reachability in vector addition systems. In *LICS 2015*, pages 56–67. IEEE Press, 2015. doi:10.1109/LICS.2015.16.

23 Oded Padon, Neil Immerman, Sharon Shoham, Aleksandr Karbyshev, and Mooly Sagiv. Decidability of inferring inductive invariants. In *POPL 2016*, pages 217–231. ACM, 2016. doi:10.1145/2837614.2837640.

24 Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. Separating regular languages by piecewise testable and unambiguous languages. In *MFCS 2013*, volume 8087 of *Lecture Notes in Computer Science*, pages 729–740. Springer, 2013. doi:10.1007/978-3-642-40313-2_64.

25 Thomas Place and Marc Zeitoun. Automata column: The tale of the quantifier alternation hierarchy of first-order logic over words. *SIGLOG News*, 2(3):4–17, 2015. doi:10.1145/2815493.2815495.

26 Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer, 1975. doi:10.1007/3-540-07407-4_23.

27 Thomas G. Szymanski and John H. Williams. Noncanonical extensions of bottom-up parsing techniques. *SIAM Journal on Computing*, 5(2):231–250, 1976. doi:10.1137/0205019.

28 W. W. Tait. A counterexample to a conjecture of Scott and Suppes. *Journal of Symbolic Logic*, 24(1):15–16, 1959. doi:10.2307/2964569.

29 Jan van Leeuwen. Effective constructions in well-partially-ordered free monoids. *Discrete Mathematics*, 21(3):237–252, 1978. doi:10.1016/0012-365X(78)90156-5.

30 Georg Zetzsche. An approach to computing downward closures. In *ICALP 2015*, volume 9135 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2015. doi:10.1007/978-3-662-47666-6_35.

Computation Tree Logic for Synchronization Properties*†

Krishnendu Chatterjee[1] and Laurent Doyen[2]

1 **IST Austria, Wien, Austria**
2 **LSV, ENS Cachan & CNRS, Paris, France**

Abstract

We present a logic that extends CTL (Computation Tree Logic) with operators that express synchronization properties. A property is synchronized in a system if it holds in all paths of a certain length. The new logic is obtained by using the same path quantifiers and temporal operators as in CTL, but allowing a different order of the quantifiers. This small syntactic variation induces a logic that can express non-regular properties for which known extensions of MSO with equality of path length are undecidable. We show that our variant of CTL is decidable and that the model-checking problem is in $\Delta_3^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$, and is hard for the class of problems solvable in polynomial time using a parallel access to an NP oracle. We analogously consider quantifier exchange in extensions of CTL, and we present operators defined using basic operators of CTL* that express the occurrence of infinitely many synchronization points. We show that the model-checking problem remains in Δ_3^{P}. The distinguishing power of CTL and of our new logic coincide if the Next operator is allowed in the logics, thus the classical bisimulation quotient can be used for state-space reduction before model checking.

1998 ACM Subject Classification F.4.1 Temporal Logic

Keywords and phrases Computation Tree Logic, Synchronization, model-checking, complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.98

1 Introduction

In computer science, it is natural to view computations as a tree, where each branch represents an execution trace, and all possible execution traces are arranged in a tree. To reason about computations, the logical frameworks that express properties of trees have been widely studied [10, 20, 24], such as CTL, CTL*, μ-calculus, MSO, etc. These logics can express ω-regular properties about trees.

A key advantage of logics is to provide concise and formal semantics, and a rigorous language to express properties of a system. For example, the logic CTL is widely used in verification tools such as NuSMV [9], and hyperproperties, i.e. tree-based properties that cannot be defined over individual traces, are relevant in security [11, 12].

One key property that has been studied in different contexts is the property of synchronization, which intuitively requires that no matter how the system behaves it synchronizes to a common good point. Note that the synchronization property is inherently a tree-based property, and is not relevant for traces. Synchronization has been studied for automata [25, 8],

* Full version [6].

† This research was partially supported by Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE), ERC Start grant (279307: Graph Games), Vienna Science and Technology Fund (WWTF) through project ICT15-003, and European project Cassting (FP7-601148).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 98; pp. 98:1–98:14

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

probabilistic models such as Markov decision processes [15, 16], as well as partial-information, weighted, and timed models [19, 17, 14], and has a rich collection of results as well as open problems, e.g., Černý's conjecture about the length of synchronizing words in automata is one of the long-standing and well-studied problems in automata theory [5, 25]. A natural question is how can synchronization be expressed in a logical framework.

First, we show that synchronization is a property that is not ω-regular. Hence it cannot be expressed in existing tree-based logics, such as MSO, CTL*, etc. A natural candidate to express synchronization in a logical framework is to consider MSO with quantification over path length. Unfortunately the quantification over path length in MSO leads to a logic for which the model-checking problem is undecidable [23, Theorem 11.6]. Thus an interesting question is how to express synchronization in a logical framework where the model-checking problem is decidable.

Contributions. In this work we introduce an elegant logic, obtained by a natural variation of CTL. The logic allows to exchange the temporal and path quantifiers in classical CTL formulas. For example, consider the CTL formula $\forall Fq$ expressing the property that in all paths there exists a position where q holds (quantification pattern $\forall\text{paths} \cdot \exists\text{position}$). In our logic, the formula $F\forall q$ with quantifiers exchanged expresses that there exists a position k such that for all paths, q holds at position k (quantification pattern $\exists\text{position} \cdot \forall\text{paths}$), see Figure 1a. Thus q eventually holds in all paths at the same position, expressing that the paths are eventually synchronized.

We show that the model-checking problem is decidable for our logic, which we show is in $\Delta_3^{\mathrm{P}} = \mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ (in the third level of the polynomial hierarchy) and is hard for the class $\mathrm{P}_{\parallel}^{\mathrm{NP}}$ of problems solvable in polynomial time using a parallel access to an NP oracle (Theorem 1). The problems in $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ can be solved by a polynomial-time algorithm that uses an oracle for a problem in $\mathrm{NP}^{\mathrm{NP}}$, and the problems in $\mathrm{NP}^{\mathrm{NP}}$ can be solved by a non-deterministic polynomial-time algorithm that uses an oracle for an NP-complete problem; the problems in $\mathrm{P}_{\parallel}^{\mathrm{NP}}$ can be solved by a polynomial-time algorithm that works in two phases, where in the first phase a list of queries is constructed, and in the second phase the queries are answered by an NP oracle (giving a list of yes/no answers) and the algorithm proceeds without further calling the oracle [26, 21].

We present an extension of our logic that can express the occurrence of infinitely many synchronization points (instead of one as in eventually sychronizing), and the absence of synchronization from some point on, with the same complexity status (Section 3). These properties are the analogue of the classical liveness and co-liveness properties in the setting of synchronization. We show that such properties cannot be expressed in the basic logic (Section 4). In Section 6, we consider the possibility to further extend our logic with synchronization to CTL*, and show that the exchange of quantifiers in CTL* formulas would lead to either a counter-intuitive semantics, or an artificial logic that would be inelegant.

We study the distinguishing power of the logics in Section 5, that is the ability of the logics, given two models, to provide a formula that holds in one model, and not in the other. The distinguishing power is different from the expressive power of a logic, as two logics with the same expressive power have the same distinguishing power but not vice versa. The distinguishing power can be used for state-space reduction before running a model-checking algorithm, in order to obtain a smaller equivalent model, that the logic cannot distinguish from the original model, and thus for which the answer of the model-checking algorithm is the same. We show that if the Next operator is allowed in the logic, then the distinguishing power coincides with that of CTL (two models are indistinguishable if and only if they

are bisimilar), and if the Next operator is not allowed, then the distinguishing power lies between bisimulation and stuttering bisimulation, and is NP-hard to decide. In particular, it follows that with or without the Next operator the state-space reduction with respect to bisimulation, which is computable in polynomial time, is sound for model-checking. Detailed proofs are available in [6].

2 CTL + Synchronization

We introduce the logic CTL+Sync after presenting basic definitions related to Kripke structures. A *Kripke structure* is a tuple $K = \langle T, \Pi, \pi, R\rangle$ where T is a finite set of states, Π is a finite set of atomic propositions, $\pi : T \to 2^{\Pi}$ is a labeling function that maps each state t to the set $\pi(t)$ of propositions that are true at t, and $R \subseteq T \times T$ is a transition relation. We denote by $R(t) = \{t' \mid (t, t') \in R\}$ the set of successors of a state t according to R, and given a set $s \subseteq T$ of states, let $R(s) = \bigcup_{t\in s} R(t)$. A Kripke structure is *deterministic* if $R(t)$ is a singleton for all states $t \in T$. A *path* in K is an infinite sequence $\rho = t_0 t_1 \dots$ such that $(t_i, t_{i+1}) \in R$ for all $i \geq 0$. For $n \in \mathbb{N}$, we denote by $\rho + n$ the suffix $t_n t_{n+1} \dots$.

2.1 Syntax and semantics

In the CTL operators, a path quantifier always precedes the temporal quantifiers (e.g., $\exists\mathcal{U}$ or $\forall\mathcal{U}$). We obtain the logic CTL+Sync from traditional CTL by allowing to switch the order of the temporal and path quantifiers. For example, the CTL formula $p\,\forall\mathcal{U}\,q$ holds in a state t if for all paths ($\forall$) from t, there is a position where q holds, and such that p holds in all positions before ($\mathcal{U}$). In the CTL+Sync formula $p\mathcal{U}\forall\, q$, the quantifiers are exchanged, and the formula holds in t if there exists a position k, such that for all positions $j < k$ before ($\mathcal{U}$), in all paths ($\forall$) from t, we have that q holds at position k and p holds at position j (see Figure 1d). Thus the formula $p\mathcal{U}\forall\, q$ requires that q holds synchronously after the same number of steps in all paths, while the formula $p\,\forall\mathcal{U}\,q$ does not require such synchronicity across several paths.

The syntax of the formulas in CTL+Sync is as follows:

$$\varphi ::= p \mid \neg\varphi_1 \mid \varphi_1 \vee \varphi_2 \mid Q\mathcal{X}\,\varphi_1 \mid \varphi_1\, Q\mathcal{U}\,\varphi_2 \mid \varphi_1\,\mathcal{U}Q\,\varphi_2$$

where $p \in \Pi$ and $Q \in \{\exists, \forall\}$. We define true and additional Boolean connectives as usual, and let

- $\exists F\varphi \equiv \mathsf{true}\,\exists\mathcal{U}\,\varphi$, and $F\exists\varphi \equiv \mathsf{true}\mathcal{U}\exists\,\varphi$, etc.
- $\exists G\varphi \equiv \neg\forall F\neg\varphi$, etc.

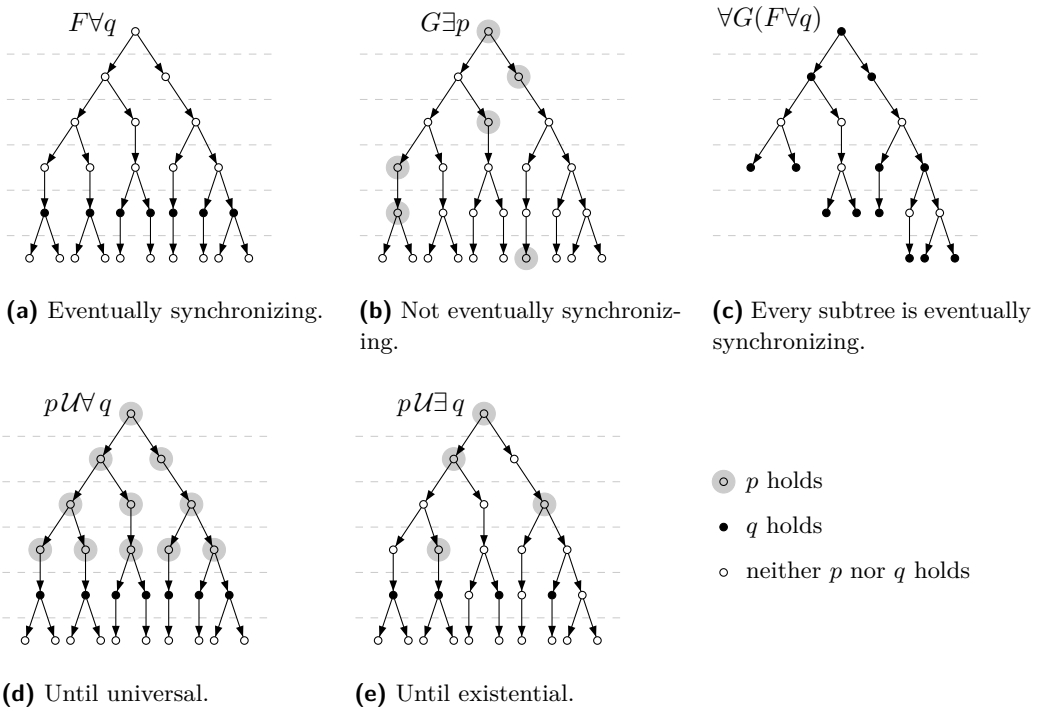
Note that the Next operators $Q\mathcal{X}$ has only one quantifier, and thus there is no point in switching quantifiers or defining an operator $\mathcal{X}Q$.

Given a Kripke structure $K = \langle T, \Pi, \pi, R\rangle$, and a state $t \in T$, we define the satisfaction relation $\models$ as follows. The first cases are standard and exist already in CTL:

- $K, t \models p$ if $p \in \pi(t)$.
- $K, t \models \neg\varphi_1$ if $K, t \not\models \varphi_1$.
- $K, t \models \varphi_1 \vee \varphi_2$ if $K, t \models \varphi_1$ or $K, t \models \varphi_2$.
- $K, t \models \exists\mathcal{X}\,\varphi_1$ if $K, t' \models \varphi_1$ for some $t' \in R(t)$.
- $K, t \models \forall\mathcal{X}\,\varphi_1$ if $K, t' \models \varphi_1$ for all $t' \in R(t)$.

The interesting new cases are built using the until operator of CTL:

- $K, t \models \varphi_1\,\exists\mathcal{U}\,\varphi_2$ if there exists a path $t_0 t_1 \dots$ in K with $t_0 = t$ and there exists $k \geq 0$ such that: $K, t_k \models \varphi_2$, and $K, t_j \models \varphi_1$ for all $0 \leq j < k$.



(a) Eventually synchronizing.

(b) Not eventually synchronizing.

(c) Every subtree is eventually synchronizing.

(d) Until universal.

(e) Until existential.

Figure 1 Formulas of CTL+Sync.

- $K, t \models \varphi_1 \mathcal{U}\exists\, \varphi_2$ if there exists $k \geq 0$ such that for all $0 \leq j < k$, there exists a path $t_0 t_1 \dots$ in K with $t_0 = t$ such that $K, t_j \models \varphi_1$ and $K, t_k \models \varphi_2$.
- $K, t \models \varphi_1 \forall\mathcal{U}\, \varphi_2$ if for all paths $t_0 t_1 \dots$ in K with $t_0 = t$, there exists $k \geq 0$ such that: $K, t_k \models \varphi_2$, and $K, t_j \models \varphi_1$ for all $0 \leq j < k$.
- $K, t \models \varphi_1 \mathcal{U}\forall\, \varphi_2$ if there exists $k \geq 0$ such that for all $0 \leq j < k$ and for all paths $t_0 t_1 \dots$ in K with $t_0 = t$, we have $K, t_j \models \varphi_1$ and $K, t_k \models \varphi_2$.

We often write $t \models \varphi$ when the Kripke structure K is clear from the context.

Examples of formulas are given in Figure 1. The examples show the first steps of the unravelling of Kripke structures defined over atomic propositions $\{p, q\}$. The formula $F\forall q$ expresses that q eventually holds synchronously on all paths, after the same number of steps (Figure 1a). This is different from the CTL formula $\forall F q$, which expresses that all paths eventually visit a state where q holds, but not necessarily after the same number of steps in all paths. The dual formula $G\exists p$ requires that at every depth (i.e., for all positions k), there exists a path where p holds at depth k (Figure 1b). On the other hand note that $F\exists q \equiv \exists F q$ and dually $G\forall p \equiv \forall G p$. Another example is the formula $\forall G(F\forall q)$ expressing that every subtree is eventually synchronizing (Figure 1c). The until universal formula $p\mathcal{U}\forall\, q$ holds if q holds at a certain position in every path (like for the formula $F\forall q$), and p holds in all positions before (Figure 1d). The until existential formula $p\mathcal{U}\exists\, q$ says that it is possible to find path(s) where q holds at the same position, and such that for all smaller positions there is one of those paths where p holds at that position (Figure 1e).

▶ Remark. The definition of CTL+Sync, although very similar to the definition of CTL, interestingly allows to define non-regular properties, thus not expressible in CTL (or even in MSO over trees). It is easy to show using a pumping argument that the property

$F\forall q$ of eventually sychronizing is not regular (Figure 1a). This property of eventually sychronizing can be expressed in MSO extended with a length predicate, by a formula such as $\exists \rho \in T^* \cdot \forall \rho' \in T^* : |\rho| = |\rho'| \implies q(\rho')$ where $T = \{0,1\}$ and $q(\cdot)$ is a monadic predicate for the proposition q over the binary tree T^*, where $q(\rho)$ means that q holds in the last state of ρ. However, model-checking for the logic MSO extended with the "equal-length" predicate p defined by $p(\rho, \rho') \equiv |\rho| = |\rho'|$ is undecidable [23, Theorem 11.6]. In contrast, we show in Theorem 1 that the logic CTL+Sync is decidable.

2.2 Model-checking

Given a CTL+Sync formula φ, a Kripke structure K, and a state t, the *model-checking problem for CTL+Sync* is to decide whether $K, t \models \varphi$ holds.

Model-checking of CTL+Sync can be decided by considering a powerset construction for the Kripke structure, and evaluating a CTL formula on it. For example, to evaluate a formula $\varphi_1 \mathcal{U} \forall \varphi_2$ from state t_I in a Kripke structure K, it suffices to consider the sequence $s_1 s_2 \dots$ defined by $s_1 = \{t_I\}$ and $s_{i+1} = R(s_i)$ for all $i \geq 1$, where a set s is labeled by φ_1 if $K, t \models \varphi_1$ for all $t \in s$ (and analogously for φ_2). The formula $\varphi_1 \mathcal{U} \forall \varphi_2$ holds in t_I if and only if the formula $\varphi_1 \mathcal{U} \varphi_2$ holds in the sequence $s_1 s_2 \dots$ (note that on a single sequence the operators $\forall\mathcal{U}$ and $\exists\mathcal{U}$ are equivalent, thus we simply write $\mathcal{U}$).

For the formula $\varphi_1 \mathcal{U} \exists \varphi_2$, intuitively it holds in t_I if there exists a set P of finite paths $\rho_1, \rho_2, \dots, \rho_n$ from t_I in K, all of the same length k, such that φ_2 holds in the last state of ρ_i for all $1 \leq i \leq n$, and for every $1 \leq j < k$ there is a path ρ_{i_j} such that φ_1 holds in the jth state of ρ_{i_j}. To evaluate $\varphi_1 \mathcal{U} \exists \varphi_2$ from t_I, we construct the Kripke structure $2^K = \langle 2^T, \{\varphi_1, \varphi_2\}, \pi, \hat{R} \rangle$ where $(s, s') \in \hat{R}$ if for all $t \in s$ there exists $t' \in s'$ such that $(t, t') \in R$, thus we have to choose (nondeterministically) at least one successor from each state in s, that is for every set P of paths $\rho_1, \rho_2, \dots, \rho_n$ as above, there is a path $s_1, s_2, \dots, s_k$ (with $s_1 = \{t_I\}$) in 2^K where the sets s_i are obtained by following simultaneously the finite paths $\rho_1, \dots, \rho_n$, thus such that s_i is the set of states at position i of the paths in P. The path $s_1, s_2, \dots, s_k$ in 2^K corresponds to a set P of finite paths in K that show that $\varphi_1 \mathcal{U} \exists \varphi_2$ holds if (1) φ_2 holds in all states of s_k, and (2) φ_1 holds in some state of s_i $(i = 1, \dots, k-1)$. Hence we define the labeling function π in 2^K as follows: for all $s \in 2^T$ let $\varphi_2 \in \pi(s)$ if $K, t \models \varphi_2$ for all $t \in s$, and let $\varphi_1 \in \pi(s)$ if $K, t \models \varphi_1$ for some $t \in s$. Finally it suffices to check whether the CTL formula $\varphi_1 \exists \mathcal{U} \varphi_2$ holds in 2^K from $\{t_I\}$.

This approach gives an exponential algorithm, and even a PSPACE algorithm by exploring the powerset construction on the fly. However, we show that the complexity of the model-checking problem is much below PSPACE. For example our model-checking algorithm for the formula $F\forall q$ relies on guessing a position $k \in \mathbb{N}$ (in binary) and checking that q holds on all paths at position k. To compute the states reachable after exactly k steps, we compute the kth power of the transition matrix $M \in \{0,1\}^{T \times T}$ where $M(t, t') = 1$ if there is a transition from state t to state t'. The power M^k can be computed in polynomial time by successive squaring of M. For this formula, we obtain an NP algorithm. For the whole logic, combining the guessing and squaring technique with a dynamic programming algorithm that evaluates all subformlas, we obtain an algorithm in $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ for the model-checking problem [6]. We present a hardness result for the class $\mathrm{P}_{\parallel}^{\mathrm{NP}}$ of problems solvable in polynomial time using a parallel access to an NP oracle [26, 21].

▶ **Theorem 1.** *The model-checking problem for CTL+Sync lies in $P^{NP^{NP}}$ and is $P_{\parallel}^{NP}$-hard.*

The complexity lower bounds for the model-checking problem in Theorem 1 are based on Lemma 2 where we establish complexity bounds for fixed formulas.

▶ **Lemma 2.** *Let $p, q \in \Pi$ be two atomic propositions. The model-checking problem is:*
- *NP-complete for the formulas $p\mathcal{U}\forall\, q$ and $F\forall q$,*
- *DP-hard for the formula $p\mathcal{U}\exists\, q$, and*
- *coNP-complete for the formula $G\exists q$.*

Proof. We prove the hardness results (complexity lower bounds), since the complexity upper bounds follow from the proof of Theorem 1.

The proof technique is analogous to the NP-hardness proof of [22, Theorem 6.1], and based on the following. Given a Boolean propositional formula ψ over variables $x_1, \ldots, x_n$, consider the first n prime numbers $p_1, \ldots, p_n$. For a number $z \in \mathbb{N}$, if $z \mod p_i \in \{0, 1\}$ for all $1 \leq i \leq n$, then the binary vector $(z \mod p_1, \ldots, z \mod p_n)$ defines an assignment to the variables of the formula. Note that conversely, every such binary vector can be defined by some number $z \in \mathbb{N}$ (by the Chinese remainder theorem).

NP-hardness of $F\forall q$ (and thus of $p\mathcal{U}\forall\, q$). The proof is by a reduction from the Boolean satisfiability problem 3SAT which is NP-complete [13]. Given a Boolean propositional formula ψ in CNF, with set C of (disjunctive) clauses over variables $x_1, \ldots, x_n$ (where each clause contains three variables), we construct a Kripke structure K_ψ as follows: for each clause $c \in C$, we construct a cycle $t_0, t_1, \ldots, t_{r-1}$ of length $r = p_u \cdot p_v \cdot p_w$ where the three variables in the clause are x_u, x_v, and x_w. We call t_0 the origin of the cycle, and we assign to every state t_i the label q if the number i defines an assignment that satisfies the clause c. The Kripke structure K_ψ is the disjoint union of the cycles corresponding to each clause, and an initial state t_I with transitions from t_I to the origin of each cycle. Note that the Kripke structure K_ψ can be constructed in polynomial time, as the sum of the first n prime numbers is bounded by a polynomial in n: $\sum_{i=1}^{n} p_i \in O(n^2 \log n)$ [2].

It follows that a number z defines an assignment that satisfies the formula ψ (i.e., satisfies all clauses of ψ) if and only if every path of length $z+1$ from t_I reaches a state labelled by q. Therefore the formula ψ is satisfiable if and only if $K_\psi, t_I \models F\forall q$, and it follows that the model-checking problem is NP-hard for the formulas $F\forall q$ and for $p\mathcal{U}\forall\, q$ (let p hold in every state of K_ψ).

NP-hardness of $p\mathcal{U}\exists\, q$. The proof is by a reduction from 3SAT [13]. The reduction is illustrated in Figure 2. Given a Boolean propositional formula ψ in CNF, with set C of (disjunctive) clauses over variables $x_1, \ldots, x_n$ (where each clause contains three variables), we construct a Kripke structure K as follows: let $m = |C|$ be the number of clauses in ψ, and construct m disjoint simple paths π_i from t_I of length $m+1$ (of the form $t_I, t_1, \ldots, t_m$), where the last state of each path π_i has a transition to the origin of a cycle corresponding to the ith clause (the cycles and their labeling are as defined in the NP-hardness proof of $F\forall q$). The state t_I and all states of the cycles are also labelled by p, and in the ith path from t_I, the ith state after t_I is labelled by p. The construction can be obtained in polynomial time.

We show that ψ is satisfiable if and only if the formula $p\mathcal{U}\exists\, q$ holds from t_I in K. Recall that $p\mathcal{U}\exists\, q$ holds if there exists $k \geq 0$ such that for all $0 \leq j < k$, there exists a path $t_0 t_1 \ldots$ in K with $t_0 = t_I$ and $K, t_j \models p$ and $K, t_k \models q$.

For the first direction of the proof, if ψ is satisfiable, then let $z \in \mathbb{N}$ define a satisfying assignment, and let $k = m + 2 + z$. Then all paths of length k from t_I in K end up in a state labelled by q. Now we consider an arbitrary $j < k$ and show that there exists a path of length k from t_I that ends up in a state labelled by q, and with the jth state labelled by p. For $j = 0$ and for $j > m$, the conditions are satisfied by all paths, and for $j \leq m$, the conditions are satisfied by the jth path from t_I.

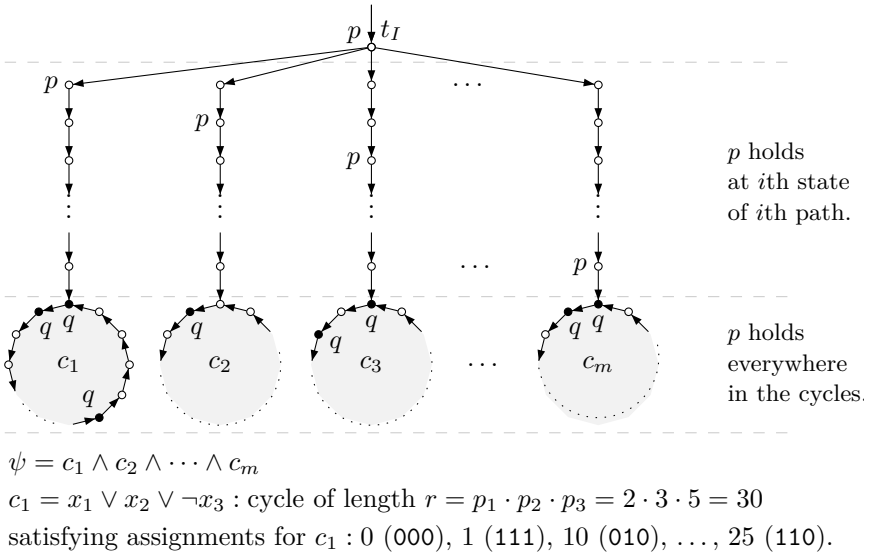


Figure 2 Reduction to show NP-hardness of $p\mathcal{U}\exists\, q$ in Lemma 2.

For the second direction of the proof, let k be a position such that for all $0 \leq j < k$, there exists a path $t_0 t_1 \dots$ in K with $t_0 = t_I$ and $K, t_j \models p$ and $K, t_k \models q$. Then $k \geq m+2$ since only the states in the cycles are labelled by q. Consider the set P containing, for each $j = 1, 2, \dots, m$, a path $t_I t_1 \dots$ in K with $K, t_j \models p$ and $K, t_k \models q$. It is easy to see by the construction of K that P contains all the paths of length k in K. Therefore, all paths of length $z = k - (m+2)$ from the origin of each cycle end up in a state labelled by q. It follows that z defines an assignment that satisfies all clauses in ψ, thus ψ is satisfiable.

DP-hardness of $p\mathcal{U}\exists\, q$. The DP-hardness proof of $p\mathcal{U}\exists\, q$ uses a reduction of the same flavor as in the NP-hardness of $F\forall q$ [6].

coNP-hardness of $G\exists q$. The result follows from the NP-hardness of $F\forall q$ since $G\exists q$ is equivalent to $\neg F\forall \neg q$. ◀

The complexity result of Theorem 1 is not tight, with a $\mathrm{P}^{\mathrm{NP}^{\mathrm{NP}}}$ upper bound and hardness for $\mathrm{P}_{\|}^{\mathrm{NP}}$. Even for the fixed formula $p\mathcal{U}\exists\, q$, the gap between the $\mathrm{NP}^{\mathrm{NP}}$ upper bound and the DP-hardness result provides an interesting open question for future work.

3 Extension of CTL+Sync with Always and Eventually

We consider an extension of CTL+Sync with formulas of the form $\mathcal{T}Q\varphi$ where $\mathcal{T} \in \{F, G\}^+$ is a sequence of unary temporal operators Eventually (F) and Always (G). For example, the formula $FG\forall p$ expresses strong synchronization, namely that from some point on, all positions on every path satisfy p; the formula $GF\forall p$ expresses weak synchronization, namely that there are infinitely many positions such that, on every path at those positions p holds. In fact only the combination of operators FG and GF need to be considered, as the other combinations of operators reduce to either FG or GF using the LTL identities $FGF\varphi \equiv GF\varphi$ and $GFG\varphi \equiv FG\varphi$. Formally, define:

- $K, t \models GF\forall\varphi_1$ if for all $k \geq 0$, there exists $j \geq k$ such that for all paths $t_0 t_1 \dots$ in K with $t_0 = t$, we have $K, t_j \models \varphi_1$.
- $K, t \models GF\exists\varphi_1$ if for all $k \geq 0$, there exists $j \geq k$ and there exists a path $t_0 t_1 \dots$ in K with $t_0 = t$ such that $K, t_j \models \varphi_1$.
- $K, t \models FG\forall\varphi_1$ if $K, t \not\models GF\exists\neg\varphi_1$.
- $K, t \models FG\exists\varphi_1$ if $K, t \not\models GF\forall\neg\varphi_1$.

The model-checking problem for the formula $GF\forall\varphi_1$ is NP-complete: guess positions $n, k \leq 2^{|T|}$ (represented in binary) and check in polynomial time that the states reachable by all paths of length n satisfy φ_1, and that set of the states reachable after $n + k$ steps is the same as the set of states reachable after n steps, where $k > 0$. This corresponds to finding a lasso in the subset construction for the Kripke structure K. A matching NP lower bound follows from the reduction in the NP-hardness proof of $F\forall q$ (Lemma 2).

The model-checking problem for the formula $GF\exists\varphi_1$ can be solved in polynomial time, as this formula is equivalent to saying that there exists a state labeled by φ_1 that is reachable from a reachable non-trivial strongly connected component (SCC) — an SCC is trivial if it consists of a single state without self-loop. To prove this, note that if a state t^* labeled by φ_1 is reachable from a reachable non-trivial SCC, then t^* can be reached by an arbitrarily long path, thus the formula $GF\exists\varphi_1$ holds. For the other direction, if no state labeled by φ_1 is reachable from a reachable non-trivial SCC, then every path to a state labeled by φ_1 is acyclic (otherwise, the path would contain a cycle, belonging to an SCC). Since acyclic paths have length at most $|T|$, it follows that the formula $GF\exists\varphi_1$ does not hold, which concludes the proof.

From the above arguments, it follows that the complexity status of the model-checking problem for this extension of CTL+Sync is the same as the complexity of CTL+Sync model-checking in Theorem 1.

▶ **Theorem 3.** *The model-checking problem for CTL+Sync extended with sequences of unary temporal operators lies in $P^{NP^{NP}}$ and is $P_{\|}^{NP}$-hard.*

4 Expressive Power

The expressive power of CTL+Sync (even extended with Always and Eventually) is incomparable with the expressive power of MSO. By the remark at the end of Section 2.1, CTL+Sync can express non-regular properties, and thus is not subsumed by MSO, and standard argument based on counting properties [27] showing that CTL is less expressive than MSO apply straightforwardly to show that formulas of MSO are not expressible in CTL+Sync [10].

We show that the formulas $GF\forall p$ and $FG\forall p$ for weak and strong synchronization cannot be expressed in the logic CTL+Sync, thus CTL+Sync extended with sequences of unary temporal operators is strictly more expressive than CTL+Sync. The result holds if the Next operator is not allowed, and also if the Next operator is allowed.

▶ **Theorem 4.** *The logic CTL+Sync (even without the Next operator) extended with sequences of unary temporal operators is strictly more expressive than CTL+Sync (even using the Next operator).*

Proof. We show that the formula $GF\forall p$ cannot be expressed in CTL+Sync, even using the Next operator. To prove this, given an arbitrary CTL+Sync formula φ, we construct two

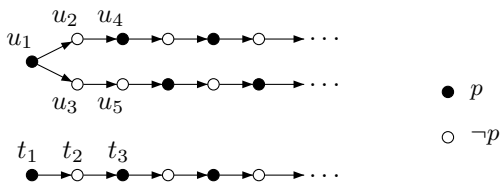


Figure 3 States t_1 and u_1 are indistinguishable by formulas of CTL+Sync.

Kripke structures such that φ holds in both Kripke structures, but the formula $GF\forall p$ holds in one and not in the other. It follows that φ is not equivalent to $GF\forall p$.

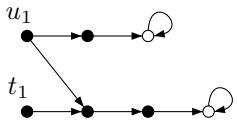
Given the formula φ, we construct the two Kripke structures as follows. Consider two Kripke structures whose unravelling is shown in Figure 3 where the states reachable from t_1 are satisfying alternately p and $\neg p$, and the states reachable from u_2 and u_5 are satisfying alternately $\neg p$ and p. Call black states the states where p holds, and white states the states where $\neg p$ holds. If n is the maximum number of nested Next operators in φ, then we construct the n-stuttering of the two Kripke structures in Figure 3, where the n-stuttering of a Kripke structure $K = \langle T, \Pi, \pi, R\rangle$ is the Kripke structure $K^n = \langle T \times \{1, \ldots, n\}, \Pi, \pi^n, R^n\rangle$ where $\pi^n(t, i) = \pi^n(t)$ for all $1 \leq i \leq n$, and the transition relation R^n contains all pairs $((t, i), (t, i + 1))$ for all $t \in T$ and $1 \leq i < n$, and all pairs $((t, n), (t', 1))$ for all $(t, t') \in R$.

We claim that the formula φ holds either in both $(t_1, 1)$ and $(u_1, 1)$, or in none of $(t_1, 1)$ and $(u_1, 1)$, while the formula $GF\forall p$ holds in $(t_1, 1)$ and not in $(u_1, 1)$. We show by induction on the nesting depth of CTL+Sync formulas φ (that have at most n nested Next operators) that (t_1, i) and (u_1, i) are equivalent for φ (for all $1 \leq i \leq n$), and that for all black states t, u, the copies $(t, 1)$ and $(u, 1)$ are equivalent for φ, and analogously for all pairs of white states.

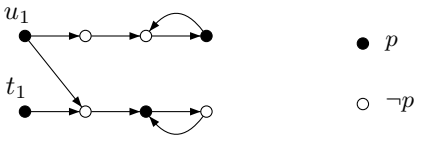
The result holds trivially for formulas of nesting depth 0, that is atomic propositions. For the induction step, assume the claim holds for formulas of nesting depth k, and consider a formula φ of nesting depth $k + 1$. If the outermost operator of φ is a Boolean operator, or a CTL operator ($Q\mathcal{X}$ or $Q\mathcal{U}$), then the result follows from the induction hypothesis and the result of [18, Theorem 2] showing two paths that differ only in the number of consecutive repetitions of a state, as long as the number of repetitions is at least n, are equivalent for the formulas with at most n nested Next operators. If the outermost operator of φ is either $\mathcal{U}\exists$ or $\mathcal{U}\forall$, that is $\varphi \equiv \varphi_1 \mathcal{U}\exists \varphi_2$ or $\varphi \equiv \varphi_1 \mathcal{U}\forall \varphi_2$, then consider a state where φ holds: either φ_2 holds in that state, and by the induction hypothesis, φ_2 also holds in the corresponding state (that we claimed to be equivalent), or φ_2 holds in the states of the other color than the current state, and φ_1 holds on the path(s) at all positions before. By the induction hypothesis, at the same distance from the claimed equivalent states, we can find a state where φ_2 holds in all paths, and φ_1 holds on all positions before, which concludes the proof for the induction step. ◀

5 Distinguishing Power

Two states of a Kripke structure can be distinguished by a logic if there exists a formula in the logic that holds in one state but not in the other. Each logic induces an indistinguishability relation (which is an equivalence) on Kripke structures that characterizes the distinguishing power of the logic. Two states t and t' of a Kripke structure K are indistinguishable by a logic $\mathcal{L}$ if they satisfy the same formulas of $\mathcal{L}$, that is $\{\varphi \in \mathcal{L} \mid K, t \models \varphi\} = \{\varphi \in \mathcal{L} \mid K, t' \models \varphi\}$.



(a) The states t_1 and u_1 are stuttering bisimilar (they satisfy the same CTL formulas without the Next operator), but they can be distinguished by the CTL+Sync formula $p\mathcal{U}\forall\,\neg p$ which holds in t_1 but not in u_1.



(b) The states t_1 and u_1 are indistinguishable by CTL+Sync formulas, but they are not bisimilar, i.e. they can be distinguished by CTL formulas with the Next operator, for example $\forall\mathcal{X}\,\forall\mathcal{X}\,p$ which holds in t_1 but not in u_1.

Figure 4 The distinguishing power of CTL+Sync lies strictly between bisimulation and stuttering bisimulation.

For CTL (with the Next operator), the distinguishing power is standard bisimulation, and for CTL without the Next operator, the distinguishing power is stuttering bisimulation [4]. Stuttering bisimulation is a variant of bisimulation where intuitively several transitions can be used to simulate a single transition, as long as the intermediate states of the transitions are all equivalent (for stuttering bisimulation). We omit the definition of bisimulation and stuttering bisimulation [4], and in this paper we consider that they are defined as the distinguishing power of respectively CTL and CTL without the Next operator.

It is easy to show by induction on the nesting depth of formulas that the distinguishing power of CTL+Sync is the same as for CTL, since (i) CTL+Sync contains CTL, and (ii) if two states t and t' are bisimilar, there is a correspondence between the paths starting from t and the paths starting from t' (for every path from t, there is a path from t' such that their states at position i are bisimilar, for all $i \in \mathbb{N}$, and analogously for every path of t' [4, Lemma 3.1]), which implies the satisfaction of the same formulas in CTL+Sync. The same argument holds for CTL+Sync extended with unary temporal operators (Section 3).

▶ **Theorem 5.** *Two states t and t' of a Kripke structure K are indistinguishable by CTL+Sync formulas (even extended with unary temporal operators) if and only if t and t' are bisimilar.*

Without the Next operator, the logic CTL+Sync has a distinguishing power that lies strictly between bisimulation and stuttering bisimulation, as shown by the examples in Figure 4a and Figure 4b. Indistinguishability by CTL+Sync formulas without the Next operator implies indistinguishability by standard CTL without the Next operator, and thus stuttering bisimilarity. We obtain the following result.

▶ **Theorem 6.** *The following implications hold for all states t, t' of a Kripke structure K:*

- *if t and t' are bisimilar, then t and t' are indistinguishable by CTL+Sync formulas without the Next operator (even extended with unary temporal operators);*
- *if t and t' are indistinguishable by CTL+Sync formulas without the Next operator (even extended with unary temporal operators), then t and t' are stuttering bisimilar.*

It follows from the first part of Theorem 6 that the state-space reduction techniques based on computing a bisimulation quotient before evaluating a CTL formula will work for CTL+Sync. Although the exact indistinguishability relation for CTL+Sync is coarser than bisimulation, we show that deciding this relation is NP-hard, and thus it may not be relevant to compute it for quotienting before model-checking, but rather use the polynomial-time computable bisimulation.

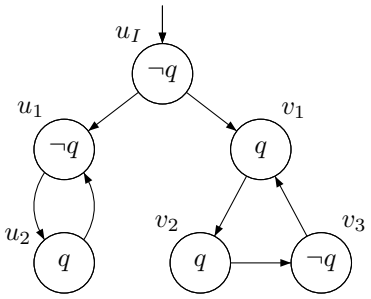


Figure 5 The Kripke structure K in the proof of Theorem 7.

▶ **Theorem 7.** *Deciding whether two states of a Kripke structure are indistinguishable by CTL+Sync formulas without the Next operator is NP-hard.*

Proof. The proof is by a reduction from the Boolean satisfiability problem 3SAT which is NP-complete [13]. Given a Boolean propositional formula ψ in CNF, we construct two Kripke structures K and K_ψ that are indistinguishable (from their initial state) if and only if ψ is satisfiable, where:

- K is the Kripke structure shown in Figure 5, and
- K_ψ is the Kripke structure constructed in the NP-hardness proof of $F\forall q$ (Lemma 2).

We know from the proof of Lemma 2 that $K_\psi, t_I \models F\forall q$ if and only if ψ is satisfiable. Hence, it suffices to show that K and K_ψ are indistinguishable if and only if the formula $F\forall q$ holds in t_I. Since the formula $F\forall q$ holds in u_I, we only need to show that if $F\forall q$ holds in t_I, then K and K_ψ are indistinguishable. To do this, we assume that $F\forall q$ holds in t_I, and we show that for all CTL+Sync formulas φ without the Next operator, $t_I \models \varphi$ if and only if $u_I \models \varphi$. The proof proceeds by induction on the nesting depth of φ and simple combinatorial arguments [6]. ◀

6 CTL* + Synchronization

CTL* is a branching-time extension of LTL (and of CTL) where several nested temporal operators and Boolean connectives can be used under the scope of a single path quantifier. For example the CTL* formula $\exists(G\varphi \to G\psi)$ says that there exists a path in which either φ does not hold in every position, or ψ holds at every position. Note that φ and ψ may also contain path quantifiers.

Extending CTL+Sync with formula quantification analogous to CTL* presents some difficulties. Even considering only Boolean connectives and $\{F, G\}$ operators leads to a logic that is hard to define. For example, one may consider a formula like $(Fp \vee Fq)\forall$ which could be naturally interpreted as there exist two positions $m, n \geq 0$ such that on all paths ρ, either p holds at position m in ρ, or q holds at position n in ρ. In this definition the $\vee$ operator would not be idempotent, that is $\psi_1 = (Fp \vee Fp)\forall$ is not equivalent to $\psi_2 = (Fp)\forall$, where ψ_1 means that the set of all paths can be partitioned into two sets of paths where p holds synchronously at some position, but not necessarily the same position in both sets, while ψ_2 expresses the property that p holds synchronously at some position in all paths.

Another difficulty with binary operators is the semantics induced by the order of the operands. For instance, the formula $(Fp \vee Gq)\forall$ can be interpreted as (i) there exists a position $m \geq 0$ such that for all positions $n \geq 0$, on all paths ρ, either $\rho + m \models p$ or $\rho + n \models q$;

or it can be interpreted as (ii) for all $n \geq 0$, there exists $m \geq 0$ such that on all paths ρ, either $\rho + m \models p$ or $\rho + n \models q$. These two interpretations differ on the Kripke structure that produces exactly two paths ρ_1 and ρ_2 such that p and q hold at the following positions (p holds nowhere except at position 1 in ρ_1 and position 3 in ρ_2, and q holds everywhere except position 2 in ρ_1 and position 4 in ρ_2):

in ρ_1:	$\{\bar{p}, q\}$	$\{p, q\}$	$\{\bar{p}, \bar{q}\}$	$\{\bar{p}, q\}$	$\{\bar{p}, q\}$	$\{\bar{p}, q\}$	$\ldots$
in ρ_2:	$\{\bar{p}, q\}$	$\{\bar{p}, q\}$	$\{\bar{p}, q\}$	$\{p, q\}$	$\{\bar{p}, \bar{q}\}$	$\{\bar{p}, q\}$	$\ldots$
	0	1	2	3	4	5	

Note that the two paths agree on their initial position, and we can construct a Kripke structure that produces exactly those two paths. It is easy to see that the formula $(Fp \vee Gq)\forall$ does not hold according to the first interpretation (indeed, for $m = 1$ we can take $n = 4$ and consider the path ρ_2 where p does not hold at position 1 and q does not hold at position 4, and for all other values of m, take $n = 2$ and consider the path ρ_1 where p does not hold at position m and q does not hold at position 2), but it does hold according to the second interpretation (for $n = 2$ take $m = 1$, for $n = 4$ take $m = 3$, and for all other values of n take arbitrary value of m, for example $m = n$). The trouble is that the order of the existential quantifier (associated to the left operand Fp) and the universal quantifier (associated to the right operand Gq) actually matters in the semantics of the formula, leading to an annoying situation that $(Fp \vee Gq)\forall$ is not equivalent to $(Gq \vee Fp)\forall$ in any of the interpretations. One way could be to use the branching Henkin quantifiers, like $\binom{\exists m}{\forall n}$ where the existential choice of m does not depend on the universal choice of m. This interpretation suffers from lack of symmetry, as the negation of such a branching Henkin quantifier is in general not expressible as a branching Henkin quantifier [3].

7 Conclusion

The logic CTL+Sync and its extensions presented in this paper provide an elegant framework to express non-regular properties of synchronization. It is intriguing that the exact optimal complexity of the model-checking problem remains open, specially even for the fixed formula $p\mathcal{U}\exists\, q$ (which we show is in $\mathrm{NP}^{\mathrm{NP}}$, and DP-hard). Extending CTL+Sync to an elegant logic *à la* CTL* seems challenging. One may want to express natural properties with the flavor of synchronization, such as the existence of a fixed number of synchronization points, or the property that all paths synchronize in either of a finite set of positions, etc. (see also Section 6). Another direction is to consider alternating-time temporal logics (ATL [1]) with synchronization. ATL is a game-based extension of CTL for which the model-checking problem remains in polynomial time. For instance, ATL can express the existence of a winning strategy in a two-player reachability game. For the synchronized version of reachability games (where the objective for a player is to reach a target state after a number of steps that can be fixed by this player, independently of the strategy of the other player), it is known that deciding the winner is PSPACE-complete [15]. Studying general game-based logics such as ATL or strategy logic [7] combined with quantifier exchange is an interesting direction for future work.

Acknowledgment. We thank Stefan Göller and anonymous reviewers for their insightful comments and suggestions.

References

1 R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.

2 E. Bach and J. Shallit. *Algorithmic Number Theory, Vol. 1: Efficient Algorithms*. MIT Press, 1996.

3 A. Blass and Y. Gurevich. Henkin quantifiers and complete problems. *Ann. Pure Appl. Logic*, 32:1–16, 1986.

4 M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theor. Comput. Sci.*, 59:115–131, 1988.

5 J. Černý. Poznámka k. homogénnym experimentom s konečnými automatmi. In *Matematicko-fyzikálny Časopis*, volume 14(3), pages 208–216, 1964.

6 K. Chatterjee and L. Doyen. Computation tree logic for synchronization properties. *CoRR*, arXiv:1604.06384, 2016.

7 K. Chatterjee, T. A. Henzinger, and N. Piterman. Strategy logic. *Inf. Comput.*, 208(6):677–693, 2010.

8 D. Chistikov, P. Martyugin, and M. Shirmohammadi. Synchronizing automata over nested words. In *Proc. of FOSSACS: Foundations of Software Science and Computation Structures*, LNCS 9634, pages 252–268. Springer, 2016.

9 A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NUSMV: A new symbolic model checker. *STTT*, 2(4):410–425, 2000.

10 E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 2001.

11 M. R. Clarkson, B. Finkbeiner, M. Koleini, K. K. Micinski, M. N. Rabe, and C. Sánchez. Temporal logics for hyperproperties. In *Proceedings of POST: Principles of Security and Trust*, LNCS 8414, pages 265–284. Springer, 2014.

12 M. R. Clarkson and F. B. Schneider. Hyperproperties. *Journal of Computer Security*, 18(6):1157–1210, 2010.

13 S. A. Cook. The complexity of theorem proving procedures. In *Proc. of STOC: Symposium on the Theory of Computing*, pages 151–158. ACM Press, 1971.

14 L. Doyen, L. Juhl, K. G. Larsen, N. Markey, and M. Shirmohammadi. Synchronizing words for weighted and timed automata. In *Proc. of FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, LIPIcs, pages 121–132. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.121.

15 L. Doyen, T. Massart, and M. Shirmohammadi. Limit synchronization in Markov decision processes. In *Proc. of FoSSaCS: Foundations of Software Science and Computation Structures*, LNCS 8412, pages 58–72. Springer-Verlag, 2014.

16 L. Doyen, T. Massart, and M. Shirmohammadi. Robust synchronization in Markov decision processes. In *Proc. of CONCUR: Concurrency Theory*, volume LNCS 8704, pages 234–248. Springer, 2014.

17 J. Kretínský, K. G. Larsen, S. Laursen, and J. Srba. Polynomial time decidability of weighted synchronization under partial observability. In *Proc. of CONCUR: Concurrency Theory*, volume 42 of *LIPIcs*, pages 142–154. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.CONCUR.2015.142.

18 A. Kučera and Jan Strejček. The stuttering principle revisited. *Acta Inf.*, 41(7-8):415–434, 2005.

19 K. G. Larsen, S. Laursen, and J. Srba. Synchronizing strategies under partial observability. In *Proc. of CONCUR: Concurrency Theory*, LNCS 8704, pages 188–202. Springer, 2014.

20 G. Lenzi. Recent results on modal mu-calculus: a survey. *Rend. Istit. Mat. Univ. Trieste*, 42(2):235–255, 2010.

21 H. Spakowski. *Completeness for Parallel Access to NP and Counting Class Separations*. PhD thesis, Heinrich-Heine-Universität Düsseldorf, 2005.

22 L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In *Proc. of STOC: Symposium on Theory of Computing*, pages 1–9. ACM, 1973.

23 W. Thomas. Automata on infinite objects. In *Handbook of Theoretical Computer Science, Vol. B: Formal Models and Sematics*, pages 133–192. MIT Press, 1990.

24 W. Thomas. Languages, automata, and logic. In *Handbook of Formal Languages, Vol. 3: Beyond Words*, pages 389–455. Springer, 1997.

25 M. V. Volkov. Synchronizing automata and the Černý conjecture. In *Proc. of LATA: Language and Automata Theory and Applications*, LNCS 5196, pages 11–27. Springer, 2008.

26 K. W. Wagner. More complicated questions about maxima and minima, and some closures of NP. *Theor. Comput. Sci.*, 51:53–80, 1987.

27 P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.

Deciding the Topological Complexity of Büchi Languages

Michał Skrzypczak[*1] **and Igor Walukiewicz**[2]

1 **Institute of Informatics, University of Warsaw, Warsaw, Poland**
mskrzypczak@mimuw.edu.pl
2 **LABRI, Bordeaux, France**
igw@labri.fr

Abstract

We study the topological complexity of languages of Büchi automata on infinite binary trees. We show that such a language is either Borel and WMSO-definable, or Σ^1_1-complete and not WMSO-definable; moreover it can be algorithmically decided which of the two cases holds. The proof relies on a direct reduction to deciding the winner in a finite game with a regular winning condition.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases tree automata, non-determinism, Borel sets, topological complexity, decidability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.99

1 Introduction

The class of regular languages of infinite trees is one of the most important classes of properties of infinite computations. Similarly to the arithmetic hierarchy, the class is structured into the so called Mostowski–Rabin index hierarchy. This hierarchy reflects the complexity of a language in terms of an alternation of fix-points needed to express it, or equivalently, in terms of the minimal complexity of the acceptance condition of an automaton accepting the language. While we know for about two decades that the hierarchy is infinite [5], we are still very far from understanding it. One important objective in this area is to effectively characterise every level of the hierarchy: for a given regular language of infinite trees calculate its level in the hierarchy.

The difficulty in understanding the Mostowski–Rabin index hierarchy of tree languages is linked to the lack of deterministic acceptors for such languages. Thus, on a smaller scale, we face here the same problem as in the complexity theory, namely the problem of understanding the structure of non-deterministic computations. When restricted to deterministic acceptors, the Mostowski–Rabin hierarchy is by now well-understood. For every level we know a pattern such that the pattern appears in a deterministic tree automaton if and only if the language recognised by this automaton is hard for this level [21, 18, 19]. The pattern method has been extended to the so called game automata [12], but there is no hope to use it for non-deterministic automata.

Apart from decidability questions, a promising way to understand the Mostowski–Rabin hierarchy is to relate it to the topological hierarchy. (For an introduction to the classes

* The first author has been supported by Poland's National Science Center grant (decision DEC-2012/07/D/ST6/02443).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 99; pp. 99:1–99:13

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

of topological complexity see for instance [15].) Topological properties of sets defined by automata are discussed in [24]. It is well-known that all regular languages of infinite trees are contained in the $\mathbf{\Delta}^1_2$ level of the projective hierarchy. The languages of Büchi automata, or equivalently those definable in existential MSO logic, are contained in the $\mathbf{\Sigma}^1_1$ level. The languages of weak-alternating automata, or equivalently definable in weak MSO (WMSO), are Borel; moreover for every finite level of the Borel hierarchy there is a complete weakly definable language [23]. In [23], Skurczyński asks if every regular language that is Borel is necessarily weakly definable. In this paper we answer this question for languages recognised by Büchi automata, as expressed by the main theorem.

▶ **Theorem 1.** *If $\mathcal{B}$ is a non-deterministic Büchi tree automaton then one of the following possibilities holds and it is possible to effectively decide which one it is:*
1. $\mathrm{L}(\mathcal{B})$ *is Borel and* WMSO*-definable,*
2. $\mathrm{L}(\mathcal{B})$ *is $\mathbf{\Sigma}^1_1$-complete and not* WMSO*-definable.*

The theorem is proved through a game construction. Given a Büchi automaton $\mathcal{B}$ we construct a finite game $\mathcal{F}(\infty)$ such that if $\exists$ wins in this game then the language of $\mathcal{B}$ is $\mathbf{\Sigma}^1_1$-complete; but if $\forall$ wins then the language of $\mathcal{B}$ can be accepted by a weak alternating automaton constructed from $\mathcal{B}$. A similar technique of relying on the finite memory determinacy of regular games was used in [1].

Related work. Colcombet, Kuperberg, Löding, and Vanden Boom [16, 8] have proved the algorithmic part of the above theorem; using some decidability result in the theory of cost functions and a reduction of Colcombet and Löding [9] they have shown how to decide if the language of a Büchi automaton is weakly definable. The topological counterpart of Theorem 1 seems not to follow from their construction. Our proof relies only on standard facts from automata theory, and may be simpler, at least for those who are not familiar with the theory of cost functions.

Finding effective characterisations of various classes of infinite tree languages is an important topic of language theory. As we noted above, for languages of deterministic tree automata the situation is quite well-understood; but for the case of all regular tree languages for some time it was only known how to decide if a given regular language can be accepted by an automaton with a trivial acceptance condition [17, 25]. The theory of cost functions allows to decide if a given language can be accepted by a non-deterministic co-Büchi automaton [8]. Bojańczyk and Idziaszek [2] have recently shown decidability of definability in a temporal logic EF. Bojańczyk and Place [4] show how to decide if a given language is a Boolean combination of open sets.

The study of topological properties of regular languages of trees has seen important advances too [10, 13, 6, 18, 19, 3, 11]. Over a decade ago an interesting gap property has been observed [20] for languages of deterministic tree automata: a language is either $\mathbf{\Pi}^1_1$-complete, or contained in the $\mathbf{\Pi}^0_3$ level of the Borel hierarchy. A similar gap property has been recently shown for languages of thin trees [14]: a regular language of thin trees, treated as a subset of all trees, is either definable in weak MSO logic or $\mathbf{\Pi}^1_1$-complete. Our theorem shows a gap property for languages of non-deterministic Büchi automata.

2 Preliminaries and an outline of the construction

We write ω for the natural numbers, and $\overline{\omega}$ for the extension of ω with a greatest element ∞. So $n < \infty$ for all $n \in \omega$, and $\infty - 1 = \infty$. We assume standard definitions of non-deterministic

and alternating automata on infinite binary trees. All trees we consider are binary, the two directions are $\mathtt{L}$ and $\mathtt{R}$. The root of a tree is ϵ. By $\preceq$ we denote the prefix order on the nodes of a tree. In this paper we will use perfect information two player games of infinite duration. The players are denoted $\exists$ (Eve) and $\forall$ (Adam).

A non-deterministic Büchi automaton is a tuple $\mathcal{B} = \langle Q, A, q_{\mathrm{I}} \in Q, \delta \subseteq Q \times A \times Q \times Q, F \subseteq Q \rangle$. We will use the standard notion of a run ρ over a tree t. A run of $\mathcal{B}$ is accepting if on every branch some state from F appears infinitely often. A weak alternating automaton is very similar but for a total quasi-order on states, and the transition relation that now sends a non-empty set of states in every direction $\delta \subseteq Q \times A \times \mathsf{P}^+(Q) \times \mathsf{P}^+(Q)$ (by $\mathsf{P}^+(X)$ we denote the set of non-empty subsets of X). The transition relation should respect the order on the states in a sense that if $(q, a, S_{\mathtt{L}}, S_{\mathtt{R}}) \in \delta$ then all the states in $S_{\mathtt{L}} \cup S_{\mathtt{R}}$ should not be bigger than q, and if $q \in F$ is accepting then all not accepting states in $S_{\mathtt{L}} \cup S_{\mathtt{R}}$ should be strictly smaller than q in this order. A weak alternating automaton $\mathcal{A}$ induces a game on every tree t: the positions of this game are (u, q) with $u \in \{\mathtt{L}, \mathtt{R}\}^*$ and $q \in Q$; the initial position is $(\epsilon, q_{\mathrm{I}})$; from a position (u, q) first $\exists$ chooses a transition $(q, t(u), S_{\mathtt{L}}, S_{\mathtt{R}})$, then $\forall$ chooses a direction d and a state $q' \in S_d$, the successive position is (ud, q'). A play is won by $\exists$ if it contains infinitely many accepting states. Without loss of generality we assume that all the considered automata are complete: for every state $q \in Q$ and every letter $a \in A$ there is at least one transition from q over a. For an automaton $\mathcal{A}$, by $\mathrm{L}(\mathcal{A})$ we denote the set of trees accepted by $\mathcal{A}$.

Our proof of Theorem 1 will use two games, or rather game families constructed from $\mathcal{B}$. The first game, $\mathcal{G}(t)$, will be played on a tree t. The game will encode in a compact way not only the acceptance of t by $\mathcal{B}$, but also possible approximations of $\mathcal{B}$ by weak automata. It is motivated by the technical core of the construction in [22]. More precisely, for every $K \in \overline{\omega}$ we will have a variant $\mathcal{G}(t, K)$ of the game. Each game defines a language of trees

$$\mathrm{L}(\mathcal{G}, K) = \{t \mid \exists \text{ wins } \mathcal{G}(t, K)\}. \tag{1}$$

The game $\mathcal{G}(t, \infty)$ will encode the acceptance of t by $\mathcal{B}$, i.e., $\mathrm{L}(\mathcal{G}, \infty) = \mathrm{L}(\mathcal{B})$. For every $K \in \omega$, the game $\mathcal{G}(t, K)$ will encode the acceptance of t by some specific weak alternating automaton obtained from $\mathcal{B}$; in particular $\mathrm{L}(\mathcal{G}, K)$ will be WMSO-definable. The parameter K will control the quality of the approximation of $\mathcal{B}$, in a sense that

$$\mathrm{L}(\mathcal{G}, 0) \supseteq \mathrm{L}(\mathcal{G}, 1) \supseteq \cdots \supseteq \mathrm{L}(\mathcal{B}) \,.$$

We will show that $\mathrm{L}(\mathcal{B})$ is WMSO-definable if and only if $\mathrm{L}(\mathcal{B}) = \mathrm{L}(\mathcal{G}, K)$ for some finite bound $K \in \omega$. Moreover, we will show that a candidate K_0 for this bound can be computed from $\mathcal{B}$. These results will be obtained from the analysis of another game that we call $\mathcal{F}$.

The game $\mathcal{F}$, and its variants $\mathcal{F}(K)$ for all $K \in \overline{\omega}$, will be central for our arguments. For every $K \in \overline{\omega}$, the game $\mathcal{F}(K)$ will be finite in a sense that there will be a finite number of positions reachable from the initial position. The game $\mathcal{F}(K)$ will in some sense simulate $\mathcal{G}(t, K)$ for an unknown t. We will show that when $K \in \omega$ is too small for $\mathcal{B}$, i.e. when $\mathrm{L}(\mathcal{G}, K) \supsetneq \mathrm{L}(\mathcal{B})$, then $\exists$ has a winning strategy in $\mathcal{F}(K)$. Next, we examine $\mathcal{F}(\infty)$ and show that the winner in this game determines if $\mathrm{L}(\mathcal{B})$ is WMSO-definable. If $\exists$ does not win in $\mathcal{F}(\infty)$ then she does not win in $\mathcal{F}(K_0)$ for some K_0 computable from $\mathcal{B}$ (Proposition 10). Thus $\mathrm{L}(\mathcal{B}) = \mathrm{L}(\mathcal{G}, K_0)$ is WMSO-definable. The most difficult part of the proof is to show that if $\exists$ wins in $\mathcal{F}(\infty)$ then $\mathrm{L}(\mathcal{B})$ is $\mathbf{\Sigma}^1_1$-complete and thus not WMSO-definable (Proposition 11).

The way in which the game $\mathcal{F}$ is obtained from $\mathcal{G}$ is motivated by the concept of *history determinism* and in particular by the combinatorial structure of *domination games*, see [7].

3 The game $\mathcal{G}(t)$

Let us start with the game $\mathcal{G}(t)$. The positions of $\mathcal{G}(t)$ are of the form (q, u, K, z) where $q \in Q^{\mathcal{B}}$ is a state, $u \in \{\mathtt{L}, \mathtt{R}\}^*$ is a node of t, $K \in \overline{\omega}$ is a counter value, and z is one of the three special symbols: *choice*, *safe*, or *reach*. The z component determines the possible choices from a position (q, u, K, z):

- $z = \textbf{\textit{choice}}$: In this case $\forall$ chooses $z' \in \{safe, reach\}$. If he chooses $z' = safe$ then $K' = K$, if he chooses $z' = reach$ then $K' = K - 1$; in particular if $K = 0$ then $\forall$ has to choose $z' = safe$. The game proceeds to the position (q, u, K', z').
- $z = \textbf{\textit{safe}}$: First $\exists$ proposes a transition of $\mathcal{B}$ of the form $(q, t(u), q_{\mathtt{L}}, q_{\mathtt{R}})$ and then $\forall$ chooses a direction $d \in \{\mathtt{L}, \mathtt{R}\}$. The game proceeds to the position $(q_d, ud, K, choice)$.
- $z = \textbf{\textit{reach}}$: First $\exists$ proposes a transition of $\mathcal{B}$ of the form $(q, t(u), q_{\mathtt{L}}, q_{\mathtt{R}})$ and then $\forall$ chooses a direction $d \in \{\mathtt{L}, \mathtt{R}\}$. If q_d is an accepting state then the game proceeds to the position $(q_d, ud, K, choice)$, otherwise it proceeds to the position $(q_d, ud, K, reach)$.

Every play of $\mathcal{G}(t)$ is infinite. Such a play is won by $\forall$ if $z = reach$ from some point on. In the opposite case (i.e. if $z = choice$ infinitely many times) $\exists$ wins.

As we can see from the definition, the game proceeds in phases. It is $\forall$ who chooses if the game should be in a *safe* phase or in a *reach* phase. In the *safe* phase players just construct a path from a run of $\mathcal{B}$ ignoring the acceptance condition. In the *reach* phase $\exists$ needs to provide a finite part of a run until the next accepting states. The counter K gives a bound on the number of *reach* phases: each time $\forall$ chooses *reach*, K is decreased. Notice that if K is ∞ then it stays ∞ during the whole play, so there can be infinitely many *reach* phases.

For $K \in \overline{\omega}$ we denote by $\mathcal{G}(t, K)$ the game $\mathcal{G}(t)$ with the initial position $(q_{\mathrm{I}}^{\mathcal{B}}, \epsilon, K, choice)$.

▶ **Example 2.** For our running example we consider trees over the alphabet $\{a, b\}$ and a Büchi automaton $\mathcal{B}$ accepting the trees with a branch having infinitely many occurrences of a. This automaton has three states $q_a, q_b, \top$, with both q_a, and $\top$ accepting. For a state q_x the transition relation $\delta^{\mathcal{B}}$ on a letter y contains the transitions $(q_x, y, q_y, \top)$ and $(q_x, y, \top, q_y)$; for $x, y \in \{a, b\}$. From $\top$ the automaton stays in $\top$ on every letter and in every direction.

Using the notation from (1) we can see that for $K = 0$ the language $L(\mathcal{G}, K)$ is simply the language of all trees. For $K > 0$, it is the language of trees having a path such that every node on this path has a descendant whose label is a and the subtree rooted in this descendant is in $\mathrm{L}(\mathcal{G}, K - 1)$.

The next three lemmas give connections between the game $\mathcal{G}(t)$ and the automaton $\mathcal{B}$. They refer to the languages $\mathrm{L}(\mathcal{G}, K)$ of the game as defined in (1).

▶ **Lemma 3.** $\mathrm{L}(\mathcal{B}) = \mathrm{L}(\mathcal{G}, \infty)$.

▶ **Lemma 4.** *If $\exists$ wins $\mathcal{G}(t)$ from a position (q, u, K, z) and $K' \leq K \in \overline{\omega}$ then $\exists$ wins $\mathcal{G}(t)$ from the position (q, u, K', z). In other words $\mathrm{L}(\mathcal{G}, K') \supseteq \mathrm{L}(\mathcal{G}, K) \supseteq \mathrm{L}(\mathcal{G}, \infty)$.*

▶ **Lemma 5.** *For every $K \in \omega$ the set of trees $\mathrm{L}(\mathcal{G}, K)$ can be recognised by a weak alternating automaton (or equivalently, it is wmso-definable).*

4 The game $\mathcal{F}$

We proceed to a definition of the game $\mathcal{F}$, and its variants $\mathcal{F}(K)$, for $K \in \overline{\omega}$. For all $K \in \overline{\omega}$, the game $\mathcal{F}(K)$ will simulate $\mathcal{G}(t, K)$ with an unknown t generated *on-the-fly*.

Let us fix a non-deterministic parity tree automaton $\mathcal{A}$ recognising the complement of $\mathrm{L}(\mathcal{B})$ ($\mathcal{A}$ may not be equivalent to a Büchi automaton). We will construct $\mathcal{F}$ from $\mathcal{A}$ and $\mathcal{B}$. Intuitively, in $\mathcal{F}$ we ask the players to proceed as follows:

- $\exists$ should successively construct a tree t and a run $\rho^{\mathcal{A}}$ of $\mathcal{A}$ on t;
- at the same time $\forall$ should select directions in this tree constructing an infinite branch α of t, aiming to show that the run $\rho^{\mathcal{A}}$ proposed by $\exists$ is not accepting;
- moreover $\exists$ would aim at showing that the tree t she constructs is difficult in a sense that she can win $\mathcal{G}(t, K)$ for all finite K. $\forall$ will aim to refute this claim, by showing that he can win with some finite K. It is crucial for our argument that $\forall$ can do this in a *history-deterministic* way in the sense of [7].

4.1 Positions and multi-transitions

The positions of $\mathcal{F}$ are of the form (S, p, κ, r) where:

- $S \in \mathsf{P}(Q^{\mathcal{B}} \times \{\mathit{safe}, \mathit{reach}\})$ is a set of *active states*,
- $p \in Q^{\mathcal{A}}$ is a state of the automaton $\mathcal{A}$,
- $\kappa \colon S \to \overline{\omega}$ assigns to the active states their *counter values*,
- $r \in \{0, 1, 2\}$ is a *sub-round number*.

Using the first and the third component $\exists$ will try to prove that she wins in all the games $\mathcal{G}(t, K)$. In the second component she will construct a run of $\mathcal{A}$. The fourth component makes the definition of the game more modular.

Similarly as before, for $K \in \overline{\omega}$ by $\mathcal{F}(K)$ we denote the game $\mathcal{F}$ with the initial position $(\{(q_{\mathrm{I}}^{\mathcal{B}}, \mathit{safe})\}, q_{\mathrm{I}}^{\mathcal{A}}, \kappa, 0)$ where $\kappa(q_{\mathrm{I}}^{\mathcal{B}}, \mathit{safe}) = K$.

We say that an active state (q, z) is *in the safe phase* if $z = \mathit{safe}$; and *in the reach phase* if $z = \mathit{reach}$. A pair (s, s') *changes phases* if s and s' are in different phases. So (s, s') can *change phases from safe to reach*, or *change phases from reach to safe*.

The edges of $\mathcal{F}$ will have an additional structure (i.e. an edge will be more than just a pair of positions of the game). This richer structure will be used to define the winning condition of $\mathcal{F}$ that will refer to a sequence of edges. From our definition it will be easy to see how to transform such a game into a standard two player game. To underline that edges have additional structure we refer to them as *multi-transitions*.

A *multi-transition* μ from a position (S, p, κ, r) to a position (S', p', κ', r') contains:

- the *pre-state* (S, p, κ, r),
- the *post-state* (S', p', κ', r') with $r' = r + 1 \mod 3$,
- a set $e \subseteq S \times S'$ of *edges* from the active states in S to the active states in S',
- a set $\bar{e} \subseteq e$ of *boldfaced edges*, with exactly one boldfaced edge leading to every $s' \in S'$:

$$\forall_{s' \in S'} \ \big|\{s : (s, s') \in \bar{e}\}\big| = 1 \tag{2}$$

We additionally require the following condition on κ, called *boldface-decreasing*. Assume that $(s, s') \in \bar{e}$. If (s, s') changes phases from *safe* to *reach* then[1] $\kappa'(s') = \max(\kappa(s) - 1, 0)$. Otherwise $\kappa'(s') = \kappa(s)$.

An example multi-transition is depicted in Figure 1. The role of e is to trace the origins of each active state in a similar way as for determinisation of Büchi automata. With the boldfaced edges $\forall$ will indicate which of the possible origins of an active state he finds the most promising for him. The *boldface-decreasing* condition says that on boldfaced traces the counter should behave in the same way as in a game $\mathcal{G}(t, K)$ for the tree t being constructed. The exact rules how the multi-transitions are selected by the players are given in Section 4.2.

[1] By the rules of the game, we shall never have $\kappa(s) < 1$ in this case.

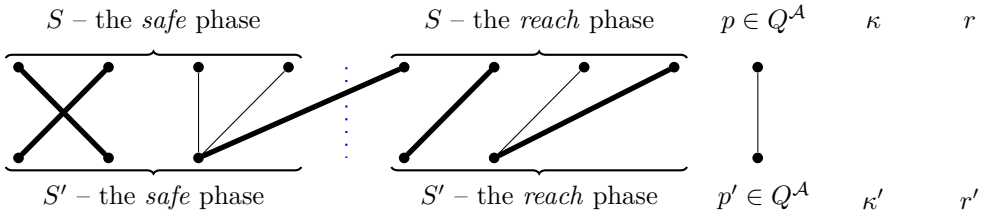


Figure 1 An example multi-transition μ. The big dots correspond to the active states and the state of $\mathcal{A}$. The convention is that all the active states from the *safe* phase are drawn on the left, all the active states from the *reach* phase are drawn in the middle, then comes the state of $\mathcal{A}$, and finally the counter values κ and the sub-round number r are drawn on the right. For the purpose of the layout, we additionally draw an edge between the states p and p' of $\mathcal{A}$ (this edge does not belong to e). The dotted line separates the *safe* phase from the *reach* phase. Boldfaced edges are boldfaced. Every active state in S' has one incoming boldfaced edge as required by (2).

Notice that for a fixed $K \in \overline{\omega}$ the game $\mathcal{F}(K)$ has only finitely many reachable positions and finitely many multi-transitions – if $K = \infty$ then all the counter values κ are equal ∞, otherwise the counter values in the reachable positions are at most K.

4.2 Rules of the game $\mathcal{F}$

In this section we describe the rules of the game $\mathcal{F}$. From a position (S, p, κ, r) the players interact constructing a new position (S', p', κ', r') and a multi-transition between the two positions. For this they select a set of edges $e \subseteq S \times (Q^{\mathcal{B}} \times \{safe, reach\})$ and a state $p' \in Q^{\mathcal{A}}$ according to the rules given below. Then $\forall$ chooses an arbitrary multi-transition μ that *respects* (S, p, κ, r), e, *and* p' in the following sense:

- the pre-state of μ is (S, p, κ, r),
- the post-state of μ is (S', p', κ', r'); where $S' = \{s' : (s, s') \in e\}$ consists of the targets of the edges e, κ' is determined by the boldface-decreasing condition, and $r' = r + 1 \mod 3$,
- the edges of μ are e,
- the boldfaced edges $\bar{e}$ of μ can be chosen arbitrarily by $\forall$ subject to condition (2).

Observe that a multi-transition μ that respects (S, p, κ, r), e, and p' is unique but for the choice of the boldfaced edges $\bar{e}$.

Assume that the current position in $\mathcal{F}$ is (S, p, κ, r) and consider the following cases depending on the number of the sub-round r. In all the cases the players construct a multi-transition μ that leads to a post-state (S', p', κ', r'):

(R0) $r = 0$**:** There are two cases. If the *reach* phase is not empty i.e. $S \cap (Q^{\mathcal{B}} \times \{reach\}) \neq \emptyset$, then e contains all the pairs (s, s) for $s \in S$. The second case is when there are no states in the *reach* phase. We call this situation a *flush*. In that case $\forall$ can choose[2] any set $C \subseteq Q^{\mathcal{B}}$ of states q such that

$$(q, safe) \in S \quad \text{and} \quad \kappa(q, safe) > 0 \ . \tag{3}$$

The chosen active states get copied to the *reach* phase thanks to the edge relation defined as: $e = \{(s, s) \mid s \in S\} \cup \big\{\big((q, safe), (q, reach)\big) \mid q \in C\big\}$. In both cases the state $p' = p$ of $\mathcal{A}$ is not changed and $\forall$ chooses μ that respects (S, p, κ, r), e, and p'.

[2] Even if $\forall$ declares $C = \emptyset$, the fact that the *reach* phase was empty implies that we have a *flush*.

(R1) $r = 1$**:** $\exists$ declares: (i) a letter $a \in A$; (ii) a transition $\delta_s = (q, a, q^s_{\mathtt{L}}, q^s_{\mathtt{R}})$ of $\mathcal{B}$, for every $s = (q, z) \in S$; (iii) a transition $\delta = (p, a, p'_{\mathtt{L}}, p'_{\mathtt{R}})$ of $\mathcal{A}$. Then $\forall$ responds by selecting a direction $d \in \{\mathtt{L}, \mathtt{R}\}$. We put $p' = p'_d$, and e contains all the pairs of the form $((q, z), (q^s_d, z))$ for $s = (q, z) \in S$. $\forall$ chooses μ that respects (S, p, κ, r), e, and p'.

(R2) $r = 2$**:** Deterministically, every active state (q, reach) in the *reach* phase with q accepting (i.e. $q \in F$) is moved to the *safe* phase. Formally, for each $(q, z) \in S$, the relation e contains pairs $((q, z), (q, z'))$ such that either: (i) $z = z' = \mathit{safe}$; or (ii) $z = z' = \mathit{reach}$ and $q \notin F$; or (iii) $z = \mathit{reach}$, $z' = \mathit{safe}$, and $q \in F$. The state $p' = p$ of $\mathcal{A}$ is not changed. $\forall$ chooses μ that respects (S, p, κ, r), e, and p'.

If $(s, s') \in e$ we say that s' is a *μ-successor* of s. By the definition of the sub-rounds of the game, we obtain the following fact.

▶ **Fact 6.** *Every active state has between one and two μ-successors. The only case when an active state (q, z) can have two μ-successors is when $r = 0$, $z = \mathit{safe}$, and we have a* flush*.*

4.3 The winning condition of $\mathcal{F}$

Now we will define the winning condition for $\exists$ in $\mathcal{F}$. It will depend on the sequence of multi-transitions $\pi = \mu_0\mu_1 \dots$ that were played in $\mathcal{F}$. We will refer to the pre-state of μ_n as $(S_n, p_n, \kappa_n, r_n)$. Analogously, we will use $(S'_n, p'_n, \kappa'_n, r'_n)$ for the post-state, e_n for the edges, and $\bar{e}_n$ for the boldfaced edges of μ_n, respectively. Since π is a play, $(S'_n, p'_n, \kappa'_n, r'_n) = (S_{n+1}, p_{n+1}, \kappa_{n+1}, r_{n+1})$ and $r_n \equiv n \mod 3$.

A *trace in π* is a sequence $\alpha = s_0, s_1, \dots$ such that $(s_i, s_{i+1}) \in e_i$ of all i. A trace is *boldfaced* if $(s_i, s_{i+1}) \in \bar{e}_i$ for all i. For every $s \in S'_n$ there is a boldfaced trace ending in s, and it is unique due to condition (2); we call it the *boldfaced history of s in π*.

The winning condition will be a boolean combination of three properties of plays. We list them separately as they will be of independent interest in the proof.

W1. Infinitely many times there is a *flush* in the sub-round (R0).
W2. Some boldfaced trace changes phases infinitely many times.
W3. The sequence of states $p_0, p_1, \dots$ of the automaton $\mathcal{A}$ is accepting.
Now we declare a play to be winning for $\exists$ if it satisfies

$$\mathrm{W1} \ \wedge \ (\mathrm{W2} \vee \mathrm{W3}). \tag{4}$$

Note that Condition W2 implies Condition W1 – if some trace in a play changes phases infinitely often then the play must have infinitely many times a *flush*.

Intuitively, Condition W1 expresses that $\exists$ has not stayed forever in the *reach* phase – she has reached an accepting state of $\mathcal{B}$ whenever $\forall$ asked for it.

Condition W2 says that $\forall$ has not succeeded to bound the number of changes of phases; so he has failed to prove that on the constructed tree t he can win in $\mathcal{G}(t, K)$ for some finite K. Condition W3 takes care of the situation when the constructed tree is not in $\mathrm{L}(\mathcal{B})$. One can think of it as an escape option for $\exists$. She uses it when $\forall$ plays very cautiously and gives $\exists$ no chance to construct a trace satisfying Condition W2; an extreme example is when $\forall$ never chooses to move some active states to the *reach* phase.

▶ **Example 7.** Let $\mathcal{B}$ be the Büchi automaton from the example on page 4. Consider an automaton $\mathcal{A}$ accepting the complement of $\mathrm{L}(\mathcal{B})$, namely the set of trees with only finitely many a's on every branch. This automaton is a deterministic co-Büchi automaton having two states: p_a and p_b, with p_a being a rejecting state. So a run of $\mathcal{A}$ will be accepting if on every branch the state p_a appears only finitely often. For a state p_x the transition relation $\delta^{\mathcal{A}}$ on a letter $y \in \{a, b\}$ contains the transition (p_x, y, p_y, p_y).

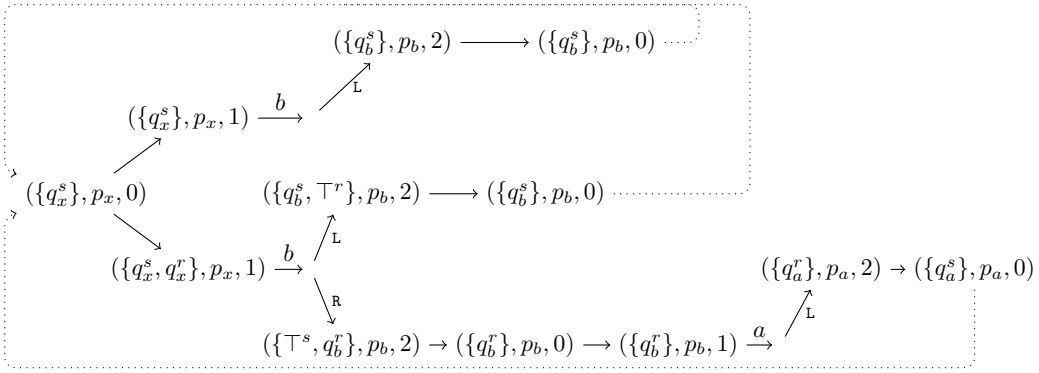


Figure 2 A symbolic representation of a strategy $\sigma_\exists$ of $\exists$. The subscript x in q_x^s or q_x^r stands for a or b. The superscript indicates if a state is in the *safe* phase or in the *reach* phase. Only branches not evidently losing for the player $\forall$ are presented.

We claim that $\exists$ has a winning strategy in $\mathcal{F}(\infty)$ constructed from $\mathcal{B}$ and $\mathcal{A}$. This strategy is schematically presented in Figure 2. For compactness of the notation we omit the third component of the position that is always ∞ and omit active states of the form $(\top, z)$ – it is trivial for $\exists$ to play from them.

The root position is a *flush*, so $\forall$ can choose whether to copy the unique active state to the *reach* phase. If he does not (i.e. he goes up in the picture) then $\exists$ chooses b and plays the transition $(q_x, b, q_b, \top)$. Then it is clear that it is better for $\forall$ to move to the left. The game gets to a similar position as in the root. If $\forall$ constantly chooses to play like this then he will lose as the play will have infinitely many times a *flush* and only states p_b, thus it will satisfy Conditions W1 and W3.

If $\forall$ decides to copy the active state during a *flush* then $\exists$ still chooses b but plays different transitions for the two copies: for the active state in the *safe* phase she chooses $(q_x, b, q_b, \top)$, for the active state in the *reach* phase she chooses $(q_x, b, \top, q_b)$. If $\forall$ chooses the left direction then the play gets to a similar position as in the root. Since there is a *flush*, and $\forall$ does not manage to see a new p_a state, the result is the same as in the case when $\forall$ has not copied the active state. If $\forall$ chooses the right direction then the play reaches a position where the only interesting active state is in the *reach* phase. Then $\exists$ chooses the letter a and the transition $(q_b, a, q_a, \top)$. It is then more interesting for $\forall$ to move to the left. The play reaches a position of the same form as the one in the root. The interesting thing that happens on this path is that the unique boldfaced trace changes phases. So, if $\forall$ chooses infinitely often to copy an active state and then go to the right then the play will satisfy Conditions W1 and W2. Otherwise there will be only finitely many occurrences of the state p_a and $\forall$ will lose since there still will be infinitely many times a *flush*.

We have already noticed that for every $K \in \overline{\omega}$, the game $\mathcal{F}(K)$ is finite. By the definition of Conditions W1, W2, and W3, the winning condition of $\mathcal{F}(K)$ is a regular property of sequences of multi-transitions. By adding multi-transitions of $\mathcal{F}(K)$ to the positions, one can obtain an equivalent game with the winning condition on sequences of positions. So up to presentation, $\mathcal{F}(K)$ is essentially a finite game with a regular winning condition, and we can solve it effectively.

▶ **Fact 8.** *For a fixed $K \in \overline{\omega}$, the winner of $\mathcal{F}(K)$ can be effectively found and he/she can win using a finite memory winning strategy. Let $m_\forall$ be the bound on the size of the memory of $\forall$ needed to win the game $\mathcal{F}(\infty)$.*

5 Characterisation

We show that $\mathcal{F}(\infty)$ characterises when $\mathrm{L}(\mathcal{B})$ is WMSO-definable. This is formulated in the following two propositions that complete the proof of Theorem 1. They rely on the following standard fact, see for instance [24].

▶ **Fact 9.** *If L is a language of a Büchi tree automaton then $L \in \mathbf{\Sigma}^1_1$. If L is a* WMSO*-definable tree language then L is Borel.*

▶ **Proposition 10.** *If $\forall$ wins $\mathcal{F}(\infty)$ then $\mathrm{L}(\mathcal{B})$ is* WMSO*-definable.*

▶ **Proposition 11.** *If $\exists$ wins $\mathcal{F}(\infty)$ then $\mathrm{L}(\mathcal{B})$ is $\mathbf{\Sigma}^1_1$-complete and not* WMSO*-definable.*

In the rest of the section we will outline the proof of Proposition 11. Suppose that $\exists$ wins in $\mathcal{F}(\infty)$. Let us fix a winning strategy $\sigma_\exists$ for $\exists$ in $\mathcal{F}(\infty)$.

We need to prove that $\mathrm{L}(\mathcal{B})$ is $\mathbf{\Sigma}^1_1$-hard, so we will construct an appropriate continuous reduction. Let $\omega\mathrm{Tr}$ denote the space of partial ω-branching trees. Such a tree τ is a non-empty, prefix-closed subset of ω^*. We say that an ω-branching tree τ is *ill-founded* if it contains an infinite branch, i.e. there exists $\alpha \in \omega^\omega$ such that for every $x \prec \alpha$ we have $x \in \tau$. We use IF to denote the set of all ill-founded ω-branching trees. If an ω-branching tree is not ill-founded then it is *well-founded*.

▶ **Fact 12.** IF *is $\mathbf{\Sigma}^1_1$-complete.*

Therefore, it is enough to construct a continuous reduction from IF to $\mathrm{L}(\mathcal{B})$. Our aim is to construct a tree $t(\tau)$ such that $t(\tau) \in \mathrm{L}(\mathcal{B})$ if and only if τ is ill-founded. The tree $t(\tau)$ will be obtained by evaluating the strategy $\sigma_\exists$ against a certain family of strategies of $\forall$, called *τ-genuine strategies*.

Let us explain this point in more detail. A strategy for $\exists$ in $\mathcal{F}(\infty)$ can be seen as a strategy tree where branching represents the choices of $\forall$ (see Figure 2). Recall from the definition of the game that $\forall$ not only chooses directions (in the sub-round (R1)), but also copies active states to the *reach* phase (during a *flush* in the sub-round (R0)), and selects boldfaced edges (in all the sub-rounds). We want to extract from the strategy tree $\sigma_\exists$ a tree where we leave only branching corresponding to the choice of directions. To this end we define τ-genuine strategies of $\forall$, where his choices to copy and to select boldfaced edges are determined by the history of the play so far. This means that a strategy tree for $\exists$ against all τ-genuine strategies of $\forall$ will be a tree with branching corresponding only to the choices of directions by $\forall$. Then we show that we have not restricted the power of $\forall$ too much, namely from this strategy tree for $\exists$ we can read out the required tree $t(\tau)$.

To properly define τ-genuine strategies of $\forall$ we will use the Kleene-Brouwer ordering, see [15, Section 2.G]. For $x, y \in \omega^*$, we say that $x \leq_{\mathrm{KB}} y$ if either: (i) $x \succeq y$, or (ii) for some $n < m < \omega$ and $v, x', y' \in \omega^*$ we have $x = vnx'$ and $y = vmy'$. Intuitively, $x \leq_{\mathrm{KB}} y$ if x is below or to the left of y. ϵ is the $\leq_{\mathrm{KB}}$-maximal element of ω^*. There is no $\leq_{\mathrm{KB}}$-minimal element in ω^*, e.g. for every $x \in \omega^*$ and its 0-successor $x \cdot 0$ we have $x \cdot 0 <_{\mathrm{KB}} x$.

▶ **Fact 13** (See [15, Proposition 2.12])**.** *An ω-branching tree τ is well-founded if and only if $\leq_{\mathrm{KB}}$ is a well-order on the vertices of τ.*

For certain technical reasons it will be useful to have the following construction.

▶ **Definition 14.** First, assume that $\mathrm{list}\colon \omega \to \omega^*$ has the property that for each $x \in \omega^*$, the pre-image $\mathrm{list}^{-1}(\{x\})$ is infinite (i.e. every vertex appears infinitely many times in this enumeration). Now, given $x \in \omega^*$ let $\mathrm{down}(x, n)$ be either $\mathrm{list}(n)$ if $\mathrm{list}(n) <_{\mathrm{KB}} x$ or $x \cdot 0$ otherwise.

▶ **Fact 15.** *The following conditions are satisfied for every $x \in \omega^*$:*

- $\forall_{n\in\omega}\ \mathrm{down}(x,n) <_{\mathrm{KB}} x$,
- *for every $y <_{\mathrm{KB}} x$ there are infinitely many n such that* $\mathrm{down}(x,n) = y$.

Now we can proceed with the definition of τ-genuine strategies. Our aim is to make sure that for every sequence of successive directions $d_0, d_1, \ldots$ played in the sub-rounds (R1), there is a unique τ-genuine strategy of $\forall$. A τ-genuine strategy will depend on certain additional information accumulated during a play, and this information will be related to the ω-branching tree τ. Therefore, $\forall$ will keep track of an *extended position* – a position (S, p, κ, r) of $\mathcal{F}$ together with a mapping $\nu\colon S \to \tau$ and a counter $c \in \omega$. The function ν will measure the progress of every active state with respect to the $\leq_{\mathrm{KB}}$ order over τ. The counter c will count the number of times when a *flush* happened in the play.

The initial extended position of the game is the initial position of $\mathcal{F}(\infty)$ together with ν assigning to $(q_{\mathrm{I}}^{\mathcal{B}}, \mathit{safe})$ the root ϵ of τ; and the counter $c = 0$.

A strategy $\sigma_\forall$ of $\forall$ is called *τ-genuine* if it satisfies the three conditions defined below: *genuine-copying*, *flush-counting*, and *KB-tracking*.

A strategy $\sigma_\forall$ satisfies *genuine-copying* if during a *flush* in the sub-round (R0) it copies an active state s from the *safe* phase to the *reach* phase if and only if $\mathrm{down}(\nu(s), c) \in \tau$.

The condition of *flush-counting* says that $\forall$ increments c by 1 exactly when there is a *flush* in the sub-round (R0); otherwise he keeps the value c unchanged.

The last condition *KB-tracking* determines how the set of boldfaced edges $\bar{e}$ should be chosen and how to update ν. We say that a multi-transition μ from $(S, p, \kappa, r, \nu, c)$ to $(S', p', \kappa', r', \nu', c')$ with edges e and boldfaced edges $\bar{e}$ satisfies the *KB-tracking* condition when:

- If μ is not a *flush* then for every $s' \in S'$:

  $$\nu'(s') = \max_{\leq_{\mathrm{KB}}} \{\nu(s) : (s, s') \in e\}\ .$$

  Moreover, the unique boldfaced edge to s' should come from s_0 realising the maximum above, i.e., $\nu'(s') = \nu(s_0)$ (if there is more than one such s_0 then we choose the smallest one according to some fixed ordering on active states).
- If μ is a *flush* then for every $s' \in S'$ from the *safe* phase, the vertex $\nu'(s')$ of τ and the boldfaced edges are determined as above. For every $s' \in S'$ in the *reach* phase there is a unique $s \in S$ with $(s, s') \in e$. This edge needs to be boldfaced and we set

  $$\nu'(s') = \mathrm{down}(\nu(s), c)\ .$$

Notice that in this last case the node $\mathrm{down}(\nu(s), c)$ is in τ thanks to the *genuine-copying* condition.

▶ **Fact 16.** *Using the above notions, the following inequalities hold:*

- *if $(s, s') \in \bar{e}$ then $\nu(s) \geq_{\mathrm{KB}} \nu'(s')$,*
- *if $(s, s') \in e$ and (s, s') changes phases from safe to reach then $\nu(s) >_{\mathrm{KB}} \nu'(s')$,*
- *if $(s, s') \in e$ and (s, s') does not change phases from safe to reach then $\nu(s) \leq_{\mathrm{KB}} \nu'(s')$.*

▶ **Corollary 17.** *Suppose τ is well-founded. If π is an infinite play of $\mathcal{F}(\infty)$ consistent with a τ-genuine strategy of $\forall$ then π does not satisfy W2 (no boldfaced trace changes phases infinitely many times).*

▶ Remark. Observe that all the choices of $\forall$ except the directions d are uniquely determined in a τ-genuine strategy. Therefore, to define a τ-genuine strategy it is enough to say what will be the directions proposed by $\forall$ in the sub-rounds (R1). For the next definition it is also useful to note that all the maximal plays in $\mathcal{F}(\infty)$ are infinite.

▶ **Definition 18.** For every $\alpha \in \{\mathtt{L},\mathtt{R}\}^\omega$ by $\sigma_{\forall}(\tau,\alpha)$ we denote the unique τ-genuine strategy of $\forall$ that for every $n \in \omega$ plays $d = \alpha(n)$ in the n-th sub-round (R1). Let $\pi(\tau,\alpha)$ be the infinite play of $\mathcal{F}(\infty)$ obtained when $\exists$ is playing $\sigma_{\exists}$ and $\forall$ is playing $\sigma_{\forall}(\tau,\alpha)$.

For every finite prefix $u \prec \alpha$ we denote by $\pi(\tau,u)$ the corresponding prefix of $\pi(\tau,\alpha)$. This play is defined until $\forall$ is asked to determine the $(n{+}1)$-th direction in the sub-round (R1). Let $(S_u, p_u, \kappa_u, r_u, \nu_u, c_u)$ be the extended position of this play at the beginning of the last round (i.e. when $r_u = 0$).

We can finally define the tree $t(\tau)$.

▶ **Definition 19.** We define the tree $t(\tau)$ together with a run $\rho^{\mathcal{A}}(\tau)$ of $\mathcal{A}$. For a vertex $u \in \{\mathtt{L},\mathtt{R}\}^*$, let $t(\tau)(u)$ and $\rho^{\mathcal{A}}(\tau)(u)$ be the letter a and the state p of $\mathcal{A}$ played by $\exists$ in the sub-round (R1) of the last round of the play $\pi(\tau,u)$.

Observe that by the construction, $\rho^{\mathcal{A}}(\tau)$ is a run of $\mathcal{A}$ over $t(\tau)$. Notice also that since the strategy $\sigma_{\forall}(\tau,u)$ queries whether $v \in \tau$ for finitely many v at a time, the function mapping τ to $t(\tau)$ is continuous. We show that indeed the mapping is the required reduction from IF as expressed by the following two lemmas.

▶ **Lemma 20.** *If τ is well-founded then $\rho^{\mathcal{A}}(\tau)$ is accepting and thus $t(\tau) \notin \mathrm{L}(\mathcal{B})$.*

▶ **Lemma 21.** *If τ is ill-founded then $t(\tau) \in \mathrm{L}(\mathcal{B})$.*

Lemma 20 follows from Corollary 17. Consider any infinite branch α of $t(\tau)$ and the corresponding play $\pi(\tau,\alpha)$ of $\sigma_{\exists}$ against $\sigma_{\forall}(\tau,\alpha)$. Since $\sigma_{\exists}$ is winning we know that it satisfies the disjunction W2 $\vee$ W3. By Corollary 17 we know that no play consistent with $\sigma_{\forall}(\tau,\alpha)$ can satisfy Condition W2. Therefore, Condition W3 needs to be satisfied and therefore, the run $\rho^{\mathcal{A}}(\tau)$ is accepting on α.

To prove Lemma 21 we fix an ω-branching ill-founded tree $\tau \in \mathrm{IF}$. We then extract an accepting run $\rho^{\mathcal{B}}$ of $\mathcal{B}$ on $t(\tau)$ from the strategy $\sigma_{\exists}$ in $\mathcal{F}(\infty)$. The crucial point is to make sure that $\forall$ will copy infinitely often the active states of the constructed run to the *reach* phase. For this we need to rely on the condition of *genuine-copying*. Let us now describe how this construction works on our running example.

▶ **Example 22.** Recall the example from page 4 and the winning strategy for $\exists$ from the example on page 7 (see Figure 2). In this strategy $\forall$ has a choice of whether to copy or not the active state q^s_x; this corresponds to going down or up from the root, respectively. Next, $\forall$ chooses a direction. In a τ-genuine strategy a state q^s_x is assigned a node $\nu(q^s_x)$ of τ and c counts the number of flushes. The condition of *genuine-copying* requires $\forall$ to copy the state q^s_x to the *reach* phase if $\mathrm{down}(\nu(q^s_x), c) \in \tau$. Since all loops of $\sigma_{\exists}$ contain a *flush*, the value c counts the number of times either of the loops has been taken.

According to the definition of a τ-genuine strategy, the only moment when the value $\nu(q^s_x)$ may change is when $\forall$ copies (i.e. takes the down successor of the root at Figure 2) and in that case the value $\nu(q^r_x)$ becomes $\mathrm{down}(\nu(q^s_x), c)$ according to the *KB-tracking* condition. This becomes the new value of $\nu(q^s_x)$ if and only if the play then follows the direction $\mathtt{R}$.

If τ is well-founded then there is no infinite $\leq_{\mathrm{KB}}$-descending chain in τ and therefore, for every branch $\alpha \in \{\mathtt{L},\mathtt{R}\}^\omega$, the play $\pi(\tau,\alpha)$ follows only finitely many times the down path of $\sigma_{\exists}$. Therefore, the produced tree $t(\tau)$ contains only finitely many letters a on every branch and $t(\tau) \notin \mathrm{L}(\mathcal{B})$.

Now assume that τ is ill-founded and $v_0 >_{\mathrm{KB}} v_1 >_{\mathrm{KB}} \dots$ is an infinite $\leq_{\mathrm{KB}}$-descending chain of nodes of τ. We will use this sequence to find a branch of $t(\tau)$ that contains infinitely many a's. We start in the root of $t(\tau)$ and keep track of the current vertex $\nu(q^s_x)$ of τ. We will

preserve an invariant that when being in a node $u \in \{\mathtt{L}, \mathtt{R}\}^*$ with n the number of occurrences of a on u in $t(\tau)$ then $\nu(q_x^s) = v_n$ – the vertex of τ pointed to by ν is the n-th vertex in our $\leq_{\mathrm{KB}}$-descending chain. Consider the following two cases:

1. If $\mathrm{down}(\nu(q_x^s), c) = v_{n+1}$ then $\forall$ chooses to copy, i.e., go down from the root. We follow the $\mathtt{R}$-successor in $t(\tau)$. Then $t(\tau)(u\mathtt{R}) = a$ and the game gets to the node $u\mathtt{RL}$. The number of times we have seen an a is incremented (i.e. $n' = n + 1$), and the invariant is preserved since after this loop we have $\nu(q_x^s) = v_{n+1}$.
2. Otherwise either (i) $\mathrm{down}(\nu(q_x^s), c) \notin \tau$ so $\forall$ does not copy, or (ii) $\mathrm{down}(\nu(q_x^s), c) \in \tau$ so $\forall$ copies, but we choose the direction $\mathtt{L}$. In both cases we end up in the left successor of our current node (i.e. in $u\mathtt{L}$). The new value $\nu(q_x^s)$ does not change, neither does n.

Therefore, in both cases the invariant $\nu(q_x^s) = v_n$ is preserved. Since the value of c tends to infinity, Fact 15 tells us that $\mathrm{down}(\nu(q_x^s), c) = v_{n+1}$ will eventually hold, and we will see an a. In the limit, the branch of $t(\tau)$ we follow will have infinitely many letters a.

6 Conclusions

While regular languages of infinite trees are widely used nowadays, their structure is still very poorly understood. The main reason for this is probably the lack of deterministic acceptors for such languages. This paper exhibits a gap property for languages of non-deterministic Büchi tree automata: such a language is either weakly definable, or $\mathbf{\Sigma}^1_1$-complete. Our proof uses a reduction to a finite game. Given a Büchi automaton $\mathcal{B}$, we construct a game $\mathcal{F}(\infty)$ of exponential size w.r.t. $\mathcal{B}$, and with a parity condition of size proportional to the size of $\mathcal{B}$. Thus our reduction gives an EXPTIME decision algorithm. This matches a known lower bound [25].

References

1 Mikołaj Bojańczyk. Star height via games. In *LICS*, pages 214–219, 2015.
2 Mikołaj Bojańczyk and Tomasz Idziaszek. Algebra for infinite forests with an application to the temporal logic EF. In *CONCUR*, pages 131–145, 2009.
3 Mikołaj Bojańczyk, Damian Niwiński, Alexander Rabinovich, Adam Radziwończyk-Syta, and Michał Skrzypczak. On the Borel complexity of MSO definable sets of branches. *Fundamenta Informaticae*, 98(4):337–349, 2010.
4 Mikołaj Bojańczyk and Thomas Place. Regular languages of infinite trees that are Boolean combinations of open sets. In *ICALP*, pages 104–115, 2012.
5 Julian Bradfield. The modal mu-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195:133–153, 1997.
6 Jérémie Cabessa, Jacques Duparc, Alessandro Facchini, and Filip Murlak. The Wadge hierarchy of max-regular languages. In *FSTTCS*, pages 121–132, 2009.
7 Thomas Colcombet. Fonctions régulières de coût. Habilitation thesis, Université Paris Diderot – Paris 7, 2013.
8 Thomas Colcombet, Denis Kuperberg, Christof Löding, and Michael Vanden Boom. Deciding the weak definability of Büchi definable tree languages. In *CSL*, pages 215–230, 2013.
9 Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP (2)*, pages 398–409, 2008.
10 Jacques Duparc, Olivier Finkel, and Jean-Pierre Ressayre. Computer science and the fine structure of Borel sets. *Theoretical Computer Science*, 257(1–2):85–105, 2001.
11 Jacques Duparc, Olivier Finkel, and Jean-Pierre Ressayre. The Wadge hierarchy of Petri nets w-languages. In *LFCS*, pages 179–193, 2013.

12 Alessandro Facchini, Filip Murlak, and Michał Skrzypczak. Rabin-Mostowski index problem: A step beyond deterministic automata. In *LICS*, pages 499–508, 2013.

13 Olivier Finkel. Borel ranks and Wadge degrees of context free omega-languages. *Mathematical Structures in Computer Science*, 16(5):813–840, 2006.

14 Tomasz Idziaszek, Michał Skrzypczak, and Mikołaj Bojańczyk. Regular languages of thin trees. *Theory of Computing Systems*, pages 1–50, 2015.

15 Alexander Kechris. *Classical descriptive set theory*. Springer-Verlag, New York, 1995.

16 Denis Kuperberg and Michael Vanden Boom. Quasi-weak cost automata: A new variant of weakness. In *FSTTCS*, volume 13 of *LIPIcs*, pages 66–77, 2011.

17 Ralf Küsters and Thomas Wilke. Deciding the first level of the μ-calculus alternation hierarchy. In *FST TCS 2002:*, volume 2556 of *LNCS*, pages 241–252, 2002.

18 Filip Murlak. The Wadge hierarchy of deterministic tree languages. *Logical Methods in Logical Methods in Comput. Sci.*, 4(4), 2008.

19 Filip Murlak. Weak index versus Borel rank. In *STACS'08*, LIPIcs, pages 573–584, 2008.

20 Damian Niwiński and Igor Walukiewicz. A gap property of deterministic tree languages. *Theor. Comput. Sci.*, 1(303):215–231, 2003.

21 Damian Niwiński and Igor Walukiewicz. Deciding nondeterministic hierarchy of deterministic tree automata. *Electronic Notes in Theoretical Computer Science*, 123:195–208, 2005.

22 Michael Oser Rabin. Weakly definable relations and special automata. In *Proceedings of the Symposium on Mathematical Logic and Foundations of Set Theory*, pages 1–23. North-Holland, 1970.

23 Jerzy Skurczyński. The Borel hierarchy is infinite in the class of regular sets of trees. *Theoretical Computer Science*, 112(2):413–418, 1993.

24 Wolfgang Thomas and Helmut Lescow. Logical specifications of infinite computations. In *REX School/Symposium*, pages 583–621, 1993.

25 Igor Walukiewicz. Deciding low levels of tree-automata hierarchy. In *Workshop on Logic, Language, Information and Computation*, volume 67 of *Electronic Notes in Theoretical Computer Science*, pages 61–75, 2002.

On the Skolem Problem for Continuous Linear Dynamical Systems

Ventsislav Chonev[*1], **Joël Ouaknine**[†2], **and James Worrell**[‡2]

1 **Institute of Science and Technology, Austria**
2 **University of Oxford, Oxford, United Kingdom**

Abstract

The Continuous Skolem Problem asks whether a real-valued function satisfying a linear differential equation has a zero in a given interval of real numbers. This is a fundamental reachability problem for continuous linear dynamical systems, such as linear hybrid automata and continuous-time Markov chains. Decidability of the problem is currently open – indeed decidability is open even for the sub-problem in which a zero is sought in a bounded interval. In this paper we show decidability of the bounded problem subject to Schanuel's Conjecture, a unifying conjecture in transcendental number theory. We furthermore analyse the unbounded problem in terms of the frequencies of the differential equation, that is, the imaginary parts of the characteristic roots. We show that the unbounded problem can be reduced to the bounded problem if there is at most one rationally linearly independent frequency, or if there are two rationally linearly independent frequencies and all characteristic roots are simple. We complete the picture by showing that decidability of the unbounded problem in the case of two (or more) rationally linearly independent frequencies would entail a major new effectiveness result in Diophantine approximation, namely computability of the Diophantine-approximation types of all real algebraic numbers.

1998 ACM Subject Classification F.1.1 Models of Computation, F.2.1 Numerical Algorithms and Problems

Keywords and phrases differential equations, reachability, Baker's Theorem, Schanuel's Conjecture, semi-algebraic sets

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.100

1 Introduction

The Continuous Skolem Problem is a fundamental decision problem concerning reachability in continuous-time linear dynamical systems. The problem asks whether a real-valued function satisfying an ordinary linear differential equation has a zero in a given interval of real numbers. More precisely, an instance of the problem comprises an interval $I \subseteq \mathbb{R}_{\geq 0}$ with rational endpoints, an ordinary differential equation

$$f^{(n)} + a_{n-1}f^{(n-1)} + \ldots + a_0 f = 0 \tag{1}$$

whose coefficients are real algebraic, together with initial conditions $f(0), \ldots, f^{(n-1)}(0)$ that are also real algebraic numbers. Writing $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$ for the unique solution of the

* Ventsislav Chonev is supported by Austrian Science Fund (FWF) NFN Grant No S11407-N23 (RiSE/SHiNE), ERC Start grant (279307: Graph Games), and ERC Advanced Grant (267989: QUAREM).
† Joël Ouaknine is supported by ERC grant AVS-ISS (648701).
‡ James Worrell is supported by EPSRC grant EP/N008197/1.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 100; pp. 100:1–100:13

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

differential equation subject to the initial conditions, the question is whether there exists $t \in I$ such that $f(t) = 0$. Decidability of this problem is currently open. Decidability of the sub-problem in which the interval I is bounded, called the Bounded Continuous Skolem Problem, is also open [4, Open Problem 17].

The nomenclature *Continuous Skolem Problem* is based on an analogy with the Skolem Problem for linear recurrence sequences, which asks whether a given linear recurrence sequence has a zero term [12]. Whether the latter problem is decidable is an outstanding question in number theory and theoretical computer science; see, e.g., the exposition of Tao [20, Section 3.9].

The continuous dynamics of linear hybrid automata and the evolution of continuous-time Markov chains, amongst many other examples, are determined by linear differential equations of the form $\boldsymbol{x}'(t) = A\boldsymbol{x}(t)$, where $\boldsymbol{x}(t) \in \mathbb{R}^n$ and A is an $n \times n$ matrix of real numbers [1]. A basic reachability question in this context is whether, starting from an initial state $\boldsymbol{x}(0)$, the system reaches a given hyperplane $\{\boldsymbol{y} \in \mathbb{R}^n : \boldsymbol{u}^T\boldsymbol{y} = 0\}$ with normal vector $\boldsymbol{u} \in \mathbb{R}^n$. For example, one can ask whether the continuous flow of a hybrid automaton leads to a particular transition guard being satisfied or an invariant being violated. Now the function $f(t) = \boldsymbol{u}^T\boldsymbol{x}(t)$ satisfies a linear differential equation of the form (1), and it turns out that the hyperplane reachability problem is inter-reducible with the Continuous Skolem Problem (see [4, Theorem 6] for further details). Moreover, under this reduction the Bounded Continuous Skolem Problem corresponds to a time-bounded version of the hyperplane reachability problem.

The *characteristic polynomial* of the differential equation (1) is

$$\chi(x) := x^n + a_{n-1}x^{n-1} + \ldots + a_0\,.$$

Let $\lambda_1, \ldots, \lambda_m$ be the distinct roots of χ. Any solution of (1) has the form $f(t) = \sum_{j=1}^{m} P_j(t)e^{\lambda_j t}$, where the P_j are polynomials with algebraic coefficients that are determined by the initial conditions of the differential equation. We call a function f in this form an *exponential polynomial*. If the roots of χ are all simple then f can be written as an exponential polynomial in which the polynomials P_j are all constant.

The Continuous Skolem Problem can equivalently be formulated in terms of whether an exponential polynomial has a zero in a given interval of reals. If the characteristic roots have the form $\lambda_j = r_j + i\omega_j$, where $r_j, \omega_j \in \mathbb{R}$, then we can also write $f(t)$ in the form $f(t) = \sum_{j=1}^{m} e^{r_j t}(Q_{1,j}(t)\sin(\omega_j t) + Q_{2,j}(t)\cos(\omega_j t))$, where the polynomials $Q_{1,j}, Q_{2,j}$ have real algebraic coefficients. We call $\omega_1, \ldots, \omega_m$ the *frequencies* of f.

Our first result is to show decidability of the Bounded Continuous Skolem Problem subject to Schanuel's Conjecture, a unifying conjecture in transcendental number theory that plays a key role in the study of the exponential function over both the real and complex numbers [21, 22]. Intuitively, decidability of the Bounded Continuous Skolem Problem is non-trivial because an exponential polynomial can approach 0 tangentially. Assuming Schanuel's Conjecture, we show that any exponential polynomial admits a factorisation such that the zeros of each factor can be detected using finite-precision numerical computations. Our method, however, does not bound the precision required to find zeros, so we do not obtain a complexity bound for the procedure.

A celebrated paper of Macintyre and Wilkie [18] obtains decidability of the first-order theory of $\mathbb{R}_{\exp} = (\mathbb{R}, 0, 1, <, \cdot, +, \exp)$ assuming Schanuel's Conjecture over $\mathbb{R}$. The proof of [17, Theorem 3.1] mentions an unpublished result of Macintyre and Wilkie that generalises [18] to obtain decidability when $\mathbb{R}_{\exp}$ is augmented with the restricted functions $\sin\restriction_{[0,2\pi]}$ and $\cos\restriction_{[0,2\pi]}$, this time assuming Schanuel's Conjecture over $\mathbb{C}$. This result immediately implies

(conditional) decidability of the Bounded Continuous Skolem Problem. However, decidability of the latter problem is simpler and, as we show below, can be established more directly.

In the unbounded case, we analyse exponential polynomials in terms of the number of rationally linearly independent frequencies. We show that the unbounded problem can be reduced to the bounded problem if there is at most one rationally linearly independent frequency, or if there are two rationally linearly independent frequencies and all characteristic roots are simple. These two reductions are unconditional and rely on the cell decomposition theorem for semi-algebraic sets [3] and Baker's Theorem on linear forms in logarithms of algebraic numbers [2].

In the full version on this paper [7] we complete the picture by showing that decidability of the unbounded problem in the case of two (or more) rationally linearly independent frequencies would entail a major new effectiveness result in Diophantine approximation – namely computability of the Diophantine-approximation types of all real algebraic numbers. As we discuss in [7], currently essentially nothing is known about Diophantine-approximation types of algebraic numbers of degree three or higher, and they are the subject of several longstanding open problems.

The question of deciding whether an exponential polynomial f has infinitely many zeros is investigated in [8]. There the problem is shown to be decidable if f satisfies a differential equation of order at most 7. This result does not rely on Schanuel's Conjecture. It is also shown in [8] that, as with the Continuous Skolem Problem, decidability of the Infinite Zeros Problem in the general case would entail significant new effectiveness results in Diophantine approximation.

2 Mathematical Background

2.1 Zero Finding

Let $f : [a, b] \to \mathbb{R}$ be a function defined on a closed interval of reals with endpoints $a, b \in \mathbb{Q}$. Suppose the following two conditions hold: (i) there exists $M > 0$ such that f is M-Lipschitz, i.e., $|f(s) - f(t)| \leq M|s - t|$ for all $s, t \in [a, b]$; (ii) given $t \in [a, b] \cap \mathbb{Q}$ and positive error bound $\varepsilon \in \mathbb{Q}$, we can compute $q \in \mathbb{Q}$ such that $|f(t) - q| < \varepsilon$. Then given a positive rational number δ we can compute piecewise linear functions $f_\delta^+, f_\delta^- : [a, b] \to \mathbb{R}$ such that $f_\delta^-(t) \leq f(t) \leq f_\delta^+(t)$ and $f_\delta^+(t) - f_\delta^-(t) \leq \delta$ for all $t \in [a, b]$. We do this as follows:

1. Pick $N \in \mathbb{N}$ such that $\frac{1}{N} < \frac{\delta}{4(b-a)M}$ and consider sample points $s_j := a + \frac{(b-a)j}{N}$, $j = 0, \ldots, N$, dividing the interval $[a, b]$ into N sub-intervals, each of length at most $\frac{\delta}{4M}$.
2. For each sample point s_j compute $q_j \in \mathbb{Q}$ such that $|q_j - f(s_j)| < \frac{\delta}{4}$, define $f_\delta^-(s_j) = q_j - \frac{\delta}{2}$, $f_\delta^+(s_j) = q_j + \frac{\delta}{2}$, and extend f_δ^- and f_δ^+ linearly between sample points.

Note that the Lipschitz condition on f ensures that $f_\delta^- \leq f \leq f_\delta^+$.

Now suppose that f satisfies the following additional conditions: (iii) $f(a) \neq 0, f(b) \neq 0$; (iv) for any $t \in (a, b)$ such that $f(t) = 0$, $f'(t)$ exists and is non-zero, i.e., f has no tangential zeros. Then we can decide the existence of a zero of f by computing upper and lower approximations f_δ^+ and f_δ^- for successively smaller values of δ. If $f_\delta^+(t) < 0$ for all t or $f_\delta^-(t) > 0$ for all t then we conclude that f has no zero on $[a, b]$; if $f_\delta^+(s) < 0$ and $f_\delta^-(t) > 0$ for some s, t then we conclude that f has a zero; otherwise we proceed to a smaller value of δ. This procedure terminates since by (iii) and (iv) either f has a zero in $[a, b]$ or it is bounded away from zero.

2.2 Number-Theoretic Algorithms

For the purposes of establishing decidability, we can assume that an instance of the Continuous Skolem Problem is a real-valued exponential polynomial $f(t) = \sum_{j=1}^{m} P_j(t)e^{\lambda_j t}$, where $\lambda_1, \ldots, \lambda_m$ and the coefficients of the polynomials $P_1, \ldots, P_m$ are algebraic, see [4, Theorem 6].

For computational purposes we represent an algebraic number α by a polynomial P with rational coefficients such that $P(\alpha) = 0$, together with a numerical approximation $p + qi$, where $p, q \in \mathbb{Q}$, of sufficient accuracy to distinguish α from the other roots of P [9, Section 4.2.1]. Given this representation we can obtain numerical approximations of α with arbitrary precision [19]

Let K be the extension field of $\mathbb{Q}$ generated by $\lambda_1, \ldots, \lambda_m$ and the coefficients of the polynomials $P_1, \ldots, P_m$. Note that K is closed under complex conjugation. We can compute a primitive element of K, that is, an algebraic number θ such that $K = \mathbb{Q}(\theta)$, together with a representation of each characteristic root λ_j as a polynomial in θ with rational coefficients (see [9, Section 4.5]). From the representation of $\lambda_1, \ldots, \lambda_m$ as elements of $\mathbb{Q}(\theta)$, it is straightforward to determine maximal $\mathbb{Q}$-linearly independent subsets of $\{\mathrm{Re}(\lambda_j) : 1 \leq j \leq m\}$ and $\{\mathrm{Im}(\lambda_j) : 1 \leq j \leq m\}$ (see [14, Section 1]).

Let log denote the branch of the complex logarithm defined by $\log(re^{i\theta}) = \log(r) + i\theta$ for a positive real number r and $0 \leq \theta < 2\pi$. Recall that one can compute $\log z$ and e^z to within arbitrarily small additive error given a sufficiently precise approximation of z [6].

2.3 Laurent Polynomials

Let K be a sub-field of $\mathbb{C}$ that has finite dimension over $\mathbb{Q}$ and is closed under complex conjugation. Fix non-negative integers r and s, and consider a single variable x and tuples of variables $\boldsymbol{y} = \langle y_1, \ldots, y_r \rangle$ and $\boldsymbol{z} = \langle z_1, \ldots, z_s \rangle$. Consider the ring of Laurent polynomials

$$\mathcal{R} := K[x, y_1, y_1^{-1}, \ldots, y_r, y_r^{-1}, z_1, z_1^{-1}, \ldots, z_s, z_s^{-1}]\,,$$

which can be seen as a localisation[1] of the polynomial ring $\mathcal{A} := K[x, y_1, \ldots, y_r, z_1, \ldots, z_s]$ in the multiplicative set generated by the set of variables $\{y_1, \ldots, y_r\} \cup \{z_1, \ldots, z_s\}$. The multiplicative units of $\mathcal{R}$ are the non-zero monomials in variables $y_1, \ldots, y_r$ and $z_1, \ldots, z_s$. As the localisation of a unique factorisation domain, $\mathcal{R}$ is itself a unique factorisation domain [10, Theorem 10.3.7]. From the proof of this fact it moreover easily follows that $\mathcal{R}$ inherits from $\mathcal{A}$ computability of factorisation into irreducibles (e.g., using the algorithm of [16]).

We extend the operation of complex conjugation to a ring automorphism of $\mathcal{R}$ as follows. Given a polynomial

$$P = \sum_{j=1}^{n} a_j x^{u_j} {y_1}^{v_{j1}} \ldots {y_r}^{v_{jr}} {z_1}^{w_{j1}} \ldots {z_s}^{w_{js}}\,,$$

where $a_1, \ldots, a_n \in K$, define its conjugate to be

$$\overline{P} := \sum_{j=1}^{n} \overline{a_j} x^{u_j} {y_1}^{v_{j1}} \ldots {y_r}^{v_{jr}} {z_1}^{-w_{j1}} \ldots {z_s}^{-w_{js}}\,.$$

[1] Recall that the *localisation* of a commutative ring $\mathcal{U}$ in a multiplicatively closed subset S such that $0_{\mathcal{U}} \notin S$ is the ring of formal fractions $\mathcal{U}_S = \{a/s : a \in \mathcal{U}, s \in S\}$, with addition and multiplication defined as usual.

This definition is motivated by thinking of the variables x and $y_1, \ldots, y_r$ as real-valued and the variables $z_1, \ldots, z_s$ as taking values in the unit circle in the complex plane.

We will need the following proposition characterising those polynomials in $P \in \mathcal{R}$ such that P and $\overline{P}$ are associates, i.e., such that $\overline{P}$ is equal to the product of P by a monomial. Here we use pointwise notation for exponentiation: given a tuple of integers $\boldsymbol{u} = \langle u_1, \ldots, u_s \rangle$, we write $\boldsymbol{z}^{\boldsymbol{u}}$ for the monomial $z_1^{u_1} \ldots z_s^{u_s}$. The proof of the proposition can be found in the full version [7].

▶ **Proposition 1.** *Let $P \in \mathcal{R}$ be such that $P = \boldsymbol{z}^{\boldsymbol{u}}\overline{P}$ for $\boldsymbol{u} \in \mathbb{Z}^s$. Then either (i) P has the form $P = \boldsymbol{z}^{\boldsymbol{u}}Q$ for some $Q \in \mathcal{R}$ with $Q = \overline{Q}$, or (ii) there exists $Q \in \mathcal{R}$ such that $P = Q + \boldsymbol{z}^{\boldsymbol{u}}\overline{Q}$ and P does not divide Q in $\mathcal{R}$.*

2.4 Transcendence Theory

We will use transcendence theory in our analysis of both the bounded and unbounded variants of the Continuous Skolem Problem. In the unbounded case we will use the following classical result.

▶ **Theorem 2** (Gelfond-Schneider). *Let a, b be algebraic numbers not equal to 0 or 1. Then for any branch of the logarithm function, $\frac{\log(b)}{\log(a)}$ is either rational or transcendental.*

In fact we will make use of the following corollary, which is obtained by applying Theorem 2 to the algebraic numbers $a = e^{i(\alpha_2 - \alpha_1)}$ and $b = e^{i(\beta_2 - \beta_1)}$.

▶ **Corollary 3.** *Let $\alpha_1 \neq \beta_1$, $\alpha_2 \neq \beta_2$ all lie in $[0, \pi]$ and suppose that $\cos(\alpha_1), \cos(\alpha_2), \cos(\beta_1)$ and $\cos(\beta_2)$ are algebraic. Then $\frac{\beta_2 - \alpha_2}{\beta_1 - \alpha_1}$ is either rational or transcendental.*

Our results in the bounded case depend on Schanuel's conjecture, a unifying conjecture in transcendental number theory [15], which, if true, greatly generalises many of the central results in the field (including the Gelfond-Schneider Theorem, above). Recall that a *transcendence basis* of a field extension L/K is a subset $S \subseteq L$ such that S is algebraically independent over K and L is algebraic over $K(S)$. All transcendence bases of L/K have the same cardinality, which is called the *transcendence degree* of the extension.

▶ **Conjecture 4** (Schanuel's Conjecture [15]). *Let $a_1, \ldots, a_n$ be complex numbers that are linearly independent over $\mathbb{Q}$. Then the field $\mathbb{Q}(a_1, \ldots, a_n, e^{a_1}, \ldots, e^{a_n})$ has transcendence degree at least n over $\mathbb{Q}$.*

A special case of Schanuel's conjecture, that is known to hold unconditionally, is the Lindemann-Weierstrass Theorem [15]: if $a_1, \ldots, a_n$ are algebraic numbers that are linearly independent over $\mathbb{Q}$, then $e^{a_1}, \ldots, e^{a_n}$ are algebraically independent.

We apply Schanuel's conjecture via the following proposition.

▶ **Proposition 5.** *Given non-negative integers r and s, let $\{a_1, \ldots, a_r\}$ and $\{b_1, \ldots, b_s\}$ be $\mathbb{Q}$-linearly independent sets of real algebraic numbers. Furthermore, let $P, Q \in \mathcal{R}$ be two polynomials that have algebraic coefficients and are coprime in $\mathcal{R}$. Then the equations*

$$P(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t}) = 0 \tag{2}$$

$$Q(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t}) = 0 \tag{3}$$

have no non-zero common solution $t \in \mathbb{R}$.

Proof. Consider a solution $t \neq 0$ of Equations (2) and (3). By passing to suitable associates, we may assume without loss of generality that P and Q lie in $\mathcal{A}$, i.e., that all variables in P and Q appear with non-negative exponent. Moreover, since P and Q are coprime in $\mathcal{R}$, their greatest common divisor R in $\mathcal{A}$ is a monomial. In particular,

$$R(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t}) \neq 0\,.$$

Thus, dividing P and Q by R, we may assume that P and Q are coprime in $\mathcal{A}$ and that Equations (2) and (3) still hold.

Since coprime univariate polynomials cannot have a common root, we may assume without loss of generality that $r + s \geq 1$. By Schanuel's conjecture, the extension

$$\mathbb{Q}(a_1 t, \ldots, a_r t, ib_1 t, \ldots, ib_s t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t})/\mathbb{Q}$$

has transcendence degree at least $r + s$. Since $a_1, \ldots, a_r$ and $b_1, \ldots, b_s$ are algebraic over $\mathbb{Q}$, writing

$$S := \langle t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t} \rangle\,,$$

it follows that the extension $\mathbb{Q}(S)/\mathbb{Q}$ also has transcendence degree at least $r + s$.

From Equations (2) and (3) we can regard S as specifying a common root of P and Q. Pick some variable $\sigma \in \{x, y_j, z_j : 1 \leq i \leq r, 1 \leq j \leq s\}$ that has positive degree in P. Then the component of S corresponding to σ is algebraic over the remaining components of S. We claim that the remaining components of S are algebraically dependent and thus S comprises at most $r + s - 1$ algebraically independent elements, contradicting Schanuel's conjecture. The claim clearly holds if σ does not appear in Q. On the other hand, if σ has positive degree in Q then, since P and Q are coprime in $\mathcal{A}$, the multivariate resultant $\mathrm{Res}_\sigma(P, Q)$ is a non-zero polynomial in the set of variables $\{x, y_j, z_j : 1 \leq i \leq r, 1 \leq j \leq s\} \setminus \{\sigma\}$ which has a root at S (see, e.g., [11, Page 163]). Thus the claim also holds in this case. In either case we obtain a contradiction to Schanuel's conjecture and we conclude that Equations (2) and (3) have no non-zero solution $t \in \mathbb{R}$. ◀

3 Decidability of the Bounded Continuous Skolem Problem

Given non-negative integers r and s, suppose that $\{a_1, \ldots, a_r\}$ and $\{ib_1, \ldots, ib_s\}$ are $\mathbb{Q}$-linearly independent sets of real and imaginary numbers respectively. Let the ring of Laurent polynomials $\mathcal{R}$ be as in Section 2.3 and consider the exponential polynomial

$$f(t) = P(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t})\,, \tag{4}$$

where $P \in \mathcal{R}$ is irreducible. We say that f is a *type-1* exponential polynomial if P and $\overline{P}$ are not associates in $\mathcal{R}$, we say that f is *type-2* if $P = \alpha\overline{P}$ for some $\alpha \in K$, and we say that f is *type-3* if $P = U\overline{P}$ for some non-constant unit $U \in \mathcal{R}$.

▶ **Example 6.** The simplest example of a type-3 exponential polynomial is $g(t) = 1 + e^{it}$. Here $g(t) = P(e^{it})$, where $P(z) = 1 + z$ is an irreducible polynomial that is associated with its conjugate $\overline{P}(z) = 1 + z^{-1}$. Note that the exponential polynomial $f(t) = 2 + 2\cos(t)$, which has infinitely many tangential zeros, factors as the product of two type-3 exponential polynomials $f(t) = g(t)\overline{g(t)}$.

In the case of a type-2 exponential polynomial $P = \alpha\overline{P}$ it is clear that we must have $|\alpha| = 1$. Moreover, by replacing P by βP, where $\beta^2 = \overline{\alpha}$, we may assume without loss of

generality that $P = \overline{P}$. Similarly, in the case of a type-3 exponential polynomial, we can assume without loss of generality that $P = \boldsymbol{z}^{\boldsymbol{u}}\overline{P}$ for some non-zero vector $\boldsymbol{u} \in \mathbb{Z}^s$.

Now consider an arbitrary exponential polynomial $f(t) := \sum_{j=1}^{m} P_j(t)e^{\lambda_j t}$. Assume that the coefficient field K of $\mathcal{R}$ contains the coefficients of $P_1, \ldots, P_m$. Let $\{a_1, \ldots, a_r\}$ be a basis of the $\mathbb{Q}$-vector space spanned by $\{\mathrm{Re}(\lambda_j) : 1 \leq j \leq m\}$ and let $\{b_1, \ldots, b_s\}$ be a basis of the the $\mathbb{Q}$-vector space spanned by $\{\mathrm{Im}(\lambda_j) : 1 \leq j \leq m\}$. Without loss of generality we may assume that each characteristic root λ is an integer linear combination of $a_1, \ldots, a_r$ and $ib_1, \ldots, ib_s$. Then $e^{\lambda t}$ is a product of positive and negative powers of $e^{a_1 t}, \ldots, e^{a_r t}$ and $e^{ib_1 t}, \ldots, e^{ib_s t}$, and hence there is a Laurent polynomial $P \in \mathcal{R}$ such that

$$f(t) = P(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t})\,. \tag{5}$$

Since P can be written as a product of irreducible factors in $\mathcal{R}$, it follows that f can be written as product of type-1, type-2, and type-3 exponential polynomials, and moreover this factorisation can be computed from f. Thus it suffices to show how to decide the existence of zeros of these three special forms of exponential polynomial. We will handle all three cases using Schanuel's conjecture.

Writing the exponential polynomial $f(t)$ in (5) in the form $f(t) = \sum_{j=1}^{m} Q_j(t)e^{\lambda_j t}$, it follows from the irreducibility of P that the polynomials $Q_1, \ldots, Q_m$ have no common root. But then by the Lindemann-Weierstrass Theorem any zero of f must be transcendental (see [4, Theorem 8]).

▶ **Theorem 7.** *The Bounded Continuous Skolem Problem is decidable subject to Schanuel's conjecture.*

Proof. Consider an exponential polynomial

$$f(t) = P(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t})\,, \tag{6}$$

where $P \in \mathcal{R}$ is irreducible. Suppose that $\{a_1, \ldots, a_r\}$ and $\{ib_1, \ldots, ib_s\}$ are $\mathbb{Q}$-linearly independent sets of, respectively, real and imaginary numbers lying in the coefficient field K of $\mathcal{R}$. We show how to decide whether f has a zero in a bounded interval $I \subseteq \mathbb{R}_{\geq 0}$, considering separately the case of type-1, type-2, and type-3 exponential polynomials.

Case (i): f is a type-1 exponential polynomial

Note that P and $\overline{P}$ are coprime in $\mathcal{R}$ since, by assumption, they are both irreducible and are not associates. We claim that in this case the equation $f(t) = 0$ has no solution $t \in \mathbb{R}$. Indeed $f(t) = 0$ implies

$$\begin{aligned} P(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t}) &= 0 \\ \overline{P}(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t}) &= 0\,, \end{aligned}$$

and the non-existence of a zero of f follows immediately from Proposition 5.

Case (ii): f is a type-2 exponential polynomial

In this case we have $P = \overline{P}$ and so f is real-valued. Our aim is to use the procedure of Section 2.1 to determine whether or not f has a zero in $[c, d]$, where $c, d \in \mathbb{Q}$. To this end, notice first that $f(c), f(d) \neq 0$ since any root of f must be transcendental. Moreover, since f' is bounded on $[c, d]$, f is Lipschitz on $[c, d]$. It remains to verify that the equations $f(t) = 0, f'(t) = 0$ have no common solution $t \in [c, d]$.

We can write $f'(t)$ in the form

$$f'(t) = Q(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t}),$$

where Q is the polynomial

$$Q = \frac{\partial P}{\partial x} + \sum_{j=1}^{r} a_j y_j \frac{\partial P}{\partial y_j} + \sum_{j=1}^{s} i b_j z_j \frac{\partial P}{\partial z_j}.$$

We claim that P and Q are coprime in $\mathcal{R}$. Indeed, since P is irreducible, P and Q can only fail to be coprime if P divides Q.

If P has strictly positive degree k in x then Q has degree $k-1$ in x and thus P cannot divide Q. (Recall that all polynomials in $\mathcal{R}$ have non-negative degree in the variable x.) On the other hand, if P has degree 0 in x then Q is obtained from P by multiplying each monomial $\boldsymbol{y}^{\boldsymbol{u}}\boldsymbol{z}^{\boldsymbol{v}}$ appearing in P by the complex-number constant $\sum_{j=1}^{r} a_j u_j + i \sum_{j=1}^{s} b_j v_j$. Moreover, by the assumption of linear independence of $\{a_1, \ldots, a_r\}$ and $\{b_1, \ldots, b_s\}$, each monomial in P is multiplied by a different constant. Since P is not a unit, it has at least two different monomials and so P is not a constant multiple of Q. Furthermore, for each variable $\sigma \in \{y_j, y_j^{-1} : 1 \leq j \leq r\} \cup \{z_j, z_j^{-1} : 1 \leq j \leq s\}$, its degree in P is equal to its degree in Q. Thus P cannot be a multiple of Q by a non-constant polynomial either.

We conclude that P does not divide Q and hence P and Q are coprime. It now follows from Proposition 5 that the equations $f(t) = f'(t) = 0$ have no solution $t \in \mathbb{R}$.

Case (iii): f is a type-3 exponential polynomial

Suppose that f is a type-3 exponential polynomial. Then in (6) we have that $P = \boldsymbol{z}^{\boldsymbol{u}}\overline{P}$ for some non-zero vector $\boldsymbol{u} \in \mathbb{Z}^s$. By Proposition 1 we can write $P = Q + \boldsymbol{z}^{\boldsymbol{u}}\overline{Q}$ for some polynomial $Q \in \mathcal{R}$ that is coprime with P.

Now define

$$g_1(t) := Q(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t})$$

and $g_2(t) := e^{ib_1 u_1} \cdots e^{ib_s u_s}\overline{g_1(t)}$, so that $f(t) = g_1(t) + g_2(t)$ for all t.

We show that $g_2(t) \neq 0$ for all $t \in \mathbb{R}$. Indeed if $g_2(t) = 0$ for some t then we also have $g_1(t) = 0$ and hence $f(t) = 0$. For such a t it follows that

$$\begin{aligned} P(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t}) &= 0 \\ Q(t, e^{a_1 t}, \ldots, e^{a_r t}, e^{ib_1 t}, \ldots, e^{ib_s t}) &= 0. \end{aligned}$$

But P and Q are coprime and so these two equations cannot both hold by Proposition 5. Not only do we have $g_2(t) \neq 0$ for all $t \in \mathbb{R}$, but, applying the sampling procedure in Section 2.1 we can compute a strictly positive lower bound on $|g_2(t)|$ over the interval $[c, d]$.

Since $g_2(t) \neq 0$ for all $t \in \mathbb{R}$ we may define the function $h : [c, d] \to \mathbb{R}$ by

$$h(t) := \pi + i \log\left(\frac{g_1(t)}{g_2(t)}\right).$$

Notice that $h(t) = 0$ if and only if $f(t) = 0$. Our aim is to use the procedure of Section 2.1 to decide the existence of a zero of h in the interval $[c, d]$, and thus decide whether f has a zero in $[c, d]$.

Let $t \in (c,d)$ be such that $h(t) = 0$. Then $g_1(t) = -g_2(t)$ and so $\frac{g_1(t)}{g_2(t)} = -1$ does not lie on the branch cut of the logarithm function. It follows that h is differentiable at t and

$$\begin{aligned} h'(t) = 0 \quad &\text{iff} \quad \frac{g_2(t)}{g_1(t)} \frac{g_1'(t)g_2(t) - g_2'(t)g_1(t)}{g_2(t)^2} = 0 \\ &\text{iff} \quad g_1'(t)g_2(t) - g_2'(t)g_1(t) = 0 \quad (\text{since } |g_1(t)| = |g_2(t)| \neq 0) \\ &\text{iff} \quad g_1'(t)g_2(t) + g_2'(t)g_2(t) = 0 \quad (\text{since } g_1(t) = -g_2(t)) \\ &\text{iff} \quad g_1'(t) + g_2'(t) = 0 \\ &\text{iff} \quad f'(t) = 0\,. \end{aligned}$$

Thus $h(t) = h'(t) = 0$ implies $f(t) = f'(t) = 0$. But the proof in Case (ii) shows that $f(t) = f'(t) = 0$ is impossible. (Nothing in that argument hinges on f being real-valued.) Thus h has no tangential zeros in (c,d).

We cannot directly use the procedure in Section 2.1 to decide whether h has a zero in $[c,d]$ since h is not necessarily continuous: its value can jump from $-\pi$ to π (or *vice versa*) due to the branch cut of the logarithm along the positive real axis. However, due to the strictly positive lower bound on $|g_2(t)|$, the function $|h|$ *is* Lipschitz on $[c,d]$. Thus, applying the sampling procedure in Section 2.1 for computing lower and upper bounds of Lipschitz functions we can compute a set $E \subseteq [c,d]$ such that E is a finite union of intervals with rational endpoints, $|f(t)| \leq \frac{2\pi}{3}$ for $t \in E$, and $|f(t)| \geq \frac{\pi}{3}$ for $t \notin E$. In particular, E contains all zeros of f in $[c,d]$ and f is Lipschitz on E. Thus we can apply the zero-finding procedure from Section 2.1 to the restriction $h \restriction E$ and thereby decide whether h has a zero on $[c,d]$. ◀

4 The Unbounded Case

In this section we consider the unbounded case of the Continuous Skolem Problem. For our analysis it is convenient to present exponential polynomials in the form

$$f(t) = \sum_{j=1}^{n} e^{r_j t} \left(P_{1,j}(t)\cos(\omega_j t) + P_{2,j}(t)\sin(\omega_j t)\right)\,, \tag{7}$$

where r_j, ω_j are real algebraic numbers and $P_{1,j}, P_{2,j}$ are polynomials with real algebraic coefficients for $j = 1, \ldots, n$. Our aim is to classify the difficulty of the problem in terms of the number of rationally linear independent frequencies $\omega_1, \ldots, \omega_n$.

Recall that in Section 3 we have shown the bounded problem to be decidable subject to Schanuel's Conjecture. In the full version of this paper [7] we give a reduction of the unbounded problem to the bounded problem in case the set of frequencies spans a one-dimensional vector space over $\mathbb{Q}$. In the present section we give a reduction of the unbounded problem to the bounded problem in case the set of frequencies spans a two-dimensional vector space over $\mathbb{Q}$ and the polynomials $P_{1,j}$ and $P_{2,j}$ are all constant. (This last condition is equivalent to the assumption that $f(t)$ is simple.) The argument in the two-dimensional case is a more sophisticated version of that in the one-dimensional case, although the result is not more general due the assumption of simplicity.

In the full version [7] we present a family of instances showing that obtaining decidability of the unbounded problem in the two-dimensional case without the assumption of simplicity would require much finer Diophantine-approximation bounds than are currently known.

4.1 Background on Semi-Algebraic Sets

A subset of $\mathbb{R}^n$ is *semi-algebraic* if it is defined by a Boolean combination of constraints of the form $P(x_1, \ldots, x_n) > 0$, where P is a polynomial with real algebraic coefficients. A partial function $f : \mathbb{R}^n \to \mathbb{R}$ is semi-algebraic if its graph is a semi-algebraic subset of $\mathbb{R}^{n+1}$. The Tarski-Seidenberg Theorem [5, Section 1] states that the semi-algebraic sets are closed under projection and are therefore precisely the first-order definable sets over the structure $(\mathbb{R}, <, +, \cdot, 0, 1)$.

Let $(i_1, \ldots, i_n)$ be a sequence of zeros and ones of length $n \geq 1$. An $(i_1, \ldots, i_n)$-*cell* is a subset of $\mathbb{R}^n$, defined by induction on n as follows:

(i) A (0)-cell is a singleton subset of $\mathbb{R}$ and a (1)-cell is an open interval $(a, b) \subseteq \mathbb{R}$.

(ii) Let $X \subseteq \mathbb{R}^n$ be a $(i_1, \ldots, i_n)$-cell and $f : X \to \mathbb{R}$ a continuous semi-algebraic function. Then $\{(\boldsymbol{x}, f(\boldsymbol{x})) \in \mathbb{R}^{n+1} : \boldsymbol{x} \in X\}$ is a $(i_1, \ldots, i_n, 0)$-cell, while $\{(\boldsymbol{x}, y) \in \mathbb{R}^{n+1} : \boldsymbol{x} \in X \wedge y < f(\boldsymbol{x})\}$ and $\{(\boldsymbol{x}, y) \in \mathbb{R}^{n+1} : \boldsymbol{x} \in X \wedge y > f(\boldsymbol{x})\}$ are both $(i_1, \ldots, i_n, 1)$-cells.

(iii) Let $X \subseteq \mathbb{R}^n$ be a $(i_1, \ldots, i_n)$-cell and $f, g : X \to \mathbb{R}$ continuous semi-algebraic functions such that $f(\boldsymbol{x}) < g(\boldsymbol{x})$ for all $\boldsymbol{x} \in X$. Then $\{(\boldsymbol{x}, y) \in \mathbb{R}^{n+1} : f(\boldsymbol{x}) < y < g(\boldsymbol{x})\}$ is a $(i_1, \ldots, i_n, 1)$-cell.

A cell in $\mathbb{R}^n$ is a $(i_1, \ldots, i_n)$-cell for some (necessarily unique) sequence $(i_1, \ldots, i_n)$.

A fundamental result about semi-algebraic sets, that we will use below, is the Cell-Decomposition Theorem [3]: given a semi-algebraic set $E \subseteq \mathbb{R}^n$ one can compute a partition of E as a disjoint union of cells $E = C_1 \cup \ldots \cup C_m$.

We will also need the following result, proved in [7].

▶ **Lemma 8.** *Let $D \subseteq \mathbb{R}^n$ be a semi-algebraic set, $g : D \to \mathbb{R}$ a bounded semi-algebraic function, and $r_1, \ldots, r_n$ real algebraic numbers. Define $S = \{t \in \mathbb{R}_{\geq 0} : (e^{r_1 t}, \ldots, e^{r_n t}) \in D\}$. Then*

1. *It is decidable whether or not S is bounded. If S is bounded then we can compute $T_0 \in \mathbb{N}$ such that $S \subseteq [0, T_0]$ and if S is unbounded then we can compute $T_0 \in \mathbb{N}$ such that $(T_0, \infty) \subseteq S$.*
2. *If S is unbounded then the limit $g^* = \lim_{t \to \infty} g(e^{r_1 t}, \ldots, e^{r_n t})$ exists, is an algebraic number, and there are effective constants $T_1, \varepsilon > 0$ such that $|g(e^{r_1 t}, \ldots, e^{r_n t}) - g^*| < e^{-\varepsilon t}$ for all $t > T_1$.*

4.2 Two Linearly Independent Frequencies

The following lemma, which is a reformulation of [4, Lemma 13], plays an instrumental role in this section. The lemma itself relies on a powerful quantitative result in transcendence theory – Baker's Theorem on linear forms in logarithms of algebraic numbers [2].

▶ **Lemma 9.** *Let b_1, b_2 be real algebraic numbers, linearly independent over $\mathbb{Q}$. Furthermore, let φ_1, φ_2 be real numbers such that $e^{i\varphi_1}$ and $e^{i\varphi_2}$ are algebraic. Then there exist effectively computable constants $N, T > 0$ such that for all $t \geq T$ and all $k_1, k_2 \in \mathbb{Z}$, at least one of $|b_1 t - \varphi_1 - 2k_1\pi| > 1/t^N$ and $|b_2 t - \varphi_2 - 2k_2\pi| > 1/t^N$ holds.*

The main result of the section is the following.

▶ **Theorem 10.** *Let $f(t) = \sum_{j=1}^{n} e^{r_j t} \left(a_{1,j} \cos(\omega_j t) + a_{2,j} \sin(\omega_j t)\right)$ be an exponential polynomial where $r_j, a_{1,j}, a_{2,j}, \omega_j$ are real algebraic numbers and the $\mathbb{Q}$-span of $\{\omega_1, \ldots, \omega_n\}$ has dimension two as a $\mathbb{Q}$-vector space. Then we can decide whether or not $\{t \in \mathbb{R}_{\geq 0} : f(t) = 0\}$ is bounded and, if bounded, we can compute an integer T such that $\{t \in \mathbb{R}_{\geq 0} : f(t) = 0\} \subseteq [0, T]$.*

Proof. Let b_1, b_2 be real algebraic numbers, linearly independent over $\mathbb{Q}$, such that ω_j is an integer linear combination of b_1 and b_2 for $j = 1, \dots, n$. For each $n \in \mathbb{Z}$, $\sin(nb_1t)$ and $\cos(nb_1t)$ can be written as polynomials in $\sin(b_1t)$ and $\cos(b_1t)$ with integer coefficients, and similarly for b_2. It follows that we can write f in the form

$$f(t) = Q(e^{r_1 t}, \dots, e^{r_n t}, \cos(b_1 t), \sin(b_1 t), \cos(b_2 t), \sin(b_2 t))$$

for some polynomial Q with real algebraic coefficients that is computable from f.

Write $R_{++} = \{t \geq 0 : \sin(b_1t) \geq 0 \wedge \sin(b_2t) \geq 0\}$, $R_{+-} = \{t \geq 0 : \sin(b_1t) \geq 0 \wedge \sin(b_2t) \leq 0\}$, and likewise define R_{-+}, R_{--} for the two remaining sign conditions on $\sin(b_1t)$ and $\sin(b_2t)$. We show how to decide boundedness of $\{t \in R_{++} : f(t) = 0\}$. (The cases for R_{+-}, R_{-+}, and R_{--} follow *mutatis mutandis*.) The idea is to compute a partition of $\{t \in R_{++} : f(t) = 0\}$ into components $Z_1, \dots, Z_m$ and to separately decide boundedness of each component Z_j.

Define a semi-algebraic set

$$E = \left\{(\boldsymbol{u}, x_1, x_2) \in \mathbb{R}^{n+2} : \exists y_1, y_2 \geq 0 \left(x_1^2 + y_1^2 = x_2^2 + y_2^2 = 1 \wedge Q(\boldsymbol{u}, x_1, y_1, x_2, y_2) = 0\right)\right\}.$$

Then for $t \in R_{++}$ we have $f(t) = 0$ if and only if $(e^{\boldsymbol{r}t}, \cos(b_1t), \cos(b_2t)) \in E$, where $\boldsymbol{r} = (r_1, \dots, r_n)$. Now consider a cell decomposition $E = C_1 \cup \dots \cup C_m$ for cells $C_1, \dots, C_m \subseteq \mathbb{R}^{n+2}$, and define

$$Z_j = \{t \in R_{++} : (e^{\boldsymbol{r}t}, \cos(b_1t), \cos(b_2t)) \in C_j\}, \qquad j = 1, \dots, m, \tag{8}$$

Then $\{t \in R_{++} : f(t) = 0\} = Z_1 \cup \dots \cup Z_m$.

Now fix $j \in \{1, \dots, m\}$. We show how to decide boundedness of Z_j. To this end, write $D_j \subseteq \mathbb{R}^n$ for the projection of the corresponding cell $C_j \subseteq \mathbb{R}^{n+2}$ on the first n coordinates.

First suppose that $\{t \in \mathbb{R} : e^{\boldsymbol{r}t} \in D_j\}$ is bounded. Then by Lemma 8 we can compute an upper bound T of this set. But $Z_j \subseteq \{t \in \mathbb{R}_{\geq 0} : e^{\boldsymbol{r}t} \in D_j\}$ and so $Z_j \subseteq [0, T]$.

On the other hand, suppose that $\{t \in \mathbb{R} : e^{\boldsymbol{r}t} \in D_j\}$ is unbounded. Then, by Lemma 8, this set contains an unbounded interval (T, ∞) for some $T \in \mathbb{N}$. Write $I = [-1, 1]$ and define functions $g_1, g_2, h_1, h_2 : D_j \to \mathbb{R}$ by

$$g_1(\boldsymbol{u}) = \inf\{x \in I : \exists y\, (\boldsymbol{u}, x, y) \in C_j\} \qquad g_2(\boldsymbol{u}) = \inf\{y \in I : \exists x\, (\boldsymbol{u}, x, y) \in C_j\} \tag{9}$$

$$h_1(\boldsymbol{u}) = \sup\{x \in I : \exists y\, (\boldsymbol{u}, x, y) \in C_j\} \qquad h_2(\boldsymbol{u}) = \sup\{y \in I : \exists x\, (\boldsymbol{u}, x, y) \in C_j\} \tag{10}$$

These functions are all semi-algebraic by quantifier elimination. Hence by Lemma 8 the limits $g_i^* = \lim_{t\to\infty} g_i(e^{\boldsymbol{r}t})$ and $h_i^* = \lim_{t\to\infty} h_i(e^{\boldsymbol{r}t})$ exist for $i = 1, 2$ and are algebraic numbers. Clearly we have $g_1^* \leq h_1^*$ and $g_2^* \leq h_2^*$. We now consider three cases according to the strictness of these inequalities.

Case I: $g_1^* = h_1^*$ and $g_2^* = h_2^*$

We show that Z_j is bounded and that we can compute T_2 such that $Z_j \subseteq [0, T_2]$.

By Lemma 8 there exist $T_1, \varepsilon > 0$ such that for all $t > T_1$ and $i = 1, 2$,

$$|g_i(e^{\boldsymbol{r}t}) - g_i^*| < e^{-\varepsilon t} \text{ and } |h_i(e^{\boldsymbol{r}t}) - h_i^*| < e^{-\varepsilon t}. \tag{11}$$

Then for $t \in R_{++}$ such that $t > T_1$ we have

$$\begin{aligned} t \in Z_j &\iff \left(e^{\boldsymbol{r}t}, \cos(b_1t), \cos(b_2t)\right) \in C_j \quad \text{(by (8))} \\ &\implies g_1(e^{\boldsymbol{r}t}) \leq \cos(b_1t) \leq h_1(e^{\boldsymbol{r}t}) \text{ and } g_2(e^{\boldsymbol{r}t}) \leq \cos(b_2t) \leq h_2(e^{\boldsymbol{r}t}) \quad \text{(by (9)(10))} \\ &\implies |\cos(b_1t) - g_1^*| < e^{-\varepsilon t} \text{ and } |\cos(b_2t) - g_2^*| < e^{-\varepsilon t} \quad \text{(by (11))} \end{aligned} \tag{12}$$

Write $g_1^* = \cos(\varphi_1)$ and $g_2^* = \cos(\varphi_2)$ for some $\varphi_1, \varphi_2 \in [0, \pi]$. Since $|\cos(\varphi_1 + x) - \cos(\varphi_1)| \geq x^3/3$ for all x sufficiently small (by a Taylor expansion), the inequality (12) implies that for some $k_1, k_2 \in \mathbb{Z}$,

$$|b_1 t - \varphi_1 - 2k_1\pi| < 3e^{-\varepsilon t/3} \text{ and } |b_2 t - \varphi_2 - 2k_2\pi| < 3e^{-\varepsilon t/3}\,. \tag{13}$$

Combining the upper bounds in (13) with the polynomial lower bounds $|b_1 t - \varphi_1 - 2k_1\pi| > 1/t^N$ and $|b_2 t - \varphi_2 - 2k_2\pi| > 1/t^N$ from Lemma 9 we obtain an effective bound T_2 for which $t \in Z_j$ implies $t < T_2$.

Case II: $g_1^* < h_1^*$

In this case we show that Z_j is unbounded. The geometric intuition is as follows. We imagine a particle in the plane whose position at time t is $(\cos(b_1 t), \cos(b_2 t))$, together with a "moving target" whose extent at time t is $\Gamma_t = \{(x, y) : (e^{\boldsymbol{r}t}, x, y) \in C_j\}$. Below we essentially argue that such a particle is bound to hit Γ_t at some time t since its orbit is dense in $[-1, +1]^2$ and Γ_t has positive dimension in the limit.

Proceeding formally, first notice that C_j cannot be a $(\ldots, 0, 1)$-cell or a $(\ldots, 0, 0)$-cell, for then we would have $g_1(\boldsymbol{u}) = h_1(\boldsymbol{u})$ for all $\boldsymbol{u} \in D_j$ and hence $g_1^* = h_1^*$. Thus C_j must either be a $(\ldots, 1, 0)$-cell or a $(\ldots, 1, 1)$-cell. In either case, C_j includes a cell of the form $\{(\boldsymbol{u}, x, \xi(\boldsymbol{u}, x)) : \boldsymbol{u} \in D, g_1(\boldsymbol{u}) < x < h_1(\boldsymbol{u})\}$ for some semi-algebraic function ξ.

Let c, d be real algebraic numbers such that $g_1^* < c < d < h_1^*$. Write $c = \cos(\psi')$ and $d = \cos(\psi)$ for $0 \leq \psi < \psi' \leq \pi$. By Lemma 8 the limits $\lim_{t\to\infty} \xi(e^{\boldsymbol{r}t}, c)$ and $\lim_{t\to\infty} \xi(e^{\boldsymbol{r}t}, d)$ exist and are algebraic numbers in the interval $[-1, 1]$. Let $\theta, \theta' \in [0, \pi]$ be such that $\cos(\theta) = \lim_{t\to\infty} \xi(e^{\boldsymbol{r}t}, d)$ and $\cos(\theta') = \lim_{t\to\infty} \xi(e^{\boldsymbol{r}t}, c)$.

By Corollary 3 we know that $\frac{\theta'-\theta}{\psi'-\psi}$ is either rational or transcendental. In particular we know that it is not equal to $\frac{b_2}{b_1}$, which is algebraic and irrational. Let us suppose that $\frac{\theta'-\theta}{\psi'-\psi} > \frac{b_2}{b_1}$ (the converse case is almost identical). Then there exists θ'' with $\theta < \theta'' < \theta'$, such that

$$\theta < \theta'' + \frac{b_2}{b_1}(\psi' - \psi) < \theta'\,. \tag{14}$$

Since $2\pi, b_1, b_2$ are linearly independent over $\mathbb{Q}$ it follows from Kronecker's approximation theorem that $\{(b_1 t, b_2 t) \bmod 2\pi : t \in \mathbb{R}_{\geq 0}\}$ is dense in $[0, 2\pi)^2$ (see [13, Chapter 23]). Thus there is an increasing sequence $t_1 < t_2 < \ldots$, with $b_1 t_n \equiv \psi \bmod 2\pi$ for all n, such that $b_2 t_n \bmod 2\pi$ converges to θ''. Then, defining $s_1 < s_2 < \ldots$ by $s_n = t_n + \frac{\psi'-\psi}{b_1}$, we have $b_1 s_n \equiv \psi' \bmod 2\pi$ for all n and, by (14),

$$\lim_{n\to\infty} b_2 s_n = \lim_{n\to\infty} b_2 t_n + \frac{b_2}{b_1}(\psi' - \psi) = \theta'' + \frac{b_2}{b_1}(\psi' - \psi) < \theta' \pmod{2\pi}\,.$$

Let $\eta(t) = \xi(e^{\boldsymbol{r}t}, \cos(b_1 t)) - \cos(b_2 t)$. Then for $t \in R_{++}$ such that $g(e^{\boldsymbol{r}t}) < \cos(b_1 t) < h(e^{\boldsymbol{r}t})$,

$$\begin{aligned} \eta(t) = 0 &\implies \cos(b_2 t) = \xi(e^{\boldsymbol{r}t}, \cos(b_1 t)) \\ &\implies (e^{\boldsymbol{r}t}, \cos(b_1 t), \cos(b_2 t)) \in C_j \\ &\implies t \in Z_j \text{ (by (8))}\,. \end{aligned}$$

Now $\lim_{n\to\infty} \eta(t_n) = \cos(\theta) - \cos(\theta'') > 0$ and $\lim_{n\to\infty} \eta(s_n) < \cos(\theta') - \cos(\theta') = 0$. Moreover for n sufficiently large we have $[t_n, s_n] \subseteq R_{++}$. It follows that $\eta(t)$ has a zero in every interval $[t_n, s_n]$ for n large enough. We conclude that Z_j is unbounded.

Case III: $g_2^* < h_2^*$

This case is symmetric to Case II and we omit details. ◀

Acknowledgements. The authors wish to thank Angus Macintyre for helpful comments.

References

1 Rajeev Alur. *Principles of Cyber-Physical Systems*. MIT Press, 2015.
2 Alan Baker. *Transcendental number theory*. Cambridge University Press, Cambridge, 1975.
3 Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer-Verlag, 2006.
4 Paul C. Bell, Jean-Charles Delvenne, Raphaël M. Jungers, and Vincent D. Blondel. The Continuous Skolem-Pisot Problem. *Theoretical Computer Science (TCS)*, 411(40-42):3625–3634, 2010.
5 Edward Bierstone and Pierre D. Milman. Semianalytic and subanalytic sets. *Publications Mathématiques de l'Institut des Hautes Études Scientifiques*, 67(1):5–42, 1988.
6 Richard P. Brent. Fast multiple-precision evaluation of elementary functions. *Journal of the ACM (JACM)*, 23(2):242–251, 1976.
7 Ventsislav Chonev, Joël Ouaknine, and James Worrell. On the skolem problem for continuous linear dynamical systems. *CoRR*, abs/1506.00695, 2015.
8 Ventsislav Chonev, Joël Ouaknine, and James Worrell. On recurrent reachability for continuous linear dynamical systems. *Logic in Computer Science (LICS)*, 2016.
9 Henri Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, 1993.
10 Paul M. Cohn. *Basic Algebra: Groups, Rings and Fields*. Springer, 2002.
11 David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2007.
12 Vesa Halava, Tero Harju, Mika Hirvensalo, and Juhani Karhumäki. Skolem's Problem – on the Border between Decidability and Undecidability. *TUCS Technical Reports*, 683, 2005.
13 Godfrey H. Hardy and Edward M. Wright. *An Introduction to the Theory of Numbers*, volume 1. Oxford, 1999.
14 Bettina Just. Integer relations among algebraic numbers. In *Mathematical Foundations of Computer Science (MFCS)*, volume 379, pages 314–320. Springer, 1989.
15 Serge Lang. *Introduction to Transcendental Numbers*. Reading, Mass., 1966.
16 Arjen K. Lenstra. Factoring multivariate polynomials over algebraic number fields. *SIAM Journal on Computing*, 16(3):591–598, 1987.
17 Angus Macintyre. Turing meets Schanuel. Preprint, to appear in the proceedings of Logic Colloquium, 2012.
18 Angus Macintyre and Alex J. Wilkie. On the decidability of the real exponential field. *In (ed. Piergiorgio Odifreddi) Kreiseliana: About and Around Georg Kreisel.*, 1996.
19 Victor Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers and Mathematics with Applications*, 31(12):97–138, 1996.
20 Terence Tao. *Structure and randomness: pages from year one of a mathematical blog*. American Mathematical Society, 2008.
21 Boris Zilber. Exponential sums equations and the Schanuel conjecture. *Journal of the London Mathematical Society*, 65:27–44, 2002.
22 Boris Zilber. Pseudo-exponentiation on algebraically closed fields of characteristic zero. *Annals of Pure and Applied Logic*, 132(1):67–95, 2005.

Analysing Decisive Stochastic Processes

Nathalie Bertrand[1], Patricia Bouyer[*2], Thomas Brihaye[†3], and Pierre Carlier[‡4]

1 **Inria Rennes Bretagne Atlantique, Rennes, France**
2 **LSV, CNRS & ENS Cachan, Cachan, France**
3 **Université de Mons, Mons, Belgium**
4 **LSV, CNRS & ENS Cachan, Cachan, France; and Université de Mons, Mons, Belgium**

Abstract

In 2007, Abdulla *et al.* introduced the elegant concept of decisive Markov chain. Intuitively, decisiveness allows one to lift the good properties of finite Markov chains to infinite Markov chains. For instance, the approximate quantitative reachability problem can be solved for decisive Markov chains (enjoying reasonable effectiveness assumptions) including probabilistic lossy channel systems and probabilistic vector addition systems with states. In this paper, we extend the concept of decisiveness to more general stochastic processes. This extension is non trivial as we consider stochastic processes with a potentially continuous set of states and uncountable branching (common features of real-time stochastic processes). This allows us to obtain decidability results for both qualitative and quantitative verification problems on some classes of real-time stochastic processes, including generalized semi-Markov processes and stochastic timed automata.

1998 ACM Subject Classification D.2.4 Software/Program Verification, F.3.1 Specifying and Verifying and Reasoning about Programs, G.3 Probabilities and Statistics

Keywords and phrases Real-time stochastic processes, Decisiveness, Approximation Scheme

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.101

1 Introduction

Given its success for finite-state systems, the model checking approach to verification has been extended to various models based on automata, and including features such as time, probability and infinite data structures. Such models allow one to represent software systems more faithfully, and at the same time, they offer the possibility to consider *quantitative* verification questions. Such problems become particularly hard to solve for infinite-state systems, often requiring the development of dedicated techniques for each class of systems.

A decade ago, Abdulla *et al.* introduced the concept of decisiveness for denumerable Markov chains [2]. A Markov chain is decisive w.r.t. a set of states F if runs almost-surely reach F or a state from which F can no longer be reached. The concept of decisiveness rules out some weird behaviours in denumerable Markov chains, and lifts most good properties of finite Markov chains to infinite Markov chains. In particular, it enables the quantitative model checking of (repeated) reachability properties, by providing an approximation scheme, which is guaranteed to terminate for decisive Markov chains. Decisiveness also elegantly subsumes other concepts such as the existence of finite attractors, or coarseness [2].

* The second authors is supported by ERC project EQualIS.
† The third author is partly supported by FP7-EU project Cassting.
‡ The fourth author is supported by ERC project EQualIS.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 101; pp. 101:1–101:14

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

Dense time required for representing real-time systems, is a potential source of infinity. However, stochastic real-time systems cannot be handled by the theory of decisive Markov chains, as both the state space and the branching are in general non-denumerable. The general philosophy for models with dense time is to design an abstraction that preserves some properties of the original model, and is amenable to efficient model checking techniques. A prominent example of such abstractions is the region graph for timed automata [4]. However, abstractions often do not preserve quantitative properties, and they may be too coarse already for the evaluation of the probability of properties as simple as reachability properties.

In this paper, we generalize the concept of decisiveness to arbitrary stochastic systems, thus including the ones generated by real-time stochastic systems. While stochastic systems are often viewed operational in the model checking community (that is, one considers executions of a system), we take here a more abstract point-of-view, and consider the general mathematical model of stochastic processes.

Our first contribution is to define a notion of decisiveness for stochastic processes, generalizing the concept introduced by Abdulla *et al.* for denumerable Markov chains. This generalization is non trivial as we consider stochastic processes with a potentially continuous state space and uncountable branching, both being common features for modelling real-time stochastic processes. Moreover, in order to discriminate which verification techniques are sound, we refine the notion of decisiveness in three variants.

Our second contribution concerns the qualitative model checking of reachability and repeated reachability properties. We show that, under some decisiveness assumption, the almost-sure model checking of (repeated) reachability properties reduces to a simpler problem, namely to a reachability problem with probability 0. We advocate that this reduction simplifies the problem: in countable models, the 0-reachability amounts to the non existence of a path, in the underlying non-probabilistic system; beyond countable models, checking that a reachability property is satisfied with 0 probability amounts to exhibiting a somehow regular set of executions with positive measure.

A third contribution concerns quantitative model checking, here again for (repeated) reachability properties. Under some further decisiveness assumption, we prove that an approximation scheme, inspired from the path enumeration algorithm [17], is guaranteed to terminate. One can thus approximate, up to a desired precision, the probability of (repeated) reachability properties.

We then realize that non-Zeno real-time stochastic processes have good decisiveness properties when focusing on time-bounded reachability properties, which enables the evaluation of such properties within arbitrary precision.

Last, but not least, we introduce a generic notion of abstraction and explain how to derive decisiveness of the concrete model, using similar properties on the abstraction. We instantiate our framework with generalized semi-Markov processes (GSMP) and stochastic timed automata (STA), two models combining dense-time and probabilities. While the decidability of the qualitative model-checking was already known for STA [9], the current approach yields general approximation results for the quantitative model-checking, which were not known before.

2 Preliminaries

2.1 Stochastic processes

Let $(\Omega, \Sigma, \mathbb{P})$ be a probabilistic space, that is, Ω is a set called the *universe*, Σ is a σ-algebra, and $\mathbb{P}$ is a probability measure over (Ω, Σ). Let (S, Σ') be a measurable space. A *stochastic*

process over $(\Omega, \Sigma, \mathbb{P})$ and (S, Σ') is a sequence $X = (X_i)_{i\geq 0}$ of random variables, where $X_i \colon \Omega \to S$ is a measurable function. Note that we do not assume stochastic processes are homogeneous or discrete.

Let $X = (X_i)_{i\geq 0}$ be a stochastic process. Given B a measurable set of S (that is, $B \in \Sigma'$), we will abuse notation and write B for the *uniform* sequence $(B_i)_{i\in\mathbb{N}}$ such that $B_i = B$ for every $i \geq 0$. Given B a measurable set of S, we will sometimes write $X_i \in B$ for the event $X_i^{-1}(B)$, and when B is a singleton $\{q\}$, we might even write $X_i = q$.

We fix for the rest of the paper a universe $(\Omega, \Sigma, \mathbb{P})$, and a measurable space (S, Σ').

▶ **Example 1.** Let us give an example of a discrete stochastic process representing a random walk over the state space $S = \{q_n \mid n \in \{-1\} \cup \mathbb{N}\}$. The stochastic process X is defined by

- $\mathbb{P}(X_0^{-1}(q_0)) = 1; \quad \forall i \geq 0\ \mathbb{P}(X_{i+1}^{-1}(q_{-1}) \mid X_i^{-1}(q_{-1})) = 1;$
- $\forall i \geq 0,\ \forall n \in \mathbb{N}\ \mathbb{P}(X_{i+1}^{-1}(q_{n+1}) \mid X_i^{-1}(q_n))) = \frac{3}{4}$ and $\mathbb{P}(X_{i+1}^{-1}(q_{n-1}) \mid X_i^{-1}(q_n))) = \frac{1}{4}$.

It should be noted that we do not mention the universe Ω. It is always possible to construct a universe that has such a probability measure $\mathbb{P}$, so that it is irrelevant to introduce it (see [14]). This remark holds true in each example of the paper.

▶ **Example 2.** Another example, is the following non-Markovian stochastic process X over the state space $S = \{q_0, q_0', q_1, q_1'\}$:

- $\mathbb{P}(X_0 = q_0) = \mathbb{P}(X_0 = q_0') = \frac{1}{2}; \quad \forall i \geq 1,\ \mathbb{P}(X_i = q_1' \mid X_0 = q_0') = 1;$
- $\mathbb{P}(X_1 = q_1 \mid X_0 = q_0) = \lambda_1$ and $\mathbb{P}(X_1 = q_1' \mid X_0 = q_0) = 1 - \lambda_1;$
- $\forall i \geq 1,\ \mathbb{P}(X_{i+1} = q_1 \mid X_i = q_1, X_0 = q_0) = \mathbb{P}(X_{i+1} = q_1 \mid X_i = q_1', X_0 = q_0) = \lambda_i;$
- $\forall i \geq 1,\ \mathbb{P}(X_{i+1} = q_1' \mid X_i = q_1, X_0 = q_0) = \mathbb{P}(X_{i+1} = q_1' \mid X_i = q_1', X_0 = q_0) = 1 - \lambda_i;$

where $(\lambda_i)_{i\in\mathbb{N}}$ is a sequence of reals in $[0, 1]$. Note that X could be made Markovian by changing the state space, with one bit of memory to remember the initial state.

Real-time stochastic processes. A particular class of stochastic processes will be of interest to us, namely real-time stochastic processes, in which the time evolution is important. We define (S_t, Σ_t) as the measurable space defined by $S_t = S \times \mathbb{R}_+$, and where Σ_t is the σ-algebra generated by Σ' and the Borel sets of $\mathbb{R}_+$ (denoted $\mathcal{B}(\mathbb{R}_+)$). A *real-time stochastic process* over (S, Σ) is a stochastic process $Z = (Z_i)_{i\geq 0}$ over (S_t, Σ_t) such that:

- for every $i \geq 0$, $Z_i = (X_i, \tau_i)$, where $X_i \colon \Omega \to S$ and $\tau_i \colon \Omega \to \mathbb{R}_+$ are random variables;
- for each $i \geq 0$, $\mathbb{P}\big(\{\omega \in \Omega \mid \tau_i(\omega) < \tau_{i+1}(\omega)\}\big) = 1$.

The process $X = (X_i)_{i\geq 0}$ somehow represents the spatial behaviour of the system, while the process $\tau = (\tau_i)_{i\geq 0}$ gives the time evolution of the system. The second condition ensures that time almost-surely progresses. We will say that Z is *almost-surely non-Zeno* whenever

$$\mathbb{P}\big(\{\omega \in \Omega \mid (\tau_i(\omega))_{i\geq 0} \text{ is bounded}\}\big) = 0.$$

In Section 5.3, we will see two classes of models that naturally fit into the framework of real-time stochastic processes. We can already mention here continuous-time Markov chains (we can find many examples of applications in [15]), or queuing systems (see below).

▶ **Example 3.** We consider a G/G/1-queue (of infinite capacity). A state of such a queue consists in the number of tasks waiting in the queue, the time delay since the last arrival in the queue (t_a) and the time delay since the last execution (t_e). Task arrivals follow a probability measure F_a, and task services are performed according to probability measure F_e. If at some point, the process is in state (n, t_a, t_e), the next arrival time in the queue is chosen according to $F_{a|t_a}$ and the next execution time is chosen according to $F_{e|t_e}$ where $F_{a|t_a}$ (resp. $F_{e|t_e}$) corresponds to the probability F_a (resp. F_e) given that at least t_a (resp.

t_e) has elapsed. This induces a stochastic process $X = (X_i)_{i\in\mathbb{N}}$ where X_i is the state of the queue after i steps. To turn it into a real-time stochastic process, one simply adds global time τ giving at step i the amount of time spent since the beginning.

▶ Remark. Real-time stochastic processes as defined above are *discrete-time* stochastic processes (if we follow standard vocabulary), since the random variables are indexed by $\mathbb{N}$. However they abstract real-time continuous behaviours by giving relevant snapshots of the system (at all times given by the τ_i's). Such abstractions are used for instance in [18, Theorem 1] for abstracting continuous-space pure jump Markov processes while keeping relevant information on the process. We will see in Section 5.3 that these processes capture behaviours of intrinsically time continuous systems.

Events. A stochastic process X over $(\Omega, \Sigma, \mathbb{P})$ and (S, Σ') allows one to define various events expressed using LTL-like notations. Let $\mathcal{L}_{S,\Sigma'}$ be the set of formulas defined by the grammar: $\varphi ::= B\,\mathbf{U}_{\bowtie n}\,B' \mid \mathbf{G}\,\mathbf{F}\,B \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi$, where $B = (B_i)_{i\geq 0}$ and $B' = (B'_i)_{i\geq 0}$ are sequences of measurable subsets of S, $\bowtie \in \{\geq, \leq, =\}$ is a comparison operator and $n \in \mathbb{N}$ is an integer. The semantics of formulas in $\mathcal{L}_{S,\Sigma'}$ in terms of events is defined inductively:

$$\mathsf{Ev}_X(B\,\mathbf{U}_{\bowtie n}B') = \bigcup_{i\bowtie n}\big(X_i^{-1}(B'_i) \cap \bigcap_{0\leq j<i} X_j^{-1}(B_j)\big)\ ;\ \mathsf{Ev}_X(\mathbf{G}\,\mathbf{F}\,B) = \bigcap_{i\geq 0}\bigcup_{j\geq i} X_j^{-1}(B_j)\ ;$$
$$\mathsf{Ev}_X(\varphi_1 \vee \varphi_2) = \mathsf{Ev}_X(\varphi_1) \cup \mathsf{Ev}_X(\varphi_2)\ ;\ \mathsf{Ev}_X(\varphi_1 \wedge \varphi_2) = \mathsf{Ev}_X(\varphi_1) \cap \mathsf{Ev}_X(\varphi_2)\ ;$$
$$\mathsf{Ev}_X(\neg\varphi) = \Omega \setminus \mathsf{Ev}_X(\varphi).$$

Note that all these events are measurable in Ω. Following the intuition behind the LTL notations, event $\mathsf{Ev}_X(B\,\mathbf{U}_{\bowtie n}B')$ means that the stochastic process X will eventually satisfy B' (within step constraint $\bowtie n$), and only visit B beforehand. Also, the intuition of $\mathbf{G}\,\mathbf{F}\,B$ is that B should be visited infinitely often. We use classical shorthands: $\top = (S)_{i\geq 0}$; $\bot = (\emptyset)_{i\geq 0}$; $B\,\mathbf{U}\,B' = B\,\mathbf{U}_{\geq 0}B'$; $\mathbf{F}\,B = \top\,\mathbf{U}\,B$; $\mathbf{F}_{\bowtie n}B = \top\,\mathbf{U}_{\bowtie n}B$; $\mathbf{G}\,B = \neg\mathbf{F}\,(\neg B)$, where $\neg B = (S \setminus B_i)_{i\geq 0}$.

2.2 Decisiveness

Abdulla *et al.* originally defined a denumerable Markov chain to be decisive w.r.t. a set of states F if its runs almost-surely reach F or a state from which F can no longer be reached [2]. In order to extend the concept of *decisiveness* to general stochastic processes, we first provide an analogue to the set of states from which F is not reachable.

▶ **Definition 4.** Let B, B' be sequences of measurable sets of S. B' is a *B-avoidance sequence* for the stochastic process X if it satisfies

$$\forall n \geq 0,\ \mathbb{P}\big(\mathsf{Ev}_X(\mathbf{F}_{=n}\,B' \wedge \mathbf{F}_{\geq n}\,B)\big) = 0. \tag{1}$$

Intuitively, B' corresponds to 'states' from which B is almost-surely avoided (due to non-homogeneity of X, it needs to be defined as a sequence by slices).

▶ Remark. For every sequence B, $(\emptyset)_{i\geq 0}$ is a B-avoidance sequence for X. One can also check that B-avoidance sequences are closed under denumerable unions and intersections.

▶ **Example 5.** Let us illustrate the notion of avoidance sequences on the stochastic processes from Examples 1 and 2. In Example 1, we consider the uniform sequence $B = \{q_5\}$. It can be shown that the set of B-avoidance sequences corresponds to all sequences B' with $B'_i \subseteq \{q_{-1}\}$. In Example 2, the following sequence defines a B-avoidance set for $B = \{q_1\}$: $B'_0 = \{q'_0\}$ and for every $n \geq 1$, $B'_n = \emptyset$.

For the rest of the section, we fix a stochastic process $X = (X_i)_{i\geq 0}$ with $X_i \colon \Omega \to S$. Several notions of decisiveness were proposed for discrete-time and implicitly denumerable Markov chains [2]. In this paper, we define three notions of decisiveness, adapting and refining the ones of [2] to general stochastic processes.

▶ **Definition 6.** Let B be a sequence of measurable sets in S and let B' be a B-avoidance sequence for X. We say that the stochastic process X is

- *initially decisive* (ID) w.r.t. B with *witness* B' if $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B \vee \mathbf{F}\, B')) = 1$
- *initially strongly decisive* (ISD) w.r.t. B with *witness* B' if $\mathbb{P}(\mathsf{Ev}_X(\mathbf{G}\,\mathbf{F}\, B \vee \mathbf{F}\, B')) = 1$
- *persistently decisive* (PD) w.r.t. B with *witness* B' if $\forall n \geq 0, \mathbb{P}\big(\mathsf{Ev}_X(\mathbf{F}_{\geq n}\, B \vee \mathbf{F}_{\geq n}\, B')\big) = 1$.

We will then say that X is ID (resp. ISD, PD) w.r.t. B whenever there is some B-avoidance sequence B' such that X is ID (resp. ISD, PD) w.r.t. B with witness B'.

▶ Remark. Note that X might be ID (resp. ISD, PD) w.r.t. B for some witness B', but not for some other B-avoidance sequence B''. However if $B' \subseteq B''$ and X is ID (resp. ISD, PD) w.r.t. B with witness B', then it is also decisive w.r.t. B with witness B'': the larger (for the inclusion) is the B-avoidance sequence, the better it is for decisiveness properties.

We can establish a relationship between the three decisiveness notions.

▶ **Lemma 7.** *Let B be a sequence of measurable sets in S and let B' be a B-avoidance sequence for X. X is* PD *w.r.t. B with witness B' implies that X is* ISD *w.r.t. B with witness B', which, in turns, implies that X is* ID *w.r.t. B with witness B'. Moreover, the converse implications do not hold.*

Example 1 shows that initial decisiveness and initial strong decisiveness are not equivalent (take $B = \{q_5\}$), and Example 2 shows the non-equivalence of initial strong decisiveness and persistent decisiveness (take $\lambda_i = \frac{1}{2}$ for every $i \in \mathbb{N}$, and $B = \{q_1\}$).

In the sequel, we will write B' for an arbitrary B-avoidance sequence for X. However, whenever X is ID (resp. ISD, PD) w.r.t. B with some witness B', we will choose an arbitrary witness and write it $\mathsf{Av}_{\mathsf{dec}}(B)$ (resp. $\mathsf{Av}_{\mathsf{str}}(B)$, $\mathsf{Av}(B)$). When they exist, it is then possible to recursively define B'- (resp. $\mathsf{Av}_{\mathsf{dec}}(B)$-, $\mathsf{Av}_{\mathsf{str}}(B)$- and $\mathsf{Av}(B)$-)avoidance sequences for X: those are then order-two avoidance sequences for B, which record states from which one avoids states, from which states in B are avoided! The previous notations extend in the same way for these order-two avoidance sequences.

▶ **Example 8.** Back to Example 2, for $B = \{q_1\}$, we saw that B' defined by $B'_0 = \{q'_0\}$ and $B'_i = \emptyset$ for each $i \geq 1$, is a B-avoidance sequence for X. Then, we can define a B'-avoidance sequence for X as follows: $B''_0 = \{q_0, q_1, q'_1\}$ and for each $i \geq 1$, $B''_i = \{q_0, q_1, q'_0, q'_1\}$. In fact, we can show that all B'-avoidance sequences are the sequences included in B''.

3 Analysis of decisive stochastic processes

In this section we show how decisiveness properties can help analysing stochastic processes. In the first part, we focus on qualitative (that is, probability 0 or 1) reachability and repeated reachability properties, and we reduce all the corresponding model-checking questions to checking that some reachability property has probability 0. While this could be reduced to graph properties in [2], this is not the case here, since our models might have infinite non-denumerable branching. When we will apply these results in Subsection 5.3, models will have good properties allowing to solve the 0-probability properties of reachability properties. In the two next parts, we will use decisiveness properties to draw general procedures for

computing (arbitrary) approximations of the probability of a (repeated) reachability property. Effectiveness of these procedures will of course rely on good effectiveness properties of the models that we want to analyze.

3.1 Qualitative reachability and repeated reachability

We aim at describing a procedure for checking the almost-sure satisfiability of a reachability (resp. a repeated reachability) property, that is, an event of the form $\mathbf{F}\, B$ (resp. $\mathbf{G}\,\mathbf{F}\, B$), where B is a sequence of measurable sets. We fix B' a B-avoidance sequence, and we recall the notations $\mathsf{Av}_{\mathsf{dec}}(B)$, $\mathsf{Av}_{\mathsf{str}}(B)$ and $\mathsf{Av}(B)$ for such sequences when X is ID, resp. ISD, resp. PD w.r.t. B.

▶ **Proposition 9.**
- *If* $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B)) = 1$, *then* $\mathbb{P}(\mathsf{Ev}_X(\neg B\,\mathbf{U}\, B')) = 0$.
- *If* X *is* ID *w.r.t.* B *and* $\mathbb{P}(\mathsf{Ev}_X(\neg B\,\mathbf{U}\,\mathsf{Av}_{\mathsf{dec}}(B))) = 0$, *then* $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B)) = 1$.

Under an initial decisiveness assumption, this reduces the almost-sure model-checking of reachability properties to checking that some kind of (constrained) reachability property is satisfied with probability 0. Note that, contrary to the case of discrete-time denumerable Markov chains, we cannot reduce to graph properties, yet we advocate that for reachability properties, checking whether the probability is 0, is simpler than checking whether it is 1. We will see in Subsection 5.3 how this can be exploited on specific examples.

Turning to almost-sure repeated reachability, one can show the following proposition:

▶ **Proposition 10.**
- *If* $\mathbb{P}(\mathsf{Ev}_X(\mathbf{G}\,\mathbf{F}\, B)) = 1$, *then* $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B')) = 0$;
- *If* X *is* ISD *w.r.t.* B *and* $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\,\mathsf{Av}_{\mathsf{str}}(B))) = 0$, *then* $\mathbb{P}(\mathsf{Ev}_X(\mathbf{G}\,\mathbf{F}\, B)) = 1$.

Under an initial strong decisiveness assumption, this reduces the almost-sure model-checking of a repeated reachability property to the 0-model-checking of some reachability property.

Concerning the positive model-checking of repeated reachability properties, one can show:

▶ **Proposition 11.**
- *If* X *is* PD *w.r.t.* B *and* ID *w.r.t.* $\mathsf{Av}(B)$, *and if* $\mathbb{P}\big(\mathsf{Ev}_X(\mathbf{G}\,\mathbf{F}\, B)\big) > 0$, *then* $\mathbb{P}\big(\mathsf{Ev}_X(\mathbf{F}\,\mathsf{Av}_{\mathsf{dec}}(\mathsf{Av}(B)))\big) > 0$;
- *If* X *is* PD *w.r.t* B *and* $\mathbb{P}\big(\mathsf{Ev}_X(\mathbf{F}\,\mathsf{Av}(B)')\big) > 0$, *then* $\mathbb{P}\big(\mathsf{Ev}_X(\mathbf{G}\,\mathbf{F}\, B)\big) > 0$.

Note that the existence of a witness B' such that X is PD w.r.t. B does not imply the existence of a witness such that X is ID w.r.t. B'.

3.2 Quantitative reachability

We assume B is a sequence of measurable sets of S and B' is a B-avoidance sequence. We define the two following sequences ($n \in \mathbb{N}$):

$$\begin{cases} p_n^{\mathsf{Yes}} & = \mathbb{P}\big(\mathsf{Ev}_X(\mathbf{F}_{\leq n}\, B)\big) \\ p_n^{\mathsf{No}} & = \mathbb{P}\big(\mathsf{Ev}_X(\neg B\,\mathbf{U}_{\leq n}\, B')\big) \end{cases}$$

The next proposition gives straightforward properties of these two sequences.

▶ **Proposition 12.** *The sequences* $(p_n^{\mathsf{Yes}})_{n\geq 0}$ *and* $(p_n^{\mathsf{No}})_{n\geq 0}$ *are non-decreasing and converge respectively to* $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B))$ *and* $\mathbb{P}(\mathsf{Ev}_X(\neg B\,\mathbf{U}\, B'))$.

▶ **Corollary 13.** *If X is* ID *w.r.t. B and $B' = \mathsf{Av}_{\mathsf{dec}}(B)$, then* $\lim_{n\to\infty} p_n^{\mathsf{Yes}} + p_n^{\mathsf{No}} = 1$.

Corollary 13 can be used to derive an approximation scheme to evaluate the probability of reachability properties in ID stochastic processes. Indeed, given a fixed error bound $\varepsilon > 0$, in order to compute $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B))$ up to ε, one only needs to iteratively compute the values p_n^{Yes} and p_n^{No} until $1 - p_n^{\mathsf{Yes}} - p_n^{\mathsf{No}} \leq \varepsilon$ to deduce that $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B)) - p_n^{\mathsf{Yes}} \leq \varepsilon$. In case p_n^{Yes} and p_n^{No} cannot be computed exactly, but can only be approximated up to any desired error bound, this scheme can be refined to obtain a 2ε-approximation for $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B))$.

▶ Remark. Note that the quality of the above approximation scheme depends on the choice of the sequence $\mathsf{Av}_{\mathsf{dec}}(B)$: the larger $\mathsf{Av}_{\mathsf{dec}}(B)$, the faster convergence. Intuitively, $\mathsf{Av}_{\mathsf{dec}}(B)$ permits to stop the exploration when the reachability goal can no longer be satisfied, hence the sooner the better.

3.3 Quantitative repeated reachability

We assume B is a sequence of measurable sets of S, B' is a B-avoidance sequence, and B'' is a B'-avoidance sequence. We define the two following sequences ($n \in \mathbb{N}$):

$$\begin{cases} q_n^{\mathsf{Yes}} = \mathbb{P}\big(\mathsf{Ev}_X(\neg B'\, \mathbf{U}_{\leq n}\, B'')\big) \\ q_n^{\mathsf{No}} = \mathbb{P}\big(\mathsf{Ev}_X(\neg B''\, \mathbf{U}_{\leq n}\, B')\big) \end{cases}$$

▶ **Proposition 14.** *The sequences $(q_n^{\mathsf{Yes}})_{n\geq 0}$ and $(q_n^{\mathsf{No}})_{n\geq 0}$ are non-decreasing and converge respectively to $\mathbb{P}\big(\mathsf{Ev}_X(\neg B'\, \mathbf{U}\, B'')\big)$ and $\mathbb{P}\big(\mathsf{Ev}_X(\neg B''\, \mathbf{U}\, B')\big)$.*

▶ **Proposition 15.** *If X is* PD *w.r.t. B (with witness $B' = \mathsf{Av}(B)$) and* ID *w.r.t. $\mathsf{Av}(B)$ (with witness $B'' = \mathsf{Av}_{\mathsf{dec}}(\mathsf{Av}(B))$), then the two sequences $(q_n^{\mathsf{Yes}})_{n\geq 0}$ and $(1 - q_n^{\mathsf{No}})_{n\geq 0}$ are adjacent and converge to $\mathbb{P}(\mathsf{Ev}_X(\mathbf{G}\, \mathbf{F}\, B))$.*

Here again, the convergence of the two adjacent sequences can be used to derive an approximation scheme for $\mathbb{P}(\mathsf{Ev}_X(\mathbf{G}\, \mathbf{F}\, B))$ in PD stochastic processes.

Note that the persistent decisiveness property is required for the approximation scheme to be correct: consider again Example 2 and assume the sequence $(\lambda_i)_{i\in\mathbb{N}}$ satisfies $\prod_{i\in\mathbb{N}}(1-\lambda_i) > 0$. Under that hypothesis, one can show that $\mathbb{P}(\mathsf{Ev}_X(\mathbf{G}\, \mathbf{F}\, B)) < \frac{1}{2}$. On the other hand, whatever the choice of the avoidance sequences, we never get that the two sequences $(q_n^{\mathsf{Yes}})_{n\geq 0}$ and $(1 - q_n^{\mathsf{No}})_{n\geq 0}$ converge to that value.

4 Time-bounded reachability in real-time stochastic processes

In this section, we explain how to use decisiveness towards the quantitative analysis of time-bounded reachability (or safety) properties for real-time stochastic processes.

We fix (S_t, Σ_t) the measurable space for real-time stochastic processes we will consider. For $\Delta \in \mathbb{R}_+$ a time bound and B a sequence of measurable sets of S_t, we define the sequence $B \cap (t \leq \Delta)$ by: $(B \cap (t \leq \Delta))_i = \{(s, \tau) \in B_i \mid \tau \leq \Delta\}$. $B \cap (t \leq \Delta)$ is thus the restriction of B in which the time component is bounded by Δ.

First, decisiveness w.r.t. a sequence B propagates to its time-bounded restriction:

▶ **Proposition 16.** *Let $Z = (X_i, \tau_i)_{i\geq 0}$ be a real-time stochastic process, B be a sequence of measurable sets of S_t, and Δ be a time bound. If Z is* ID*, resp.* ISD*, resp.* PD *w.r.t. B, then Z is* ID*, resp.* ISD*, resp.* PD *w.r.t. $B \cap (t \leq \Delta)$.*

More importantly, non-Zeno real-time stochastic processes are PD w.r.t. time-bounded sequences:

▶ **Theorem 17.** *Let $Z = (X_i, \tau_i)_{i\geq 0}$ be a real-time stochastic process, let B be a sequence of measurable sets, and let $\Delta \in \mathbb{R}_+$ be a time bound. If Z is almost-surely non-Zeno, then Z is* PD *w.r.t. any time-bounded sequence $B \cap (t \leq \Delta)$.*

The main argument to establish this theorem is that, assuming almost-sure non-Zenoness, the sequence defined by $t > \Delta$ is a $(B \cap (t \leq \Delta))$-avoidance sequence for every B.

The almost-sure non-Zenoness hypothesis is standard and a desirable property of a system, and it expresses that the system should not have infinitely many discrete changes in a bounded amount of time. This assumption is satisfied by continuous-time Markov chains [6], and is easily enforced in many other models, such as continuous-time Markov processes with bounded transition rates [12], or continuous-space pure jump Markov processes (cPJMPs) assuming a non-explosive property [18].

This result then implies that for all these systems, if B is simple enough (like a uniform sequence of sets), provided one can compute (or approximate) the probability in n steps to reach some set of states, one can approximate the probability of satisfying a time-bounded reachability or safety property. This allows us to partly recover the result for cPJMPs [18, Theorem 3] which was established in a more analytical way.

In its generality, our approximation scheme does not provide any convergence rate, but for stochastic processes for which we can have an upper bound on the probability of completing at least n discrete changes within Δ time units, we will be able to compute a convergence rate for the various schemes. For instance for continuous-time (denumerable) Markov chains whose transition rates are upper-bounded by Λ, that probability can be bounded using a Poisson process of rate Λ.

5 Effectivity through abstraction

Proving decisiveness of general stochastic processes can be a hard task, in particular when their state-space is continuous. Decidability results in this context are often obtained through discrete abstractions. Therefore, in order to analyze the decisiveness of such stochastic processes, we propose to rely on an abstraction. More precisely, we give in this section sufficient conditions on an abstraction to ensure decisiveness of the original stochastic process. The qualitative verification algorithms and quantitative approximation schemes can then be applied to the concrete model. Note that, in general, the abstractions we propose only preserve qualitative properties, so that approximation schemes should be applied to the concrete model, not to the abstraction.

We then explain how this methodology can be applied to two classes of real-time stochastic processes, namely the ones generated by generalized semi-Markov processes and by stochastic timed automata. These models can be abstracted into discrete-time Markov chains while preserving the almost-sure satisfaction of reachability properties; this allows us to derive good decisiveness properties of the original models, and thus to infer approximation schemes.

5.1 Decisiveness for homogeneous denumerable Markov chains

Let us recall some basics of Markov chains. A denumerable Markov chain (MC, for short) is a stochastic process $Y = (Y_i)_{i\geq 0}$ with a denumerable state space T and which has the Markov property: for every $n \geq 0$, for all $t, t_0, t_1, \ldots, t_n \in T$, as soon as $\mathbb{P}(\bigwedge_{i=0}^{n} Y_i = t_i) > 0$, then $\mathbb{P}(Y_{n+1} = t \mid \bigwedge_{i=0}^{n} Y_i = t_i) = \mathbb{P}(Y_{n+1} = t \mid Y_n = t_n)$. The MC is *homogeneous* if for every n and for all $t, t' \in T$, $\mathbb{P}(Y_{n+1} = t' \mid Y_n = t) = \mathbb{P}(Y_n = t' \mid Y_{n-1} = t)$. In that case, the Markov chain is generated by a transition matrix $p_Y \colon T \times T \to [0, 1]$ such that

for every $t \in T$, $\sum_{t' \in T} p_Y(t,t') = 1$ and for every $n \geq 0$, $p_Y(t,t') = \mathbb{P}(Y_{n+1} = t' \mid Y_n = t)$. Under the condition that Y is homogeneous, one can simply define runs generated by Y. Precisely, a sequence $t_0, t_1, \cdots, t_n$ is a run, denoted $t_0 \to t_1 \cdots \to t_n$ if $p_Y(t_i, t_{i+1}) > 0$ for every $0 \leq i < n$; n is then the *length* of the run, and we write $t \to^* S$ as soon as there exists a state $t' \in S \subseteq T$ and a run $t = t_0 \to t_1 \cdots \to t_k = t'$.

Let us first explain how to characterize our various notions of decisiveness in the case of homogeneous MCs, and how they compare to the decisiveness of [2]. For every subset of states $C \subseteq T$, borrowing notations from [2], we let $\widetilde{C} = \{t \in T \mid t \not\to^* C\}$. State $t_0 \in T$ is an initial state of Y if $\mathbb{P}_Y(Y_0 = t_0) > 0$, and Y is said initialized at t_0 whenever t_0 is the unique initial state, that is $\mathbb{P}_Y(Y_0 = t_0) = 1$. If Y is a homogeneous MC, we write $Y[t]$ for the MC initialized at t, with transition matrix p_Y.

▶ **Lemma 18.** *Let Y be a homogeneous MC. For every C, $\widetilde{C}$ is a C-avoidance uniform sequence for Y. Moreover, it is maximal for the inclusion.*

The maximality property stated above allows to check decisiveness properties only with the witness $\widetilde{C}$: if Y is decisive with witness B', since $B' \subseteq \widetilde{C}$, it will also be decisive with witness $\widetilde{C}$. Recovering partly the original definitions of [2], we obtain the following characterization of our three notions of decisiveness:

▶ **Corollary 19.** *Let Y be a homogeneous MC, and C a set of states. Then:*

1. *Y is* ID *w.r.t. C iff $\mathbb{P}_Y\big(\mathsf{Ev}_Y(\mathbf{F}\, C \vee \mathbf{F}\, \widetilde{C})\big) = 1$;*
2. *Y is* ISD *w.r.t. C iff $\mathbb{P}_Y\big(\mathsf{Ev}_Y(\mathbf{G}\,\mathbf{F}\, C \vee \mathbf{F}\, \widetilde{C})\big) = 1$;*
3. *Y is* PD *w.r.t. C iff for every $p \geq 0$, $\mathbb{P}_Y\big(\mathsf{Ev}_Y(\mathbf{F}_{\geq p}\, C \vee \mathbf{F}_{\geq p}\, \widetilde{C})\big) = 1$ iff for every state t reachable from an initial state, $Y[t]$ is* ID *w.r.t. C.*

The third characterization implies that the decisiveness notion of [2] corresponds to our persistent decisiveness notion, in the case of homogeneous MCs.

Contrary to the case of general stochastic processes, initial strong decisiveness and persistent decisiveness coincide for homogeneous MCs (we recover here [2, Lemma 3.2]).

▶ **Lemma 20.** *Let Y be a homogeneous MC, and C a set of states. Then, Y* ISD *w.r.t. C iff Y is* PD *w.r.t. C.*

Note though that, even in this restricted context, initial decisiveness is not equivalent to initial strong decisiveness (recall Example 5). Finally, as already noticed in [2]:

▶ **Lemma 21.** *Let Y be a finite homogeneous MC, and C a set of states. Then, Y is* PD *w.r.t. C.*

5.2 Sound abstraction for decisiveness

Let us define a suitable notion of abstraction relating an arbitrary stochastic process X and a homogeneous MC Y such that decisiveness of Y implies decisiveness for X.

▶ **Definition 22.** Let $X = (X_i)_{i \geq 0}$ be a stochastic process, $Y = (Y_i)_{i \geq 0}$ be a homogeneous MC with denumerable state-space T equipped with the discrete σ-algebra $\Theta = 2^T$, and $\alpha : (S, \Sigma') \to (T, \Theta)$ be a mapping such that α and α^{-1} are measurable. The MC Y is an *α-abstraction* of X if for every sequence $A = (A_n)_{n \geq 0}$ of sets in Θ and for every $n \geq 0$

$$\mathbb{P}_Y\big(Y_n^{-1}(A_n) \mid \bigcap_{i<n} Y_i^{-1}(A_i)\big) > 0 \iff \mathbb{P}\big(X_n^{-1}(\alpha^{-1}(A_n)) \mid \bigcap_{i<n} X_i^{-1}(\alpha^{-1}(A_i))\big) > 0 \ .$$

Intuitively, Y is an α-abstraction of X if through the mapping α it preserves the events that may happen with positive probability.

In order to lift avoidance sequence from the abstraction to the concrete stochastic process, we rely on α-closed sets. A set $B \in \Sigma'$ is *α-closed* if $b \in B$ and $\alpha(b) = \alpha(b')$ implies $b' \in B$. Given $B \in \Sigma'$, by Lemma 18, $\widetilde{\alpha(B)}$ is an $\alpha(B)$-avoidance sequence for Y. Assuming B is α-closed, we obtain a B-avoidance sequence for X.

▶ **Lemma 23.** *Let Y be an α-abstraction of X and $B \in \Sigma'$ be an α-closed set. Then $(\alpha^{-1}(\widetilde{\alpha(B)}))$ is a B-avoidance sequence for X.*

However, α-abstractions do not necessarily preserve decisiveness properties, yet these can be ensured thanks to the following soundness notions.

▶ **Definition 24.** Let Y be an α-abstraction of X.

- Y is *sound* if for every α-closed set B, $\mathbb{P}_Y(\mathsf{Ev}_Y(\mathbf{F}\, \alpha(B))) = 1$ implies $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}\, B)) = 1$.
- Y is *persistently sound* if for every α-closed set B and every $p \geq 0$, $\mathbb{P}_Y(\mathsf{Ev}_Y(\mathbf{F}_{\geq p}\, \alpha(B))) = 1$ implies $\mathbb{P}(\mathsf{Ev}_X(\mathbf{F}_{\geq p}\, B)) = 1$.

Roughly said, sound abstractions preserve almost-sure satisfaction of reachability properties. Moreover, they allow to transfer decisiveness properties to the original stochastic process.

▶ **Proposition 25.** *Let Y be an α-abstraction of X and B an α-closed set.*

- *If Y is sound and* ID *w.r.t. $\alpha(B)$, then X is* ID *w.r.t. B with witness $(\alpha^{-1}(\widetilde{\alpha(B)}))$.*
- *If Y is persistently sound and* PD *w.r.t. $\alpha(B)$, then X is* PD *w.r.t. B with witness $(\alpha^{-1}(\widetilde{\alpha(B)}))$.*

▶ **Example 26.** Back to the queue of Example 3, we assume it is M/M/1, that is, F_a and F_e are exponential distributions of parameters λ and μ, respectively. Assuming $\lambda < \mu$, we can exhibit a persistently sound abstraction. Indeed, consider the random walk over $\mathbb{N}$ defined by $p(0,1) = 1$, and if $i \geq 1$, $p(i, i+1) = \frac{\lambda}{\lambda+\mu}$ and $p(i, i-1) = \frac{\mu}{\lambda+\mu}$. Since $\lambda < \mu$, this MC is PD w.r.t. each set of states and thus, the queue is PD w.r.t. each set of states that is closed under the abstraction.

5.3 Applications

We apply the previous study to two classes of systems.

5.3.1 Generalized semi-Markov processes

A generalized semi-Markov process [10, 13] is a stochastic process built on a finite set of events. Each event is equipped with a random variable representing its duration: either a variable-delay defined by a density function or a fixed-delay modelled by a Dirac distribution. A transition is characterized by a set of events which expire, and schedules a set of new events. This model is known to generalize continuous-time Markov chains.

The semantics of a GSMP $\mathcal{M}$ is given as a general state-space Markov chain (GSSMC), defined by a set of configurations and a transition kernel. Configurations of a GSMP are pairs consisting of a state and a valuation assigning a time value to each *scheduled* event. Such a value represents the time elapsed since the event was scheduled. Transitions between configurations combine a time-elapse and the occurrence of some scheduled events and/or the scheduling of new events. The set of configurations can be equipped with a natural σ-algebra $\mathcal{G}$, and the transition system induced by $\mathcal{M}$ is equipped with a transition kernel: for a configuration γ and a set $A \in \mathcal{G}$, $P_{\mathcal{M}}(\gamma, A)$ is the probability to move in one step from

configuration γ to some configuration in A. This probability expresses a race between all enabled events, taking into account their residual density functions. The set of runs, *i.e.* infinite sequences of configurations, can then be equipped with a probability measure $\mathbb{P}_{\mathcal{M}}$. The GSSMC associated with a GSMP $\mathcal{M}$ can thus naturally be viewed as a (real-time) stochastic process $X^{\mathcal{M}}$.

Decisiveness. In general, GSMPs do not enjoy any decisiveness property. Indeed, [10, Section 3] presents an example that is not ID w.r.t. any region-closed set. Still, Brázdil *et al.* identified a sufficient condition, that ensures some kind of fairness: GSMP should be *single-ticking* (GSMP with some restriction on fixed-delay events, see [10] for the definition of this condition). Under that condition and using results of [10], one can show that the standard region abstraction for GSMPs is a persistently sound abstraction. More precisely, we consider as an abstraction the (finite-state) Markov chain $Y^{\mathcal{M}}$, whose states are regions, and such that there is a transition between region r and region r' as soon as there is a configuration $\gamma \in r$ from which the probability to reach r' in one step in $X^{\mathcal{M}}$ is positive. Probabilities are assumed to be uniform in $Y^{\mathcal{M}}$.

▶ **Theorem 27.** *Let $\mathcal{M}$ be a single-ticking GSMP with stochastic process $X^{\mathcal{M}}$. Then the region Markov chain $Y^{\mathcal{M}}$ is an persistently sound abstraction of $X^{\mathcal{M}}$.*

Proposition 25 then suffices to derive the decisiveness of the original stochastic process $X^{\mathcal{M}}$:

▶ **Corollary 28.** *Let $\mathcal{M}$ be a single-ticking GSMP with stochastic process $X^{\mathcal{M}}$. Then for every region-closed set B, the stochastic process $X^{\mathcal{M}}$ is* PD *w.r.t. B.*

As a consequence, we can apply all results of Section 3 to single-ticking GSMPs. We remark here that checking whether a reachability property has probability 0 can easily be done using the region graph abstraction: it amounts to checking in the (finite) region abstraction that there is no path from the initial state to the target [10]. Hence all qualitative questions related to region-based (repeated) reachability properties can be solved. For what concerns quantitative verification, assuming the distributions equipping the GSMPs can be handled numerically, this allows one to approximate the probability of reachability or repeated reachability properties, as well as all time-bounded reachability properties. We believe our approach gives new hints into the approximate model-checking problem for GSMPs, for which, up to our knowledge, only few results are known. For instance in [3, 7], the authors approximate the probability of until formulas of the form "the system reaches a target before time T within k discrete events, while staying within a set of safe states" (resp. "the system reaches a target while staying within a set of safe states") for GSMPs (resp. a restricted class of GSMPs which can be proved to be PD), and study numerical aspects. Our result permits to do the same with any reachability (resp. time-bounded reachability) property on the whole class of single-ticking GSMPs (resp. which are a.s. non-Zeno). The numerical aspects in our computations can be dealt with as in [3, 7].

5.3.2 Stochastic timed automata

Stochastic timed automata [9] are stochastic processes derived from timed automata [4] by randomizing both the delays and the edge choices. One can naturally associate a (real-time) stochastic process $X^{\mathcal{A}}$ with an STA $\mathcal{A}$.

Decisiveness. Similarly to GSMP, STA are not decisive in general. Adapting an example from [9], one can indeed exhibit an STA which is not ID w.r.t. a given region-closed set. Still, under some fairness property, one can build an abstraction of the STA, that is sound for the almost-sure model checking of LTL properties and the-like [9]. It turns out that this fairness assumption ensures that the same abstraction is also sound for decisiveness. As for GSMPs, we consider the natural region abstraction, and we define a finite-state Markov chain $Y^{\mathcal{A}}$, in which the states are regions, and there is a transition from one region r to another r' as soon as there exists a configuration γ in r from which the probability to reach r' in one step in $X^{\mathcal{A}}$ is positive. As mentioned earlier, as such, the abstraction $Y^{\mathcal{A}}$ is not sound in general. Yet, it preserves almost-sure satisfaction of LTL properties when the stochastic timed automaton is *almost-surely fair* [9]. Here fairness refers as the following property, which depends on $Y^{\mathcal{A}}$: every edge of $Y^{\mathcal{A}}$ which is enabled infinitely often along a run should be taken infinitely often. Denoting fair this property, the assumption $\mathbb{P}(\mathsf{Ev}_{X^{\mathcal{A}}}(\mathsf{fair})) = 1$ suffices to prove that $Y^{\mathcal{A}}$ is a sound abstraction for the almost-sure model checking for LTL properties [9].

▶ **Theorem 29.** *Let $\mathcal{A}$ be an STA with associated stochastic process $X^{\mathcal{A}}$, and let $Y^{\mathcal{A}}$ be its region abstraction. If $\mathbb{P}(\mathsf{Ev}_{X^{\mathcal{A}}}(\mathsf{fair})) = 1$ then $Y^{\mathcal{A}}$ is a persistently sound abstraction for $X^{\mathcal{A}}$.*

As a consequence of Proposition 25 and Theorem 29, we derive the decisiveness of the original stochastic process $X^{\mathcal{A}}$:

▶ **Corollary 30.** *Let $\mathcal{A}$ be an STA with stochastic process $X^{\mathcal{A}}$ and $Y^{\mathcal{A}}$ its region abstraction. If $\mathbb{P}(\mathsf{Ev}_{X^{\mathcal{A}}}(\mathsf{fair})) = 1$ then for every region-closed set B, $X^{\mathcal{A}}$ is PD w.r.t. B.*

As a consequence, we can apply all results of Section 3 to almost-surely fair stochastic timed automata. While the decidability of qualitative model-checking questions that we can infer from Section 3.1 were already known [9] and can be solved on the region abstraction, the approximation schemes that we can derive from Sections 3.2 and 3.3 are new. Obtaining decidability results even for the qualitative model-checking of large classes of STA required quite some effort (now combined in [9]). Here, we show that our earlier approach importantly implied decisiveness properties for STA. Moreover, the approximation schemes of Sections 3.2 and 3.3 can now be effectively applied to STA, as soon as distributions in the model have good numerical properties. Notice that a first decidability result was obtained in [8] for the quantitative model-checking of a restricted class of single-clock STA: while the current approximation schemes apply to all single-clock STA, the closed-form expression obtained in [8], although more precise requires a condition on the cycles of the automaton.

6 Conclusion and future work

In this paper, we introduced and studied decisiveness for general stochastic processes, setting sufficient conditions for the decidability of qualitative model-checking of (repeated) reachability properties, and more importantly for the approximability of the quantitative evaluation of such properties. We then showed that non-Zeno real-time stochastic processes have good decisiveness properties, allowing one to approximate the probability of all time-bounded properties. Finally we described a framework to obtain decisiveness properties through abstractions, and demonstrated its applicability to generalized semi-Markov processes and stochastic timed automata, thus yielding new approximability results for the quantitative model-checking of stochastic timed automata.

As further work, we would like to extend the applicability of our approach to other classes of stochastic timed systems, like probabilistic extensions of timed lossy channel systems [1]

or communicating timed systems [11]. Also, the approximation scheme for reachability properties can be adapted to evaluate an expected accumulated reward, provided the reward evolves linearly in the model, as in Markov reward models [5, 16]. Finally, extending the approximation schemes to Muller conditions would enable the quantitative analysis of properties given as LTL formulas.

References

1 Parosh Aziz Abdulla, Mohamed Faouzi Atig, and Jonathan Cederberg. Timed lossy channel systems. In *Proc. 31sth Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'12)*, volume 18 of *LIPIcs*, pages 374–386. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.374.

2 Parosh Aziz Abdulla, Noomene Ben Henda, and Richard Mayr. Decisive Markov chains. *Logical Methods in Computer Science*, 3(4), 2007.

3 Rajeev Alur and Mikhail Bernadsky. Bounded model checking for GSMP models of stochastic real-time systems. In *Proc. 9th International Workshop on Hybrid Systems: Computation and Control (HSCC'06)*, volume 3927 of *Lecture Notes in Computer Science*, pages 19–33. Springer, 2006.

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

5 Christel Baier. Reasoning about cost-utility constraints in probabilistic models. In *Proc. 9th Workshop on Reachability Problems in Computational Models (RP'15)*, volume 9328 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2015.

6 Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(7):524–541, 2003.

7 Mikhail Bernadsky and Rajeev Alur. Symbolic analysis for GSMP models with one stateful clock. In *Proc. 10th International Workshop on Hybrid Systems: Computation and Control (HSCC'07)*, volume 4416 of *Lecture Notes in Computer Science*, pages 90–103. Springer, 2007.

8 Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Quantitative model-checking of one-clock timed automata under probabilistic semantics. In *Proc. 5th International Conference on Quantitative Evaluation of Systems (QEST'08)*. IEEE Computer Society Press, 2008.

9 Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, Quentin Menet, Christel Baier, Marcus Größer, and Marcin Jurdziński. Stochastic timed automata. *Logical Methods in Computer Science*, 10(4):1–73, 2014.

10 Tomáš Brázdil, Jan Krčál, Jan Křetínský, and Vojtěch Řehák. Fixed-delay events in generalized semi-Markov processes revisited. In *Proc. 22nd International Conference on Concurrency Theory (CONCUR'11)*, volume 6901 of *Lecture Notes in Computer Science*, pages 140–155. Springer, 2011.

11 Lorenzo Clemente, Frédéric Herbreteau, Amélie Stainer, and Grégoire Sutre. Reachability of communicating timed processes. In *Proc. 16th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'13)*, volume 7794 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2013.

12 Josée Desharnais and Prakash Panangaden. Continuous stochastic logic characterizes bisimulation of continuous-time Markov processes. *Journal of Logic and Algebraic Programming*, 56:99–115, 2003.

13 Peter W. Glynn. A GSMP formalism for discrete event systems. *Proceedings of the IEEE*, 77(1):14–23, 1989.

14 Geoffrey R. Grimmett and David R. Stirzaker. *Probability and Random Processes*. Oxford University Press, 1992.

15 Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. Time-bounded model checking of infinite-state continuous-time Markov chains. *Fundamenta Informaticae*, 95(1):129–155, 2009.

16 Ronald A. Howard. *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*. John Wiley & Sons, 1971.

17 S. Purushothaman Iyer and Murali Narasimha. Probabilistic lossy channel systems. In *Proc. 7th International Joint Conference on Theory and Practice of Software Development (TAPSOFT'97)*, volume 1214 of *Lecture Notes in Computer Science*, pages 667–681. Springer, 1997.

18 Sadegh Soudjani, Rupak Majumdar, and Alessandro Abate. Safety verification of continuous-space pure jump Markov processes. In *Proc. 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'16)*, volume 9636 of *Lecture Notes in Computer Science*, pages 147–163. Springer, 2016.

Composition of Stochastic Transition Systems Based on Spans and Couplings*

Daniel Gburek[1], Christel Baier[2], and Sascha Klüppelholz[3]

1 **Faculty of Computer Science, Technische Universität Dresden, Dresden, Germany**
daniel.gburek@tu-dresden.de
2 **Faculty of Computer Science, Technische Universität Dresden, Dresden, Germany**
christel.baier@tu-dresden.de
3 **Faculty of Computer Science, Technische Universität Dresden, Dresden, Germany**
sascha.klueppelholz@tu-dresden.de

Abstract

Conventional approaches for parallel composition of stochastic systems relate probability measures of the individual components in terms of product measures. Such approaches rely on the assumption that components interact stochastically independent, which might be too rigid for modeling real world systems. In this paper, we introduce a parallel-composition operator for stochastic transition systems that is based on couplings of probability measures and does not impose any stochastic assumptions. When composing systems within our framework, the intended dependencies between components can be determined by providing so-called spans and span couplings. We present a congruence result for our operator with respect to a standard notion of bisimilarity and develop a general theory for spans, exploiting deep results from descriptive set theory. As an application of our general approach, we propose a model for stochastic hybrid systems called stochastic hybrid motion automata.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Stochastic Transition System, Composition, Stochastic Hybrid Motion Automata, Stochastically Independent, Coupling, Span, Bisimulation, Congruence, Polish Space

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.102

1 Introduction

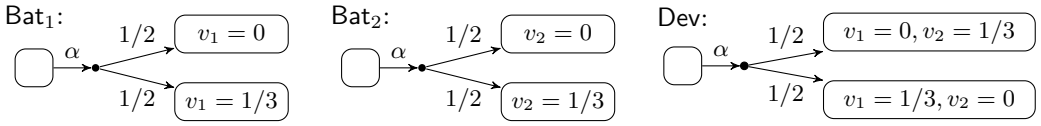
When modeling complex systems, compositional approaches enjoy many favorable properties compared to their monolithic counterparts. They allow for a systematic system design, facilitate the interchangeability and reusability of components, and thus also ease the maintainability. A major objective in defining compositional frameworks is to separate concerns into components – specifying the operational behavior – and composition operators – addressing the communication and interaction of the components. Within conventional approaches for stochastic systems, the composition operator relates probability distributions of the individual components in terms of product distributions. Therefore, such operators

* The authors are supported by the DFG through the Collaborative Research Center SFB 912 – HAEC, the Excellence Initiative by the German Federal and State Governments (cluster of excellence cfAED and Institutional Strategy), the Research Training Groups QuantLA (GRK1763) and RoSI (GRK1907).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 102; pp. 102:1–102:15

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

are based on the assumption that the components interact stochastically independent, which is often not adequate. For instance, let us regard a systems composed of a device Dev and two batteries Bat_1 and Bat_2 providing the energy for Dev as detailed below:



In this example, the device provides the environmental context in which Bat_1 and Bat_2 are operating. Hence, Dev may, e.g., be the reason for common cause failures arising in the system. Let the variables v_1 and v_2 capture the amount of energy stored within Bat_1 and Bat_2, respectively. The action label α stands for the occurrence of a failure after which all the components will crash. As a consequence, the level of the stored energy of the batteries instantaneously drops to either 0 or 1/3 with probability 1/2, respectively. When considering the batteries in isolation, Bat_1 and Bat_2 appear stochastically independent in the first place and thus, product distributions in the parallel composition $\mathsf{Bat}_1 \parallel \mathsf{Bat}_2$ seem to be adequate. However, additional dependencies can be imposed by Dev, influencing the interplay between the batteries. The assumption that Bat_1 and Bat_2 are stochastically independent is hence not adequate. Assume, e.g., that Dev uses Bat_1 as the default power supply and Bat_2 as a backup. Then, within a failure situation, Bat_1 is more likely to be affected than Bat_2. The most likely case is that Bat_1 drops to 0 whereas Bat_2 drops to 1/3. Hence, v_1 and v_2 might be not independent in the composite system $(\mathsf{Bat}_1 \parallel \mathsf{Bat}_2) \parallel \mathsf{Dev}$.

Motivated by this example, we consider hybrid systems that combine discrete behaviors and continuous dynamics. In this setting, the most prominent modeling formalism are hybrid automata, which comprise a control graph with discrete jumps between (control) locations and flows that model the evolution of continuous variables over time. When time passes in a hybrid system, a flow starting from the current variable evaluation is selected non-deterministically and then the variables evolve according to the chosen flow. Besides the stochastic independence, additional aspects are relevant for the composition of hybrid systems. Let us assume that α_1 is a (local) action of Bat_1 which cannot be observed by Bat_2 or Dev. Particularly, α_1 does not affect the value of variable v_2. The hybrid automaton $\mathsf{Bat}_1 \parallel \mathsf{Bat}_2$ has states of the form $\langle s^1, s^2 \rangle$. Suppose $\langle s_1^1, s_1^2 \rangle \to^{t_1} \langle s_2^1, s_2^2 \rangle \to^{\alpha_1} \langle s_3^1, s_3^2 \rangle \to^{t_2} \langle s_4^1, s_4^2 \rangle$ is a finite path in $\mathsf{Bat}_1 \parallel \mathsf{Bat}_2$, comprising two timed transitions with time passages t_1 and t_2 and one jump transition involving action α_1. As α_1 cannot be observed by Bat_2, we expect $s_1^2 \to^{t_1+t_2} s_4^2$ in Bat_2. In particular, a faithful model for the composite system would allow for selecting a flow for v_1 within time passage t_1, which is continued within the subsequent time passage. Thus, the adaption of the flow for variable v_2 should only be possible when executing an action involving Bat_2 or Dev. This aspect is also crucial in the context of modeling controller strategies for hybrid systems. Typically, control decisions are made at distinct points and fixed until a next control decision is enabled. For instance, when considering a traffic alert and collision avoidance systems on aircraft, the advise of a corrective maneuver is determined when a critical situation occurs and fixed until sensor values exceed a threshold that indicates changes of the situation. A crucial point is to identify exactly those situations where adaptation of flows is allowed and required, as from a practical point of view it is important to minimize costs of adaptation and to keep the complexity of controllers manageable.

Contribution. We introduce a generic composition operator for stochastic transition systems (STSs) [16] based on spans and span couplings. Our operator does not rely on the assumption that the STSs to be composed are stochastically independent and covers standard composition

operators by dealing with specific spans. Spans provide a formal approach for introducing a universal notion of coupling probability measures. We develop an extensive theory for spans exploiting profound results known from descriptive set theory [34]. Based on a standard notion of bisimulation, we provide a congruence result with respect to our span composition. In the second part of the paper, we instantiate our general approach and introduce stochastic hybrid motion automata (SHMA) in which the progressing flow is recorded within states. We present a compositional framework for SHMA including an STS-semantics, a composition operator that does not rely on the assumption of stochastic independence, and where the adjustment of flows is always accompanied with an action. We show that the congruence result for STSs transfers to our SHMA framework.

Additional material and detailed proofs can be found in the technical report [25].

Related Work. We are not aware of a compositional modeling approach of stochastic systems which does not rely on the assumption that the components to be composed are stochastically independent. Our work thus addresses a fundamental challenge in the context of probabilistic operational models. The recent work [28] gives a comprehensive overview on compositional probabilistic modeling formalisms regarding expressive power and available analysis techniques. The concept of compositionality has its roots in the theory of process calculi [37, 32] and there are many fundamental contributions in the field of stochastic extensions of process calculi and probabilistic automata [39, 1, 2, 18, 13, 19]. Results on discrete systems have been extended to formalisms with continuous state spaces [16, 35]. The theory on non-deterministic labeled Markov processes (NLMPs) provide elegant notions and results on bisimulation and its logical characterization [21, 22, 17, 20, 6, 31]. Unfortunately, NLMPs are a priori not appropriate for our purposes as the class of NLMPs is not closed under the composition of stochastic transition systems [25]: Given two NLMPs, the transition function of their composition does not need to be measurable. When considering real-time systems, an important distinguishing aspect is the notion of residence time, which is the time spend in a state before moving to a successor state. In prominent compositional frameworks, timing behavior is modeled by clocks (timed automata) [4, 9, 38, 11] or one has exponential-distributed holding times (Markov automata and interactive Markov chains) [30, 23]. A general theory on compositionality and behavioral equivalences has been also achieved for probabilistic real-time systems modeled by interactive generalized semi-Markov processes [14, 12]. When adding flows to specify the evolution of continuous variables between jumps, one enters the field of hybrid systems [3, 29, 10]. The spirit of our work concerning hybrid systems is closest to the compositional frameworks developed for hybrid extensions of I/O-automata [36] and reactive modules [5] in the non-stochastic case. [36] studies parallel composition, simulation relations, and the receptiveness property and deals with prefix-, suffix- and concatenation-closed sets of flows on a syntactic level to obtain time-transitivity. Probabilistic hybrid automata [40, 26] extend classical hybrid automata by discrete probabilistic updates for the jumps. In [24, 27, 26], stochastic hybrid automata are considered where variables can be updated according to continuous distributions. Different from these hybrid automata, the change of flows in SHMA is only possible when some action is executed. Stochastic flows, i.e., where stochastic choices can be made continuously over time, are considered in [15, 33]. Our framework does not incorporate this kind of flows so far.

2 Preliminaries

We suppose the reader is familiar with standard concepts from measure and probability theory [8]. We briefly summarize our notations used throughout this paper.

Couplings. Within our work we understand couplings as a "modeling tool". Intuitively, couplings relate given measures in a product space by a measure with corresponding marginals. $\mathsf{Prob}(X)$ denotes the set of all probability measures on the measurable space X. Let X_1 and X_2 be measurable spaces. Given $\mu_1 \in \mathsf{Prob}(X_1)$ and $\mu_2 \in \mathsf{Prob}(X_2)$, $\mu \in \mathsf{Prob}(X_1 \times X_2)$ is called a *coupling of* (μ_1, μ_2) if $\mu(M_1 \times X_2) = \mu_1(M_1)$ and $\mu(X_1 \times M_2) = \mu_2(M_2)$ for all measurable $M_1 \subseteq X_1$ and $M_2 \subseteq X_2$. The *independent coupling of* (μ_1, μ_2) is the *product measure* of μ_1 and μ_2 denoted by $\mu_1 \otimes \mu_2$. If $\mu_1 = \mathsf{Dirac}[x_1]$ for some $x_1 \in X_1$, then there is exactly one coupling of (μ_1, μ_2), namely the independent one. Here, $\mathsf{Dirac}[x_1]$ denotes the probability measure where for all measurable $M_1 \subseteq X_1$, $\mathsf{Dirac}[x_1](M_1) = 1$ iff $x_1 \in M_1$.

Polish spaces. A separable and completely metrizable topological space is called a *Polish space* [34]. If X is a Polish space, then $\mathsf{Prob}(X)$ is well equipped with the topology induced by the weak convergence of probability measures. To obtain a measurable space, Polish spaces are equipped with the Borel sigma algebra, i.e., the coarsest sigma-algebra where all open sets are measurable. We call a measurable space X *standard Borel* if there exists a Polish topology on X where the induced Borel sigma-algebra coincides with the given one. The Polish topology is in general not uniquely determined. We refer to measurable subsets of standard Borel spaces as *Borel sets*. Of course, every Polish space is standard Borel.

Functions for probability measures. Given a measurable function $f\colon X_1 \to X_2$ between measurable spaces X_1 and X_2, the *pushforward of* f is defined by $f_\sharp\colon \mathsf{Prob}(X_1) \to \mathsf{Prob}(X_2)$, $f_\sharp(\mu)(M_2) = \mu(f^{-1}(M_1))$. Assuming Polish spaces X_1 and X_2, a *Markov kernel* is a Borel function $k\colon X_1 \to \mathsf{Prob}(X_2)$. Here, for every $\mu_1 \in \mathsf{Prob}(X_1)$ we define *semi-product measure* $\mu_1 \rtimes k \in \mathsf{Prob}(X_1 \times X_2)$, $\mu_1 \rtimes k(M_1 \times M_2) = \int_{M_1} k(x_1)(M_2)\, d\mu_1(x_1)$.

Relations. Let $R \subseteq X_1 \times X_2$ be a binary relation over some sets X_1 and X_2. We usually write $x_1\, R\, x_2$ instead of $\langle x_1, x_2\rangle \in R$. Then, R is called *lr-total in* $X_1 \times X_2$ if for all $x_1 \in X_1$ there exists $x_2 \in X_2$ such that $x_1\, R\, x_2$ and vice versa, i.e., also for all $x_2 \in X_2$ there exists $x_1 \in X_1$ where $x_1\, R\, x_2$. Assume X_1 and X_2 constitute measurable spaces and let $\mu_1 \in \mathsf{Prob}(X_1)$ and $\mu_2 \in \mathsf{Prob}(X_2)$. A *weight function for* (μ_1, R, μ_2) is a coupling W of (μ_1, μ_2) such that $x_1\, R\, x_2$ for W-almost all $\langle x_1, x_2\rangle \in X_1 \times X_2$. We write $\mu_1\, R^w\, \mu_2$ if there exists a weight function for (μ_1, R, μ_2). Notice, R^w constitutes a relation in $\mathsf{Prob}(X_1) \times \mathsf{Prob}(X_2)$. Notice, weight functions are also well-established in the discrete setting [39].

Variables. Let Var denote a countable set of variables and $V \subseteq \mathsf{Var}$. We denote by $\mathsf{Ev}(V)$ the set of all *variable evaluations for* V, i.e., functions from V to $\mathbb{R}$. As the countable product of Polish spaces equipped with the product topology again yields a Polish space, $\mathsf{Ev}(V)$ constitutes a Polish space. Let $e \in \mathsf{Ev}(\mathsf{Var})$ and $\eta \in \mathsf{Prob}(\mathsf{Ev}(\mathsf{Var}))$. The projection $e_{|V} \in \mathsf{Ev}(V)$ is given by $e_{|V}(v) = e(v)$ for all $v \in V$. As $f\colon \mathsf{Ev}(\mathsf{Var}) \to \mathsf{Ev}(V)$, $f(e) = e_{|V}$ is measurable, we can safely define $\eta_{|V} = f_\sharp(\eta)$. $\mathsf{Cond}(\mathsf{Var})$ denotes the set of all Boolean conditions over Var and we write $e \models c$ if the variable evaluation e satisfies condition c. For instance, $e \models (v \leq 3.14159) \wedge (v \geq 2.71828)$ iff $e(v) \leq 3.14159$ and $e(v) \geq 2.71828$.

Stochastic transition systems. An *STS* is a triple $\mathcal{T} = (S, \Gamma, \to)$ comprising a measurable space S of *states*, a set Γ of *labels*, and a relation $\to \subseteq S \times \Gamma \times \mathsf{Prob}(S)$ of *transitions*. If S is a standard Borel space, then $\mathcal{T}$ is called *standard Borel*. Let $\mathcal{T}_a = (S_a, \Gamma, \to_a)$ and $\mathcal{T}_b = (S_b, \Gamma, \to_b)$ be STSs with the same sets of labels. A relation $R \subseteq S_a \times S_b$ is a *bisimulation for* $(\mathcal{T}_a, \mathcal{T}_b)$ if R is lr-total in $S_a \times S_b$ and for all $s_a\, R\, s_b$ and $\gamma \in \Gamma$ it holds:

Given $\mu_a \in \mathsf{Prob}(S_a)$ where $s_a \to_a^\gamma \mu_a$, then there exists $\mu_b \in \mathsf{Prob}(S_b)$ such that $s_b \to_b^\gamma \mu_b$ and $\mu_a \, R^w \, \mu_b$. Vice versa, given $\mu_b \in \mathsf{Prob}(S_b)$ with $s_b \to_b^\gamma \mu_b$, then there is $\mu_a \in \mathsf{Prob}(S_a)$ where $s_a \to_a^\gamma \mu_a$ and $\mu_a \, R^w \, \mu_b$. We emphasize that a bisimulation is not required to be measurable. In the context of bisimulation an important question is how to lift a relation $R \subseteq S_a \times S_b$ to probability measures. However, there are other approaches using R-stable pairs instead [22], closely related to the weight lifting [41, 39]. Given STSs $\mathcal{T}_1 = (S_1, \Gamma_1, \to_1)$ and $\mathcal{T}_2 = (S_2, \Gamma_2, \to_2)$ and a set of synchronization labels $\mathsf{Sync} \subseteq \Gamma_1 \cap \Gamma_2$, their composition is the STS $\mathcal{T}_1 \parallel_{\mathsf{Sync}}^{\otimes} \mathcal{T}_2 = (S_1 \times S_2, \Gamma_1 \cup \Gamma_2, \to)$ with $\langle s_1, s_2 \rangle \to^\gamma \mu_1 \otimes \mu_2$ iff the following holds [16]: If $\gamma \in \Gamma_1 \setminus \mathsf{Sync}$, then $s_1 \to^\gamma \mu_1$ and $\mu_2 = \mathsf{Dirac}[s_2]$. If $\gamma \in \Gamma_2 \setminus \mathsf{Sync}$, then $\mu_1 = \mathsf{Dirac}[s_1]$ and $s_2 \to^\gamma \mu_2$. If $\gamma \in \mathsf{Sync}$, then $s_1 \to^\gamma \mu_1$ and $s_2 \to^\gamma \mu_2$.

Flows. By $\mathbb{T} = \mathbb{R}_{\geq 0}$ we denote the *time axis*. A *flow* is a function $\vartheta \colon \mathbb{T} \to \mathsf{Ev}(\mathsf{Var})$ that has the càdlàg property, i.e., ϑ is right continuous and has left limits everywhere. $\mathsf{Flow}(\mathsf{Var})$ denotes the set of all flows. Let $\vartheta \oplus T(t) = \vartheta(T+t)$ denote the *shift of* ϑ *at time* $T \in \mathbb{T}$ *by time* $t \in \mathbb{T}$. A subset F of $\mathsf{Flow}(\mathsf{Var})$ is *shift invariant* if $\vartheta \oplus T \in F$ for every $\vartheta \in F$ and $T \in \mathbb{T}$. In the theory of stochastic processes, the càdlàg property is well established as, amongst others, there is a topology on $\mathsf{Flow}(\mathsf{Var})$ such that $\mathsf{Flow}(\mathsf{Var})$ becomes a Polish space [7]. The exact definition of this topology is not relevant for our purposes. If $V \subseteq \mathsf{Var}$ and $\vartheta \in \mathsf{Flow}(\mathsf{Var})$, then $\vartheta_{|V} \in \mathsf{Flow}(V)$ is given by $\vartheta_{|V}(t) = \vartheta(t)_{|V}$ for all $t \in \mathbb{T}$. Given $V_1, V_2 \subseteq \mathsf{Var}$ where $V_1 \cap V_2 = \varnothing$ and $\vartheta_1 \in \mathsf{Flow}(V_1)$ and $\vartheta_2 \in \mathsf{Flow}(\mathsf{Var}_2)$, then $\vartheta_1 \uplus \vartheta_2 \in \mathsf{Flow}(V_1 \cup V_2)$ is the flow obtained by merging ϑ_1 and ϑ_2.

3 Composition of stochastic transition systems

We develop our approach towards the composition of STSs. As a preparation, we introduce spans first and give some insights on our mathematical theory for those. After that, we present the main contribution of the paper, namely our composition operator for STSs. We then give a congruence theorem having a quite challenging proof. Section 4 presents an application of our framework in the context of stochastic hybrid systems.

3.1 Spans

We will formalize dependencies for the composition of STSs using spans and span couplings, which is a generic and flexible formalism our framework benefits from in many occasions. The idea is to allow for arbitrary Polish spaces together with continuous functions that specify the relationships between the components. Various properties of spans then transfer to their probabilistic version, e.g., properness or the existence of inverses. This is an essential point in the context of stochastic models and hence also for STSs. We will then use spans within the definition of our composition in STS and later on also in the context of stochastic hybrid systems as a mathematical tool for our argumentation.

▶ **Definition 1.** A *span* is a tuple $\mathcal{X} = (X, X_1, X_2, \iota_1, \iota_2)$ consisting of Polish spaces X, X_1, and X_2 and continuous functions $\iota_1 \colon X \to X_1$ and $\iota_2 \colon X \to X_2$. We call $\mathcal{X}$ *proper*, if $\iota_1^{-1}(K_1) \cap \iota_2^{-1}(K_2)$ is compact in X for all compact sets $K_1 \subseteq X_1$ and $K_2 \subseteq X_2$.

Intuitively, X denotes the joint state space of X_1 and X_2, where ι_1 and ι_2 are projective functions from X to X_1 and X_2, respectively. Properness connects topological aspects of the involved spaces. The following examples are natural instances of proper spans:

- $\mathcal{X}$ is a *Cartesian span* if $X = X_1 \times X_2$ and ι_1 and ι_2 are the natural projections.

- $\mathcal{X}$ is a *variable span* if $X_1 = \mathsf{Ev}(\mathsf{Var}_1)$, $X_2 = \mathsf{Ev}(\mathsf{Var}_2)$, and $X = \mathsf{Ev}(\mathsf{Var}_1 \cup \mathsf{Var}_2)$ for some sets of variables Var_1 and Var_2, and ι_1 and ι_2 are the natural projections.
- $\mathcal{X}$ is a *identity span* if $X = X_1 = X_2$ and $\iota_1(x) = x$ and $\iota_2(x) = x$ for all $x \in X$.

Span couplings are a crucial notion for our approach towards a composition operator in the next section. Given $\mu_1 \in \mathsf{Prob}(X_1)$ and $\mu_2 \in \mathsf{Prob}(X_2)$, we call $\mu \in \mathsf{Prob}(X)$ a $\mathcal{X}$*-coupling of* (μ_1, μ_2) if $(\iota_1)_\sharp(\mu) = \mu_1$ and $(\iota_2)_\sharp(\mu) = \mu_2$. Recall that $(\iota_1)_\sharp$ and $(\iota_2)_\sharp$ denote the pushforwards of ι_1 and ι_2, respectively. A span coupling places two probability measures in the same probabilistic space specified by the span by exhibiting an adequate witness measure over pairs. Thus, the ordinary notion for couplings is generalized. For all x and μ we use $x_{|1}$, $x_{|2}$, $\mu_{|1}$, and $\mu_{|2}$ as shorthand notations for $\iota_1(x)$, $\iota_2(x)$, $(\iota_1)_\sharp(\mu)$, and $(\iota_2)_\sharp(\mu)$ respectively. Given $x_1 \in X_1$ and $x_2 \in X_2$, we then write $x_1 \,\mathcal{X}\, x_2$ if there exists $x \in X$ where $x_{|1} = x_1$ and $x_{|2} = x_2$. Similarly, we write $\mu_1 \,\mathcal{X}^c\, \mu_2$ if there is a $\mathcal{X}$-coupling of (μ_1, μ_2). We sometimes drop the projection functions from the tuple and refer to (X, X_1, X_2) as a span.

Probabilistic version. There are various operations for spans that yield complex spans out of some given basic spans. The question whether the operation preserves properness is important for practical purposes. For instance, using Tychonoff's theorem, a countable product of proper spans yields a proper span again. Within stochastic models, the following operation is important: For a span $\mathcal{X} = (X, X_1, X_2, \iota_1, \iota_2)$ its *probabilistic version* is given by the tuple $\mathsf{Prob}(\mathcal{X}) = (\mathsf{Prob}(X), \mathsf{Prob}(X_1), \mathsf{Prob}(X_2), (\iota_1)_\sharp, (\iota_2)_\sharp)$. Notice, $\mathsf{Prob}(\mathcal{X})$ involves all $\mathcal{X}$-couplings and $\mu_1 \,\mathcal{X}^c\, \mu_2$ iff $\mu_1 \,\mathsf{Prob}(\mathcal{X})\, \mu_2$ for all $\mu_1 \in \mathsf{Prob}(X_1)$ and $\mu_2 \in \mathsf{Prob}(X_2)$.

▶ **Proposition 2.** *The probabilistic version of a span is a span. Moreover, the probabilistic version of a proper span is proper as well.*

The claim regarding properness follows from Prokhorov's theorem [7], which characterizes relatively compact subsets of $\mathsf{Prob}(X)$: If $P \subseteq \mathsf{Prob}(X)$ is a set of probability measures, then P is relatively compact in $\mathsf{Prob}(X)$ iff P is tight in $\mathsf{Prob}(X)$, i.e., for every $\varepsilon \in \mathbb{R}_{>0}$ there is a compact set $K \subseteq X$ where $\mu(K) > 1 - \varepsilon$ for all $\mu \in P$.

Span inverse. In a compositional setting, the states of the components determine the states of the composed system. Within our approach, a state as an element of X in the composed system is not required to be uniquely determined: Given a span $\mathcal{X} = (X, X_1, X_2)$, $x_1 \in X_1$, and $x_2 \in X_2$, every $x \in X$ where $x_{|1} = x_1$ and $x_{|2} = x_2$ stands for a state in the composed system resulting from the states x_1 and x_2 of the components. However, in applications later it is important to have a mapping with additional properties: Given a span $\mathcal{X} = (X, X_1, X_2)$, a Borel function $f \colon X_1 \times X_2 \to X$ is called an $\mathcal{X}$*-inverse*, if for all $x_1 \in X_1$ and $x_2 \in X_2$, if $x_1 \,\mathcal{X}\, x_2$, then $f(x_1, x_2)_{|1} = x_1$ and $f(x_1, x_2)_{|2} = x_2$.

▶ **Theorem 3.** *Every proper span $\mathcal{X}$ has an $\mathcal{X}$-inverse.*

It follows $\mu_1 \,\mathcal{X}^c\, \mu_2$ iff $\mu_1 \,\mathsf{Rel}(\mathcal{X})^w\, \mu_2$ for all $\mu_1 \in \mathsf{Prob}(X_1)$ and $\mu_2 \in \mathsf{Prob}(X_2)$, where $\mathsf{Rel}(\mathcal{X}) = \{\langle x_{|1}, x_{|2}\rangle \,;\, x \in X\}$. Our proof of Theorem 3 is an application of a *measurable selection theorem* [8]: Take some $\hat{x} \in X$ and define $\Phi \colon X_1 \times X_2 \to 2^X$, $\Phi(x_1, x_2) = \{x \in X \,;\, x_{|1} = x_1 \text{ and } x_{|2} = x_2\}$, if the set on the right-hand side is non-empty, and $\Phi(x_1, x_2) = \{\hat{x}\}$, otherwise. It suffices to argue that Φ admits a measurable selection, i.e., there is a measurable function $f \colon X_1 \times X_2 \to X$ where $f(x_1, x_2) \in \Phi(x_1, x_2)$ for all $x_1 \in X_1$ and $x_2 \in X_2$. To do so, we rely on results from descriptive set theory. Notice, together with Proposition 2, Theorem 3 yields an $\mathsf{Prob}(\mathcal{X})$-inverse if $\mathcal{X}$ is proper, which is an important observation for our discussions later. This is not obvious even for simple spans considering

for instance the probabilistic version of a variable span. We remark that there are spans $\mathcal{X}$ that have no $\mathcal{X}$-inverses and thus, the properness assumption is important [25].

3.2 Composition

A major objective in defining compositional frameworks is to separate the concerns of components specifying the operational behavior and composition operators addressing their interaction or coordination. We start with two STSs $\mathcal{T}_1 = (S_1, \Gamma_1, \to_1)$ and $\mathcal{T}_2 = (S_2, \Gamma_2, \to_2)$, where we assume S_1 and S_2 are Polish spaces. To declare the interactions between $\mathcal{T}_1$ and $\mathcal{T}_2$, we specify a set of synchronization labels $\mathsf{Sync} \subseteq \Gamma_1 \cap \Gamma_2$, a span $\mathcal{S} = (S, S_1, S_2)$ to characterizes the state space of the composition, and an so-called agreement $\mathcal{G} = (LC_1, LC_2)$ between $\mathcal{T}_1$ and $\mathcal{T}_2$. Here, LC_1 and LC_2 are so-called local constraints and for the moment, to present the central definition of this paper, it suffices to require $LC_1, LC_2 \subseteq S \times \mathsf{Prob}(S)$. Intuitively, we use local constraints to specify the behavior of local variables within local transitions (see below).

▶ **Definition 4.** We define the STS $\mathcal{T}_1 \parallel_{\mathcal{S},\mathcal{G},\mathsf{Sync}} \mathcal{T}_2 = (S, \Gamma_1 \cup \Gamma_2, \to)$, where for all $s \in S$, $\gamma \in \Gamma$, and $\mu \in \mathsf{Prob}(S)$ it holds $s \to^\gamma \mu$ iff one of the following three conditions hold:

- $\gamma \in \Gamma_1 \setminus \mathsf{Sync}$ and $s_{|1} \to_1^\gamma \mu_{|1}$ and $s \, LC_2 \, \mu$.
- $\gamma \in \Gamma_2 \setminus \mathsf{Sync}$ and $s \, LC_1 \, \mu$ and $s_{|2} \to_2^\gamma \mu_{|2}$.
- $\gamma \in \mathsf{Sync}$ and $s_{|1} \to_1^\gamma \mu_{|1}$ and $s_{|2} \to_2^\gamma \mu_{|2}$.

To illustrate the crux of our composition operator, we regard the case where $\mathcal{S}$ is a Cartesian span, i.e., $S = S_1 \times S_2$. Former approaches [39, 16] assume that $\mathcal{T}_1$ and $\mathcal{T}_2$ behave stochastically independent in a synchronizing step, i.e., if $s_{|1} \to_1^\gamma \mu_1$ and $s_{|2} \to_2^\gamma \mu_2$, then $s \hookrightarrow^\gamma \mu_1 \otimes \mu_2$ in $\mathcal{T}_1 \parallel_H^\otimes \mathcal{T}_2$. Our operator does not rely on any stochastic assumptions: Instead of considering only the independent coupling we take all the couplings into account, i.e., if $s_{|1} \to_1^\gamma \mu_1$ and $s_{|2} \to_2^\gamma \mu_2$, then $s \to^\gamma \mu$ for all couplings μ of (μ_1, μ_2). During a discussion about the example from the introduction and SHMAs, we will see how additional stochastic information between the components can be incorporated within our general framework.

Local constraints. Our composition operator is indexed by a span, which determines the dependencies between the states of $\mathcal{T}_1$ and $\mathcal{T}_2$. For instance, one can specify shared and local variables using the variable span. When composing STSs, one has to ensure that local transitions and variables of the components behave in a compatible way. Let us illustrate this and regard again the case where $\mathcal{S}$ is a Cartesian span. If $\mathcal{T}_1$ performs a local transition, i.e., a transition that is labeled by some $\gamma \in \Gamma_1 \setminus \mathsf{Sync}$, then the current state of $\mathcal{T}_2$ must not change. The properties of a local constraint should hence guarantee $s \, LC_2 \, \mu$ iff $\mu_{|2} = \mathsf{Dirac}[s_{|2}]$. It then follows that $\langle s_1, s_2 \rangle \to^\gamma \mu_1 \otimes \mathsf{Dirac}[s_2]$ for all $s_2 \in S_2$ and $s_1 \to_1^\gamma \mu_1$ where $\gamma \in \Gamma_1 \setminus \mathsf{Sync}$. Of course, the same discussion applies for $\mathcal{T}_1$ and the local constraint LC_1. This leads to the following requirements for a local constraint $LC_2 \subseteq S \times \mathsf{Prob}(S)$:

- For all $s \in S$ and $\mu \in \mathsf{Prob}(S)$, if $\mu_{|2} = \mathsf{Dirac}[s_{|2}]$, then $s \, LC_2 \, \mu$.
- For all $s \, LC_2 \, \mu$ and $\mu' \in \mathsf{Prob}(S)$, if $\mu_{|1} = \mu'_{|1}$ and $\mu_{|2} = \mu'_{|2}$, then $s \, LC_2 \, \mu'$.
- For all $s \, LC_2 \, \mu$, if $\mu_{|1} \, \mathcal{S}^c \, \mathsf{Dirac}[s_{|2}]$, then μ is a $\mathcal{S}$-coupling of $(\mu_{|1}, \mathsf{Dirac}[s_{|2}])$.

The requirements for LC_1 are similar. Intuitively, the first requirement for LC_2 ensures that the STS $\mathcal{T}_2$ cannot block a local transition of $\mathcal{T}_1$ which is not critical from the view of $\mathcal{T}_2$, i.e., variables of $\mathcal{T}_2$ are not affected within the transition of $\mathcal{T}_1$. Thus, such local transition of $\mathcal{T}_1$ are independent of $\mathcal{T}_2$ and can happen autonomously. Different couplings of given probability measures cannot be distinguished within local constraints imposed by the second property.

The third requirement intuitively demands that whenever $\mathcal{T}_1$ performs a local transition where no local variables of $\mathcal{T}_2$ are modified, the state of $\mathcal{T}_2$ must not change. In case of a Cartesian span the above requirements yield

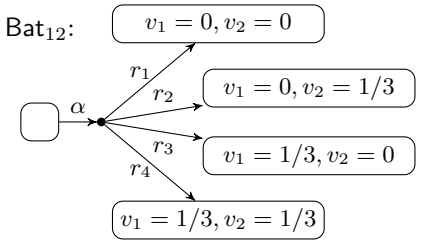
$$LC_2 = \{\langle\langle s_1, s_2\rangle, \mu_1 \otimes \mathsf{Dirac}[s_2]\rangle \,;\, s_1 \in S_1 \text{ and } s_2 \in S_2 \text{ and } \mu_1 \in \mathsf{Prob}(S_1)\} \quad \text{and}$$
$$LC_1 = \{\langle\langle s_1, s_2\rangle, \mathsf{Dirac}[s_1] \otimes \mu_2\rangle \,;\, s_1 \in S_1 \text{ and } s_2 \in S_2 \text{ and } \mu_2 \in \mathsf{Prob}(S_2)\}.$$

Hence, the agreement $\mathcal{G}$ is uniquely determined by STSs $\mathcal{T}_1$ and $\mathcal{T}_2$. We thus simply write $\mathcal{T}_1 \parallel_{\times,\mathsf{Sync}} \mathcal{T}_2$ instead of $\mathcal{T}_1 \parallel_{\mathcal{S},\mathcal{G},\mathsf{Sync}} \mathcal{T}_2$. Observe that $\mathcal{T}_1 \parallel_{\times,\mathsf{Sync}} \mathcal{T}_2$ and $\mathcal{T}_1 \parallel^{\otimes}_{\mathsf{Sync}} \mathcal{T}_2$ are not bisimilar in general. This is due to the fact that our composition operator does not incorporate any stochastic assumptions concerning the interaction of $\mathcal{T}_1$ and $\mathcal{T}_2$. In case where $\mathcal{S}$ is a variable span, i.e., $S_1 = \mathsf{Ev}(\mathsf{Var}_1)$, $S_2 = \mathsf{Ev}(\mathsf{Var}_2)$, and $S = \mathsf{Ev}(\mathsf{Var}_1 \cup \mathsf{Var}_2)$ for some sets of variables Var_1 and Var_2, there are more possible local constraints:

$$LC_2' = \{\langle e, \eta\rangle \in S \times \mathsf{Prob}(S) \,;\, \eta_{|\mathsf{Var}_1} = \mathsf{Dirac}[e_{|\mathsf{Var}_1}] \text{ implies } \eta = \mathsf{Dirac}[e]\},$$
$$LC_2'' = \{\langle e, \eta\rangle \in S \times \mathsf{Prob}(S) \,;\, \eta_{|\mathsf{Var}_2 \setminus \mathsf{Var}_1} = \mathsf{Dirac}[e_{|\mathsf{Var}_2 \setminus \mathsf{Var}_1}]\}, \quad \text{and}$$
$$LC_2''' = \{\langle e, \eta\rangle \in S \times \mathsf{Prob}(S) \,;\, \eta_{|\mathsf{Var}_2} = \mathsf{Dirac}[e_{|\mathsf{Var}_2}]\}$$

all enjoy the requirements for a local constraint where $LC_2' \supseteq LC_2'' \supseteq LC_2'''$. Considering for instance LC_2'', all the variables in $\mathsf{Var}_2 \setminus \mathsf{Var}_1$ cannot be modified within a local transition of $\mathcal{T}_1$. Constraint LC_2''' is more restrictive: Here, all the variables in Var_2 are controlled by $\mathcal{T}_2$ and cannot be modified in a local transition of $\mathcal{T}_1$, i.e., variables in $\mathsf{Var}_1 \cap \mathsf{Var}_2$ can be observed by $\mathcal{T}_1$ only. It turns out $LC_2' \supseteq LC_2$ for every local constraint LC_2. Every local constraint hence enjoys the property that variables in $\mathsf{Var}_2 \setminus \mathsf{Var}_1$ must not be adapted within a local transition of $\mathcal{T}_1$ if the evaluations of the variables in Var_1 remain the same.

Example from the introduction. We return to the introductory stochastic systems illustrated in Section 1. Of course, Bat_1, Bat_2, and Dev can be seen as STSs with sets of states $\mathsf{Ev}(\{v_1\})$, $\mathsf{Ev}(\{v_2\})$, and $\mathsf{Ev}(\{v_1, v_2\})$, respectively. When composing them, we need not to worry about local constraints as there is only one synchronization action α. In what follows, we rely on the obvious variable spans. The composition of Bat_1 and Bat_2 yields the STS Bat_{12} depicted below.



There are infinitely many transitions: Every solution of the linear equation system $r_1 + r_2 = r_1 + r_3 = r_2 + r_4 = r_3 + r_4 = 1/2$ where $r_1, r_2, r_3, r_4 \in [0, 1]$ represents a coupling of the involved measures. When composing Bat_{12} and Dev, the set of all couplings is refined. We are moreover able to handle more complex stochastic information that depend on the operational behavior of the components. To illustrate this, assume systems which result from Bat_1 and Bat_2 such that α can be executed repeatedly (e.g., add some local transitions back to the blank state). An additional component might encode that, if the system has crashed repeatedly in the past, the event that the stored energy drops to 0 in both batteries

at the same time becomes more likely within an execution of α. We emphasize that the ordinary composition of STSs can be expressed within our framework using an additional component [25].

3.3 Congruence

In the context of process calculi, an important issue of bisimulation is the compatibility with syntactic operators in the process calculus, such as parallel composition. We show that bisimulation is a congruence for our composition operator under reasonable side-constraints, i.e., our composition operator enjoys the substitution property with respect to bisimulation. Suppose STSs $\mathcal{T}_{a1} = (S_{a1}, \Gamma_1, \to_{a1})$, $\mathcal{T}_{a2} = (S_{a2}, \Gamma_2, \to_{a2})$, $\mathcal{T}_{b1} = (S_{b1}, \Gamma_1, \to_{b1})$, and $\mathcal{T}_{b2} = (S_{b2}, \Gamma_2, \to_{b2})$ such that $\mathcal{T}_{a1} \sim \mathcal{T}_{b1}$ and $\mathcal{T}_{a2} \sim \mathcal{T}_{b2}$. Define

$$\mathcal{T}_a = \mathcal{T}_{a1} \parallel_{\mathcal{S}_a, \mathcal{G}_a, \mathsf{Sync}} \mathcal{T}_{a2} \quad \text{and} \quad \mathcal{T}_b = \mathcal{T}_{b1} \parallel_{\mathcal{S}_b, \mathcal{G}_b, \mathsf{Sync}} \mathcal{T}_{b2},$$

where $\mathsf{Sync} \subseteq \Gamma_1 \cap \Gamma_2$, $\mathcal{S}_a = (S_a, S_{a1}, S_{a2})$ and $\mathcal{S}_b = (S_b, S_{b1}, S_{b2})$ are proper spans, and $\mathcal{G}_a = (LC_{a1}, LC_{a2})$ and $\mathcal{G}_b = (LC_{b1}, LC_{b2})$ are agreements. Assume R_1 is a bisimulation for $(\mathcal{T}_{a1}, \mathcal{T}_{b1})$ and R_2 is a bisimulation for $(\mathcal{T}_{a2}, \mathcal{T}_{b2})$ and define

$$R_1 \wedge R_2 = \{\langle s_a, s_b \rangle \in S_a \times S_b \,;\, s_{a|1}\, R_1\, s_{b|1} \text{ and } s_{a|2}\, R_2\, s_{b|2}\}.$$

We aim to show that $R_1 \wedge R_2$ is a bisimulation for $(\mathcal{T}_a, \mathcal{T}_b)$ and hence $\mathcal{T}_a \sim \mathcal{T}_b$. However, we cannot expect this result without any compatibility requirements for the involved spans and agreements, since important relationships concerning the communication of the components are determined within our composition operator. This motivates the following notions: We refer to the tuple $\mathcal{C} = (\mathcal{S}_a, \mathcal{S}_b, R_1, R_2)$ as *span connection* and call $\mathcal{C}$ *adequate* if for all $\mu_{a1}\, R_1^w\, \mu_{b1}$ and $\mu_{a2}\, R_2^w\, \mu_{b2}$ it holds $\mu_{a1}\, \mathcal{S}_a^c\, \mu_{a2}$ iff $\mu_{b1}\, \mathcal{S}_b^c\, \mu_{b2}$. Intuitively, adequacy requires that the existence of span couplings is preserved by the relations R_1 and R_2. Observe, if $\mathcal{S}_a$ and $\mathcal{S}_b$ are Cartesian spans, then $\mathcal{C}$ is always adequate. The local constraints LC_{a2} and LC_{b2} are called *$\mathcal{C}$-bisimilar* if for all $s_a\, (R_1 \wedge R_2)\, s_b$ holds:

- For all $\mu_a \in \mathsf{Prob}(S_a)$ and $\mu_{b1} \in \mathsf{Prob}(S_{b1})$, if $s_a\, LC_{a2}\, \mu_a$ and $\mu_{a|1}\, R_1^w\, \mu_{b1}$, then there is $\mu_b \in \mathsf{Prob}(S_b)$ where $s_b\, LC_{b2}\, \mu_b$, $\mu_{b|1} = \mu_{b1}$, and $\mu_{a|2}\, R_2^w\, \mu_{b|2}$.
- For all $\mu_b \in \mathsf{Prob}(S_b)$ and $\mu_{a1} \in \mathsf{Prob}(S_{a1})$, if $s_b\, LC_{b2}\, \mu_b$ and $\mu_{a1}\, R_1^w\, \mu_{b|1}$, then there is $\mu_a \in \mathsf{Prob}(S_a)$ where $s_a\, LC_{a2}\, \mu_a$, $\mu_{a|1} = \mu_{a1}$, and $\mu_{a|2}\, R_2^w\, \mu_{b|2}$.

LC_{a1} and LC_{b1} are called *$\mathcal{C}$-bisimilar* if analogous properties are fulfilled. Observe that the stated requirement is motivated by the definition of bisimulation in the sense that each element of a local constraint LC_{a2} can be mimicked by LC_{b2} regarding the relations R_1 and R_2. If LC_{a2} and LC_{b2} as well as LC_{a1} and LC_{b1} are $\mathcal{C}$-bisimilar, respectively, then we refer to $\mathcal{G}_a$ and $\mathcal{G}_b$ as *$\mathcal{C}$-bisimilar*.

▶ **Theorem 5.** *If the span connection $\mathcal{C}$ is adequate and the agreements $\mathcal{G}_a$ and $\mathcal{G}_b$ are $\mathcal{C}$-bisimilar, then $R_1 \wedge R_2$ is a bisimulation for $(\mathcal{T}_a, \mathcal{T}_b)$.*

The challenging part of the proof can be summarized by the following claim [25]: Let $\mu_a \in \mathsf{Prob}(S_a)$, $\mu_{b1} \in \mathsf{Prob}(S_{b1})$, and $\mu_{b2} \in \mathsf{Prob}(S_{b2})$ where $\mu_{a|1}\, R_1^w\, \mu_{b1}$ and $\mu_{a|2}\, R_2^w\, \mu_{b2}$. Then there is an $\mathcal{S}_b$-coupling μ_b of (μ_{b1}, μ_{b2}) such that $\mu_a\, (R_1 \wedge R_2)^w\, \mu_b$. Our proof of this claim proceeds as follows. Assume W_1 is a weight function for $(\mu_{a|1}, R_1, \mu_{b1})$ and W_2 is a weight function for $(\mu_{a|2}, R_2, \mu_{b2})$. Using disintegration of measures [34], there are Markov kernels $k_1 \colon S_{a1} \to \mathsf{Prob}(S_{a2})$ and $k_2 \colon S_{b1} \to \mathsf{Prob}(S_{b2})$ such that $W_1 = \mu_{a|1} \rtimes k_1$ and $W_2 = \mu_{a|2} \rtimes k_2$. The crucial point is now to argue that there is a Markov kernel $k \colon S_a \to \mathsf{Prob}(S_b)$ where $k(s_a)$

is an $\mathcal{S}_b$-coupling of $(k_1(s_{a|1}), k_2(s_{a|2}))$ for μ_a-almost all $s_a \in S_a$. Here, we make use of an $\mathcal{S}_b$-inverse (cf. Theorem 3). With this Markov kernel at hand, we define $W \in \mathsf{Prob}(S_a \times S_b)$ by $W = \mu \rtimes k$ and $\mu_b \in \mathsf{Prob}(S_b)$ by $\mu_b(M_b) = W(S_a \times M_b)$. It turns out that μ_b is an appropriate $\mathcal{S}_b$-coupling. To summarize, we defined a potential weight function W out of the weight functions W_1 and W_2 and then introduced the measure μ_b via W.

Path measures. When resolving the non-determinism in STSs using schedulers, one obtains a probability measure – the path measure – on the set of all infinite paths of the STS [16]. Besides our congruence result, we expect compatibility of path measures induced by schedulers in our compositional framework. To provide an intuition, assume STSs $\mathcal{T}_1$ and $\mathcal{T}_2$ and let $\mathcal{T}$ be an STSs obtained by a composition involving $\mathcal{T}_1$ and $\mathcal{T}_2$. Assume that $\mathfrak{S}_1$ and $\mathfrak{S}_2$ are schedulers for $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively, and $\mathfrak{S}$ is a scheduler for $\mathcal{T}$. If $\mathfrak{S}$ satisfies certain compatibility requirements regarding $\mathfrak{S}_1$ and $\mathfrak{S}_2$, one can show that the induced path measure for $\mathcal{T}$ is a coupling of the corresponding path measures for $\mathcal{T}_1$ and $\mathcal{T}_2$. Here, we consider a natural span that connects the sets of all infinite paths of $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}$.

4 Stochastic hybrid motion automata

We apply our general results of the preceding sections and develop a compositional modeling framework for stochastic hybrid systems. The formal definition of our model relies on a standard schema of hybrid automata [3, 29, 26], i.e., there are a discrete control structure consisting of locations and jumps in-between, and continuous variables whose values evolve according to a flow formalized by a motion function. Within a jump, the variables can be updated instantaneously. The novelty of our approach is that every jump is indexed by a set of those variables that are not affected in the corresponding discrete step. As a consequence, the adjustment of flows is always accompanied by a specific command.

Syntax. Every jump in our hybrid-automaton model is labeled by a command: Given a set Var of variables and a set Act of actions, a *command on* $(\mathsf{Var}, \mathsf{Act})$ is a tuple $\langle c, \alpha, V, \mathsf{upd}\rangle$ consisting of a *guard* $c \in \mathsf{Cond}(\mathsf{Var})$, an *action* $\alpha \in \mathsf{Act}$, a set of *disabled* variables $V \subseteq \mathsf{Var}$, and an *(non-deterministic) update* $\mathsf{upd}\colon \mathsf{Ev}(\mathsf{Var}) \to 2^{\mathsf{Prob}(\mathsf{Ev}(\mathsf{Var}))}$ where $\eta_{|V} = \mathsf{Dirac}[e_{|V}]$ for all $\eta \in \mathsf{upd}(e)$ and $e \in \mathsf{Ev}(\mathsf{Var})$. $\mathsf{Cmd}(\mathsf{Var}, \mathsf{Act})$ denotes the set of all commands on $(\mathsf{Var}, \mathsf{Act})$. Intuitively, a jump is enabled if the current variable evaluation satisfies the guard. The action name indicates whether the jump is an internal location switch or subject to an interaction with another component. The set of disabled variables specifies those variables which are not affected within the jump. This also clarifies the additional requirement for updates.

▶ **Definition 6.** An *SHMA* is a tuple $(\mathsf{Loc}, \mathsf{Var}, \mathsf{Act}, \mathsf{Inv}, \mathsf{Mot}, \twoheadrightarrow)$ where Loc is a finite set of *locations*, Var is a set of variables, Act is a set of actions, $\mathsf{Inv}\colon \mathsf{Loc} \to \mathsf{Cond}(\mathsf{Var})$ is an *invariant function*, $\mathsf{Mot}\colon \mathsf{Loc} \to 2^{\mathsf{Flow}(\mathsf{Var})}$ is a *motion function* which assigns a shift-invariant set of flows to every location, and $\twoheadrightarrow \subseteq \mathsf{Loc} \times \mathsf{Cmd}(\mathsf{Var}, \mathsf{Act}) \times \mathsf{Prob}(\mathsf{Loc})$ is a *jump relation*.

We write $l \mathrel{-\!\![} \mathit{cmd} \mathrel{]\!\!\rightarrow\!\!\!\!\!\rightarrow} \lambda$ instead of $\langle l, \mathit{cmd}, \lambda\rangle \in \twoheadrightarrow$. The behavior in a location l depends on the current variable evaluation e. In a discrete step, a jump $l \mathrel{-\!\![} c, \alpha, V, \mathsf{upd} \mathrel{]\!\!\rightarrow\!\!\!\!\!\rightarrow} \lambda$ where $e \models c$ is chosen non-deterministically. Then, action α is executed and a successor location is sampled according to λ. The evaluation of the variables changes according to a non-deterministically chosen probability measure contained in $\mathsf{upd}(e)$. Entering a location l', a flow in $\mathsf{Mot}(l')$ is also chosen non-deterministically and the variables then evolve according to this flow.

Semantics. Every SHMA $\mathcal{H} = (\mathsf{Loc}, \mathsf{Var}, \mathsf{Act}, \mathsf{Inv}, \mathsf{Mot}, \twoheadrightarrow)$ can be interpreted as an STS resulting from unfolding. In what follows, $S = \mathsf{Loc} \times \mathsf{Flow}(\mathsf{Var})$ denotes the set of states. Notice, S constitutes a Polish space as $\mathsf{Flow}(\mathsf{Var})$ is known to be a Polish space [7]. Intuitively, a state $\langle l, \vartheta \rangle$ represents the actual location l and the current active flow ϑ, i.e., ϑ corresponds to the flow chosen in the preceding jump. Moreover, $\vartheta(0)$ stands for the present variable evaluation. We call $\langle l, \vartheta \rangle$ *well-formed* if $\vartheta \in \mathsf{Mot}(l)$ and $\vartheta(0) \models \mathsf{Inv}(l)$.

There are two kinds of transitions within our STS for $\mathcal{H}$, namely transitions where time passes and transitions corresponding to a jump. Time can pass in a location l as long as the flow does not violate the invariant $\mathsf{Inv}(l)$. Transitions for jumps are more intricate. Assume $l \multimap[\, c, \alpha, V, \mathsf{upd} \,]\twoheadrightarrow \lambda$ is enabled in state $\langle l, \vartheta \rangle$, i.e., $e \models c$ where $e = \vartheta(0)$. Basically, jumps in SHMAs proceed in two phases: First, a successor location and a variable evaluation are sampled according to λ and some $\eta \in \mathsf{upd}(e)$, respectively. In the second phase, a flow is chosen non-deterministically for those variables which are not disabled, i.e., the variables in $\mathsf{Var} \setminus V$. This is formalized as follows: A *flow adapter for* (ϑ, V) is a Borel function $\chi \colon \mathsf{Loc} \times \mathsf{Ev}(\mathsf{Var}) \to \mathsf{Flow}(\mathsf{Var})$ such that for all $l' \in \mathsf{Loc}$ and $e', \tilde{e}' \in \mathsf{Ev}(\mathsf{Var})$:

$$\chi(l', e')_{|V} = \vartheta_{|V} \quad \text{and} \quad e'_{|\mathsf{Var}\setminus V} = \tilde{e}'_{|\mathsf{Var}\setminus V} \text{ implies } \chi(l', e')_{|\mathsf{Var}\setminus V} = \chi(l', \tilde{e}')_{|\mathsf{Var}\setminus V}.$$

Intuitively, if state $\langle l', e' \rangle$ is sampled within the first phase of a jump, then $\chi(l', e')$ represents the new flow, i.e., the flow which determines the evolution of variables in a subsequent time passage. The first condition for a flow adapter requires that the flow for disabled variables is not allowed to change. The required implication ensures that a flow is chosen independently of the disabled variables. This is important for our compositional approach, as we want to make sure that the choice of a new flow in an SHMA obtained by composition does not depend on the local variables of the respective communication partners. If χ is a flow adapter, then we define the auxiliary function $\hat{\chi} \colon \mathsf{Loc} \times \mathsf{Ev}(\mathsf{Var}) \to S$, $\hat{\chi}(l, e) = \langle l, \chi(l, e) \rangle$.

▶ **Definition 7.** The *semantics of* $\mathcal{H}$ is given by the STS $[\![\mathcal{H}]\!] = (S, \mathbb{T} \cup \mathsf{Act}, \to)$, where $\to$ is the smallest relation satisfying the following requirements for all well-formed states $s = \langle l, \vartheta \rangle$:

- For all $T \in \mathbb{T}$, if $\vartheta(t) \models \mathsf{Inv}(l)$ for every $t \in [0, T]$, then $s \to^t \mathsf{Dirac}[\langle l, \vartheta \oplus T \rangle]$.
- For all $l \multimap[\, c, \alpha, V, \mathsf{upd} \,]\twoheadrightarrow \lambda$, $\eta \in \mathsf{upd}(e)$, couplings ν of (λ, η), and flow adapter χ for (ϑ, V), if $e \models c$ and for ν-almost all $\langle l', e' \rangle \in \mathsf{Loc} \times \mathsf{Ev}(\mathsf{Var})$ the state $\hat{\chi}(l', e')$ is well-formed, then $s \to^\alpha \hat{\chi}_\sharp(\nu)$. Here, we abbreviate $e = \vartheta(0)$.

An SHMA almost surely enters a well-formed state, i.e., if $s \to^\gamma \mu$ where $\gamma \in \mathbb{T} \cup \mathsf{Act}$, then s' is well-formed for μ-almost all $s' \in S$. We emphasize that for our approach concerning the adaption of flows it is crucial that the current flow is part of a state. Otherwise, it would be not possible to ensure that the flow for disabled variables is not allowed to change.

Composition. We now introduce a composition operator for SHMAs. For $i \in \{1, 2\}$ let $\mathcal{H}_i = (\mathsf{Loc}_i, \mathsf{Var}_i, \mathsf{Act}_i, \mathsf{Inv}_i, \mathsf{Mot}_i, \twoheadrightarrow_i)$ be SHMAs. When running $\mathcal{H}_1$ and $\mathcal{H}_2$ in parallel, $\mathcal{H}_1$ and $\mathcal{H}_2$ synchronize on all actions contained in $\mathsf{Act}_1 \cap \mathsf{Act}_2$ and the variables in $\mathsf{Var}_1 \cap \mathsf{Var}_2$ are shared, i.e., $\mathsf{Var}_1 \setminus \mathsf{Var}_2$ and $\mathsf{Var}_2 \setminus \mathsf{Var}_1$ represent the sets of the respective local variables. Abbreviate $\mathsf{Loc} = \mathsf{Loc}_1 \times \mathsf{Loc}_2$, $\mathsf{Var} = \mathsf{Var}_1 \cup \mathsf{Var}_2$, and $\mathsf{Act} = \mathsf{Act}_1 \cup \mathsf{Act}_2$. Let upd_1 and upd_2 be updates for Var_1 and Var_2, respectively. The Var-*lifting of* $(\mathsf{upd}_1, \mathsf{upd}_2)$ is the update upd for Var such that for all $e \in \mathsf{Ev}(\mathsf{Var})$, $\mathsf{upd}(e)$ consists of all $\eta \in \mathsf{Prob}(\mathsf{Ev}(\mathsf{Var}))$ where $\eta_{|\mathsf{Var}_1} = \eta_1$ and $\eta_{|\mathsf{Var}_2} = \eta_2$ for some $\eta_1 \in \mathsf{upd}(e_{|\mathsf{Var}_1})$ and $\eta_2 \in \mathsf{upd}(e_{|\mathsf{Var}_2})$. We define Var-liftings with respect to an update accordingly, i.e., upd is a Var-lifting of upd_1 if for all $e \in \mathsf{Ev}(\mathsf{Var})$, $\mathsf{upd}(e)$ consists of all $\eta \in \mathsf{Prob}(\mathsf{Ev}(\mathsf{Var}))$ where $\eta_{|\mathsf{Var}_1} = \eta_1$ for some $\eta_1 \in \mathsf{upd}(e_{|\mathsf{Var}_1})$ and $\eta_{|\mathsf{Var}\setminus\mathsf{Var}_1} = \mathsf{Dirac}[e_{|\mathsf{Var}\setminus\mathsf{Var}_1}]$. Notice, the definition of Var-liftings involves couplings concerning a variable span, which provides a connection to the preceding sections.

▶ **Definition 8.** $\mathcal{H}_1 \parallel \mathcal{H}_2 = (\mathsf{Loc}, \mathsf{Var}, \mathsf{Act}, \mathsf{Inv}, \mathsf{Mot}, \twoheadrightarrow)$ is the SHMA with $\mathsf{Inv}(l_1, l_2) = \mathsf{Inv}(l_1) \wedge \mathsf{Inv}(l_2)$ and $\mathsf{Mot}(l_1, l_2) = \{\vartheta \in \mathsf{Flow}(\mathsf{Var}) \,;\, \vartheta_{|\mathsf{Var}_1} \in \mathsf{Mot}_1(l_1) \text{ and } \vartheta_{|\mathsf{Var}_2} \in \mathsf{Mot}_2(l_2)\}$ for all $\langle l_1, l_2\rangle \in \mathsf{Loc}$ and $\twoheadrightarrow$ is the smallest relation such that $\langle l_1, l_2\rangle \dashv[\, c, \alpha, V, \mathsf{upd} \,]\!\!\rightarrow\!\!\!\!\twoheadrightarrow \lambda$, if λ is a coupling of $\lambda_1 \in \mathsf{Prob}(\mathsf{Loc}_1)$ and $\lambda_2 \in \mathsf{Prob}(\mathsf{Loc}_2)$ and one of the following holds:

- $\alpha \in \mathsf{Act}_1 \setminus \mathsf{Act}_2$, $\lambda_2 = \mathsf{Dirac}[l_2]$, and there is $l_1 \dashv[\, c_1, \alpha, V_1, \mathsf{upd}_1 \,]\!\!\rightarrow\!\!\!\!\twoheadrightarrow_1 \lambda_1$ such that $c = c_1$, $V = V_1 \cup (\mathsf{Var}_2 \setminus \mathsf{Var}_1)$, and upd is the Var-lifting of upd_1.
- $\alpha \in \mathsf{Act}_2 \setminus \mathsf{Act}_1$, $\lambda_1 = \mathsf{Dirac}[l_1]$, and there is $l_2 \dashv[\, c_2, \alpha, V_2, \mathsf{upd}_2 \,]\!\!\rightarrow\!\!\!\!\twoheadrightarrow_2 \lambda_2$ such that $c = c_2$, $V = V_2 \cup (\mathsf{Var}_1 \setminus \mathsf{Var}_2)$, and upd is the Var-lifting of upd_2.
- $\alpha \in \mathsf{Act}_1 \cap \mathsf{Act}_2$ and there are $l_1 \dashv[\, c_1, \alpha, V_1, \mathsf{upd}_1 \,]\!\!\rightarrow\!\!\!\!\twoheadrightarrow_1 \lambda_1$ and $l_2 \dashv[\, c_2, \alpha, V_2, \mathsf{upd}_2 \,]\!\!\rightarrow\!\!\!\!\twoheadrightarrow_2 \lambda_2$ where $c = c_1 \wedge c_2$, $V = V_1 \cup V_2$, and upd is the Var-lifting of $(\mathsf{upd}_1, \mathsf{upd}_2)$.

When composing SHMAs, local variables of participating SHMAs become disabled for corresponding internal jumps. Within our semantics, flow adapters thus ensure that the adaption of flows in internal jumps in $\mathcal{H}_1 \parallel \mathcal{H}_2$ are independent of the local variables of the respective communication partners. Moreover, flows for local variables of $\mathcal{H}_2$ cannot be adapted within an internal jump of $\mathcal{H}_1$ and vice versa. It is easy to see that the composition operator for SHMAs is commutative and associative.

Congruence. We aim for a congruence theorem for SHMAs relying on Theorem 5. For this, we relate the composition of SHMAs with our general approach towards a composition of STSs, i.e., we represent the STS $[\![\mathcal{H}_1 \parallel \mathcal{H}_1]\!]$ as a composition involving the components $[\![\mathcal{H}_1]\!]$ and $[\![\mathcal{H}_2]\!]$. Notice that sampling a successor location in $\mathcal{H}_1 \parallel \mathcal{H}_2$ happens according to a coupling measure. This observation also applies when combining measures for locations and variable evaluations within our semantics of SHMAs. To this end, it is easy to define the corresponding span $\mathcal{S}$ and agreement $\mathcal{G}$ such that

$$[\![\mathcal{H}_1 \parallel \mathcal{H}_2]\!] = [\![\mathcal{H}_1]\!] \parallel_{\mathcal{S},\mathcal{G},\mathsf{Act}_1 \cap \mathsf{Act}_2} [\![\mathcal{H}_2]\!].$$

More precisely, $\mathcal{S}$ is a span arising from a Cartesian span for the locations and a span for the sets of flows. For the agreement $\mathcal{G}$, we regard local constraints where the shared variables can be modified by both involved systems $\mathcal{H}_1$ and $\mathcal{H}_2$. The obtained representation of $[\![\mathcal{H}_1 \parallel \mathcal{H}_2]\!]$ underpins again the flexibility of our composition operator for STS.

We rephrase Theorem 5 in the context SHMAs. Two SHMAs are bisimilar if their semantics in terms of STSs are bisimilar. Let $\mathcal{H}_{a1}$ and $\mathcal{H}_{b1}$ be SHMAs with the same sets of variables Var_1 and actions Act_1 and similar, let $\mathcal{H}_{a2}$ and $\mathcal{H}_{b2}$ be SHMAs with variables Var_2 and actions Act_2. Abbreviate $\mathsf{LVar}_1 = \mathsf{Var}_1 \setminus \mathsf{Var}_2$, $\mathsf{LVar}_2 = \mathsf{Var}_2 \setminus \mathsf{Var}_1$, and $\mathsf{SVar} = \mathsf{Var}_1 \cap \mathsf{Var}_2$.

▶ **Theorem 9.** *Let R_1 and R_2 be bisimulations for $(\mathcal{H}_{a1}, \mathcal{H}_{b1})$ and $(\mathcal{H}_{a2}, \mathcal{H}_{b2})$, respectively. $\mathcal{H}_{a1} \parallel \mathcal{H}_{a2}$ and $\mathcal{H}_{b1} \parallel \mathcal{H}_{b2}$ are bisimilar if R_1 and R_2 do not involve shared variables, i.e.,*

$$\begin{aligned} R_1 = \{\langle\langle l_{a1}, \vartheta_{a1|\mathsf{LVar}_1} \uplus \vartheta^S\rangle, \langle l_{b1}, \vartheta_{b1|\mathsf{LVar}_1} \uplus \vartheta^S\rangle\rangle \,;\, \\ \langle l_{a1}, \vartheta_{a1}\rangle \, R_1 \, \langle l_{b1}, \vartheta_{b1}\rangle \text{ and } \vartheta^S \in \mathsf{Flow}(\mathsf{SVar})\}, \\ R_2 = \{\langle\langle l_{a2}, \vartheta_{a2|\mathsf{LVar}_2} \uplus \vartheta^S\rangle, \langle l_{b2}, \vartheta_{b2|\mathsf{LVar}_2} \uplus \vartheta^S\rangle\rangle \,;\, \\ \langle l_{a2}, \vartheta_{a2}\rangle \, R_2 \, \langle l_{b2}, \vartheta_{b2}\rangle \text{ and } \vartheta^S \in \mathsf{Flow}(\mathsf{SVar})\}. \end{aligned}$$

Our requirement that R_1 and R_2 do not distinguish between shared variables yields the compatibility assumption required for Theorem 5. Our proof then simply exploits the representation of $[\![\mathcal{H}_{a1} \parallel \mathcal{H}_{a2}]\!]$ and $[\![\mathcal{H}_{b1} \parallel \mathcal{H}_{b2}]\!]$ in terms of a composition of STSs.

5 Concluding remarks

In this paper, we introduced a generic parallel-composition operator for STSs and SHMAs. The essential new feature that distinguishes the novel composition from previous ones is that it uses the mathematical concepts of spans and couplings to model the effect of composing (potentially dependent) stochastic behaviors. The latter is crucial for systems where the components communicate via shared variables. A further feature of the novel stochastic-hybrid-system model (SHMA) is that the adaption of flows depends on commands rather happening on arbitrary occasions. We proved important algebraic properties in the context of composition, e.g., congruence with respect to bisimulation. This shows that even within our generic operator one does not have to forgo desired properties of compositional systems. There is plenty room for further elaborations. Firstly, we are going to develop a mathematical theory for SHMA that also involves stochastic flows. Furthermore, we will work on a modeling language for couplings and spans in order to obtain a theoretical basis for practical tools. Also other kinds of models, where spans yield a powerful approach for compositional modeling, could be investigated. Moreover, our approach concerning couplings as a modeling formalism enables many new verification questions, e.g., for directly reasoning about the coordination between components.

References

1 L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, University of Stanford, 1997.

2 L. de Alfaro. Stochastic transition systems. In *9th International Conference on Concurrency Theory (CONCUR)*, LNCS 1446, pages 423–438. Springer, 1998.

3 R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

4 R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

5 R. Alur and T. A. Henzinger. Modularity for timed and hybrid systems. In *8th International Conference on Concurrency Theory (CONCUR)*, LNCS 1243, pages 74–88. Springer, 1997.

6 G. Bacci, G. Bacci, K. G. Larsen, and R. Mardare. Bisimulation on Markov processes over arbitrary measurable spaces. In *Horizons of the Mind. A Tribute to P. Panangaden*, LNCS 8464, pages 76–95. Springer, 2014.

7 P. Billingsley. *Convergence of Probability Measures*. Wiley-Interscience, 2 edition, 1999.

8 V. I. Bogachev. *Measure Theory Volume*, volume 1 and 2. Springer, 2007.

9 H. Bohnenkamp, P. R. D'Argenio, H. Hermanns, and J.-P. Katoen. MoDeST: A compositional modeling formalism for real-time and stochastic systems. *IEEE Transactions on Software Engineering*, 32(10):812–830, 2006.

10 S. Bornot and J. Sifakis. On the composition of hybrid systems. In *1st International Workshop on Hybrid Systems: Computation and Control (HSCC)*, volume 1386 of *Lecture Notes in Computer Science*, pages 49–63. Springer, 1998.

11 P. Bouyer, T. Brihaye, P. Carlier, and Q. Menet. Compositional design of stochastic timed automata. In *11th International Computer Science Symposium in Russia (CSR)*, LNCS, 2016 (to appear).

12 M. Bravetti. Real time and stochastic time. In *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM-RT)*, LNCS 3185, pages 132–180. Springer, 2004.

13 M. Bravetti and P. R. D'Argenio. Tutte le algebre insieme: Concepts, discussions and relations of stochastic process algebras with general distributions. In *Validation of Stochastic Systems – A Guide to Current Research*, LNCS 2925, pages 44–88. Springer, 2004.

14 M. Bravetti and R. Gorrieri. The theory of interactive generalized semi-Markov processes. *Theoretical Computer Science*, 282(1):5–32, 2002.

15 M. L. Bujorianu and John Lygeros. Towards a general theory of stochastic hybrid systems. In *Stochastic Hybrid Systems*, volume 337 of *Lecture Notes in Control and Information Science*, pages 3–30. Springer, 2006.

16 S. Cattani, R. Segala, M. Z. Kwiatkowska, and G. Norman. Stochastic transition systems for continuous state spaces and non-determinism. In *8th International Conference on Foundations of Software Science and Computational Structures (FOSSACS)*, LNCS 3441, pages 125–139. Springer, 2005.

17 V. Danos, J. Desharnais, F. Laviolette, and P. Panangaden. Bisimulation and cocongruence for probabilistic systems. *Information and Computation*, 204(4):503–523, 2006.

18 P. R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, University of Twente, 1999.

19 P. R. D'Argenio and J.-P. Katoen. A theory of stochastic systems part I: Stochastic automata and part II: Process algebra. *Information and Computation*, 203(1):1–74, 2005.

20 P. R. D'Argenio, P. Sánchez Terraf, and N. Wolovick. Bisimulations for non-deterministic labelled Markov processes. *Mathematical Structures in Computer Science*, 22:43–68, 2012.

21 E. P. de Vink and J.J.M.M. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. *Theoretical Computer Science*, 221:271–293, 1999.

22 J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labelled Markov processes. *Information and Computation*, 179(2):163–193, 2002.

23 C. Eisentraut, H. Hermanns, and L. Zhang. On probabilistic automata in continuous time. In *25th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 342–351. IEEE Computer Society, 2010.

24 M. Fränzle, E. M. Hahn, H. Hermanns, N. Wolovick, and L. Zhang. Measurability and safety verification for stochastic hybrid systems. In *14th International Conference on Hybrid Systems: Computation and Control (HSCC)*, pages 43–52. ACM, 2011.

25 D. Gburek, C. Baier, and S. Klüppelholz. Composition of stochastic transition systems based on spans and couplings. Technical report, Technische Universität Dresden, 2016. URL: `http://wwwtcs.inf.tu-dresden.de/ALGI/PUB/ICALP16/`.

26 E. M. Hahn. *Model checking stochastic hybrid systems*. PhD thesis, Universität des Saarlandes, 2013.

27 E. M. Hahn, A. Hartmanns, H. Hermanns, and J.-P. Katoen. A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods in System Design*, 2012.

28 A. Hartmanns and H. Hermanns. In the quantitative automata zoo. *Science of Computer Programming*, 112:3–23, 2015.

29 T. A. Henzinger. The theory of hybrid automata. In *11th Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 278–292. IEEE Computer Society, 1996.

30 H. Hermanns. *Interactive Markov Chains: And the Quest for Quantified Quality*. Springer, 2002.

31 H. Hermanns, J. Krcál, and J. Kretínský. Probabilistic bisimulation: Naturally on distributions. In *25th International Conference on Concurrency Theory (CONCUR)*, LNCS 8704, pages 249–265. Springer, 2014.

32 C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.

33 J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In *12th International Conference on Hybrid Systems: Computation and Control (HSCC)*, LNCS 1790, pages 160–173. Springer, 2000.

34 A. S. Kechris. *Classical Descriptive Set Theory*, volume 156 of *Graduate Texts in Mathematics*. Springer, 1995.

35 H. Kerstan and B. König. Coalgebraic trace semantics for probabilistic transition systems based on measure theory. In *23th International Conference on Concurrency Theory (CONCUR)*, LNCS 7454, pages 410–424. Springer, 2012.

36 N. Lynch, R. Segala, and F. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, 2003.

37 R. Milner. *A Calculus of Communicating Systems*. Springer, 1982.

38 S. Mitra and N. Lynch. Trace-based semantics for probabilistic timed I/O automata. In *12th International Conference on Hybrid Systems: Computation and Control (HSCC)*, LNCS 4416, pages 718–722. Springer, 2007.

39 R. Segala. *Modeling and Verification of Randomized Distributed Real-Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

40 J. Sproston. *Model Checking for Probabilistic Timed and Hybrid Systems*. PhD thesis, University of Birmingham, 2001.

41 V. Strassen. The existence of probability measures with given marginals. *The Annals of Mathematical Statistics*, 36(2):423–439, 1965.

On Restricted Nonnegative Matrix Factorization*†

Dmitry Chistikov‡[1], Stefan Kiefer§[2], Ines Marušić[3], Mahsa Shirmohammadi[4], and James Worrell[5]

1 **Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern and Saarbrücken, Germany¶**
dch@mpi-sws.org
2 **University of Oxford, Oxford, United Kingdom**
Stefan.Kiefer@cs.ox.ac.uk
3 **University of Oxford, Oxford, United Kingdom**
Ines.Marusic@cs.ox.ac.uk
4 **University of Oxford, Oxford, United Kingdom**
Mahsa.Shirmohammadi@cs.ox.ac.uk
5 **University of Oxford, Oxford, United Kingdom**
James.Worrell@cs.ox.ac.uk

Abstract

Nonnegative matrix factorization (NMF) is the problem of decomposing a given nonnegative $n \times m$ matrix M into a product of a nonnegative $n \times d$ matrix W and a nonnegative $d \times m$ matrix H. Restricted NMF requires in addition that the column spaces of M and W coincide. Finding the minimal inner dimension d is known to be NP-hard, both for NMF and restricted NMF. We show that restricted NMF is closely related to a question about the nature of minimal probabilistic automata, posed by Paz in his seminal 1971 textbook. We use this connection to answer Paz's question negatively, thus falsifying a positive answer claimed in 1974.

Furthermore, we investigate whether a rational matrix M always has a restricted NMF of minimal inner dimension whose factors W and H are also rational. We show that this holds for matrices M of rank at most 3 and we exhibit a rank-4 matrix for which W and H require irrational entries.

1998 ACM Subject Classification F.1.1 Models of Computation, F.2.1 Numerical Algorithms and Problems

Keywords and phrases nonnegative matrix factorization, nonnegative rank, probabilistic automata, labelled Markov chains, minimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.103

1 Introduction

Nonnegative matrix factorization (NMF) is the task of factoring a matrix of nonnegative real numbers M (henceforth a *nonnegative* matrix) as a product $M = W \cdot H$ such that matrices W and H are also nonnegative. The smallest inner dimension of any such factorization is called the *nonnegative rank* of M, written $\mathrm{rank}_+(M)$.

* A full version of the paper is available at http://arxiv.org/abs/1605.07061.
† Kiefer, Marušić, Shirmohammadi, and Worrell gratefully acknowledge the support of the EPSRC.
‡ Chistikov was sponsored in part by the ERC Synergy award ImPACT.
§ Kiefer is supported by a University Research Fellowship of the Royal Society.
¶ Present address: Department of Computer Science, University of Oxford, UK.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 103; pp. 103:1–103:14

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

In machine learning, NMF was popularized by the seminal work of Lee and Seung [14] as a tool for finding features in facial-image databases. Since then, NMF has found a broad range of applications – including document clustering, topic modelling, computer vision, recommender systems, bioinformatics, and acoustic signal processing [5, 4, 7, 19, 21, 22]. In applications, matrix M can typically be seen as a matrix of data points: each column of M corresponds to a data point and each row to a feature. Then, computing a nonnegative factorization $M = W \cdot H$ corresponds to expressing the data points (columns of M) as convex combinations of latent factors (columns of W), i.e., as linear combinations of latent factors with nonnegative coefficients (columns of H).

From a computational perspective, perhaps the most basic problem concerning NMF is whether a given nonnegative matrix of rational numbers M admits an NMF with inner dimension at most a given number k. Formally, the *NMF problem* asks whether $\text{rank}_+(M) \leq k$. In practical applications, various heuristics and local-search algorithms are used to compute an approximate nonnegative factorization, but little is known in terms of their theoretical guarantees. The NMF problem under the *separability* assumption of Donoho and Stodden [9] is tractable: an NMF $M = W \cdot H$ is called *separable* if every column of W is also a column of M. In 2012, Arora et al. [2] showed that it is decidable in polynomial time whether a given matrix admits a separable NMF with a given inner dimension. Further progress was made recently, with several efficient algorithms for computing near-separable NMFs [13, 12].

Vavasis [20] showed that the problem of deciding whether the rank of a nonnegative matrix is equal to its nonnegative rank is NP-hard. This result implies that generalizations of this problem, such as the aforementioned NMF problem, the problem of computing the factors W, H (in both exact and approximate versions), and nonnegative rank determination, are also NP-hard. It is not known whether any of these problems are in NP.

Vavasis [20] notes that the difficulty in proving membership in NP lies in the fact that a certificate for a positive answer to the NMF problem seems to require the sought factors: a pair of nonnegative matrices W, H such that $M = W \cdot H$. Related to this, Cohen and Rothblum [8] posed the question of whether, given a nonnegative matrix of rational numbers M, there always exists an NMF $M = W \cdot H$ of inner dimension equal to $\text{rank}_+(M)$ such that both W and H are also matrices of rational numbers. A natural route to proving membership of the NMF problem in NP would be to give a positive answer to the question of Cohen and Rothblum (as well as a polynomial bound on the bit-length of the factors W and H). However, the question remains open. Currently the best complexity bound for the NMF problem is membership in PSPACE, which is obtained by translation into the existential theory of real-closed fields [2]. Such a translation shows that one can always choose the entries of W and H to be algebraic numbers.

In this work, we focus on the so-called *restricted* NMF (RNMF) problem, introduced by Gillis and Glineur [11]. The RNMF problem is defined as the NMF problem, except that the column spaces of M and W are required to coincide. (Note that for any NMF, the column space of M is a subspace of the column space of W.) This problem has a natural geometric interpretation as the *nested polytope problem (NPP)*: the problem of finding a minimum-vertex polytope nested between two given convex polytopes. In more detail, for a rank-r matrix M, finding an RNMF with inner dimension d is known to correspond exactly to finding a nested polytope with d vertices in an $(r-1)$-dimensional NPP.

Our contributions are as follows.

1. We establish a tight connection between NMF and the coverability relation in labelled Markov chains (LMCs). The latter notion was introduced by Paz [15]. Loosely speaking, an LMC $\mathcal{M}'$ *covers* an LMC $\mathcal{M}$ if for any initial distribution over the states of $\mathcal{M}$ there is

an initial distribution over the states of $\mathcal{M}'$ such that $\mathcal{M}$ and $\mathcal{M}'$ are equivalent. In 1971, Paz [15] asked a question about the nature of minimal covering LMCs. The question was supposedly answered positively in 1974 [3]. However, we show that the correct answer is negative, thus falsifying the claim in [3]. Instrumental to our counterexample is the observation that restricted nonnegative rank and nonnegative rank can be different. (Indeed, the wrong claims in [3] seem to implicitly rely on the opposite assumption, although the notions of NMF and RNMF had not yet been developed.)

2. We show that the RNMF problem for matrices M of rank 3 or less can be solved in polynomial time. In fact, we show that there is always a *rational* NMF of M with inner dimension $\mathrm{rank}_+(M)$, and that it can be computed in polynomial time in the Turing model of computation. This improves a result in [11] where the RNMF problem is shown to be solvable in polynomial time assuming a RAM model with unit-cost arithmetic. Both our algorithm and the one in [11] exploit the connection to the 2-dimensional NPP, allowing us to take advantage of a geometric algorithm by Aggarwal et al. [1]. We need to adapt the algorithm in [1] to ensure that the occurring numbers are rational and can be computed in polynomial time in the Turing model of computation.
3. We exhibit a rank-4 matrix that has an RNMF with inner dimension 5 but *no rational* RNMF with inner dimension 5. We construct this matrix via a particular instance of the 3-dimensional NPP, again taking advantage of the geometric interpretation of RNMF. Our result answers the RNMF variant of Cohen and Rothblum's question in [8] negatively. The original (NMF) variant remains open.

Detailed proofs of all results can be found in the full version of this paper.

2 Nonnegative Matrix Factorization

Let $\mathbb{N}$ and $\mathbb{N}_0$ denote the set of all positive and nonnegative integers, respectively. For every $n \in \mathbb{N}$, we write $[n]$ for the set $\{1, 2, \ldots, n\}$ and write I_n for the identity matrix of order n. For any ordered field $\mathbb{F}$, we denote by $\mathbb{F}_+$ the set of all its nonnegative elements. For any vector v, we write v_i for its i^{th} entry. A vector v is called *stochastic* if its entries are nonnegative real numbers that sum up to one. For every $i \in [n]$, we write e_i for the i^{th} coordinate vector in $\mathbb{R}^n$. We write $\mathbf{1}^{(n)}$ for the n-dimensional column vector with all ones. We omit the superscript if it is understood from the context.

For any matrix M, we write M_i for its i^{th} row, M^j for its j^{th} column, and $M_{i,j}$ for its $(i,j)^{\text{th}}$ entry. The *column space* (resp., *row space*) of M, written $\mathrm{Col}(M)$ (resp., $\mathrm{Row}(M)$), is the vector space spanned by the columns (resp., rows) of M. A matrix is called *nonnegative* (resp., *zero* or *rational*) if so are all its entries. A nonnegative matrix is *column-stochastic* (resp., *row-stochastic*) if the element sum of each of its columns (resp., rows) is one.

2.1 Nonnegative Rank

Let $\mathbb{F}$ be an ordered field, such as the reals $\mathbb{R}$ or the rationals $\mathbb{Q}$. Given a nonnegative matrix $M \in \mathbb{F}_+^{n\times m}$, a *nonnegative matrix factorization (NMF) over* $\mathbb{F}$ of M is any representation of the form $M = W \cdot H$ where $W \in \mathbb{F}_+^{n\times d}$ and $H \in \mathbb{F}_+^{d\times m}$ are nonnegative matrices. Note that $\mathrm{Col}(M) \subseteq \mathrm{Col}(W)$. We refer to d as the *inner dimension* of the NMF, and hence refer to NMF $M = W \cdot H$ as being *d-dimensional*. The *nonnegative rank over* $\mathbb{F}$ of M is the smallest number $d \in \mathbb{N}_0$ such that there exists a d-dimensional NMF over $\mathbb{F}$ of M. An equivalent characterization [8] of the nonnegative rank over $\mathbb{F}$ of M is as the smallest number of rank-1 matrices in $\mathbb{F}_+^{n\times m}$ such that M is equal to their sum. The nonnegative rank over $\mathbb{R}$ will

henceforth simply be called nonnegative rank, and will be denoted by $\text{rank}_+(M)$. For any nonnegative matrix $M \in \mathbb{R}_+^{n\times m}$, it is easy to see that $\text{rank}(M) \leq \text{rank}_+(M) \leq \min\{n, m\}$.

Given a nonzero matrix $M \in \mathbb{F}_+^{n\times m}$, by removing the zero columns of M and dividing each remaining column by the sum of its elements, we obtain a column-stochastic matrix M' with equal nonnegative rank. Similarly, if $M = W \cdot H$ then after removing zero columns in W and multiplying with a suitable diagonal matrix D, we get $M = W \cdot H = WD \cdot D^{-1}H$ where WD is column-stochastic. If M is column-stochastic then $\mathbf{1}^\top = \mathbf{1}^\top M = \mathbf{1}^\top WD \cdot D^{-1}H = \mathbf{1}^\top D^{-1}H$, hence $D^{-1}H$ is column-stochastic as well. Thus, without loss of generality one can consider NMFs of column-stochastic matrices into column-stochastic matrices [8, Theorem 3.2].

NMF problem: Given a matrix $M \in \mathbb{Q}_+^{n\times m}$ and $k \in \mathbb{N}$, is $\text{rank}_+(M) \leq k$?

The NMF problem is NP-hard, even for $k = \text{rank}(M)$ (see [20]). On the other hand, it is reducible to the existential theory of the reals, hence by [6, 16] it is in PSPACE.

For a matrix $M \in \mathbb{Q}_+^{n\times m}$, its nonnegative rank over $\mathbb{Q}$ is clearly at least $\text{rank}_+(M)$. While those ranks are equal if $\text{rank}(M) \leq 2$, a longstanding open question by Cohen and Rothblum asks whether they are always equal [8]. In other words, it is conceivable that there exists a rational matrix $M \in \mathbb{Q}_+^{n\times m}$ with $\text{rank}_+(M) = d$ that has no *rational* NMF with inner dimension d. Recently, Shitov [17] exhibited a nonnegative matrix (with irrational entries) whose nonnegative rank over a subfield of $\mathbb{R}$ is different from its nonnegative rank over $\mathbb{R}$.

2.2 Restricted Nonnegative Rank

For all matrices $M \in \mathbb{F}_+^{n\times m}$, an NMF $M = W \cdot H$ is called *restricted NMF (RNMF)* [11] if $\text{rank}(M) = \text{rank}(W)$. As we know $\text{Col}(M) \subseteq \text{Col}(W)$ holds for all NMF instances, the condition $\text{rank}(M) = \text{rank}(W)$ is then equivalent to $\text{Col}(M) = \text{Col}(W)$. The *restricted nonnegative rank over* $\mathbb{F}$ of M is the smallest number $d \in \mathbb{N}_0$ such that there exists a d-dimensional restricted nonnegative factorization over $\mathbb{F}$ of M. Unless indicated otherwise, henceforth we will assume $\mathbb{F} = \mathbb{R}$ when speaking of the restricted nonnegative rank of M, and denote it by $\text{rrank}_+(M)$.

RNMF problem: Given a matrix $M \in \mathbb{Q}_+^{n\times m}$ and $k \in \mathbb{N}$, is $\text{rrank}_+(M) \leq k$?

We have the following basic properties.

▶ **Lemma 1** ([11]). *Let* $M \in \mathbb{R}_+^{n\times m}$. *Then* $\text{rank}(M) \leq \text{rank}_+(M) \leq \text{rrank}_+(M) \leq m$. *Moreover, if* $\text{rank}(M) = \text{rank}_+(M)$ *then* $\text{rank}(M) = \text{rrank}_+(M)$.

Thus, with the above-mentioned NP-hardness result, it follows that the RNMF problem is also NP-hard and in PSPACE.

For a matrix $M \in \mathbb{Q}_+^{n\times m}$, its restricted nonnegative rank over $\mathbb{Q}$ is clearly at least $\text{rrank}_+(M)$. As with nonnegative rank, in general it is not known whether the restricted nonnegative ranks of M over $\mathbb{R}$ and over $\mathbb{Q}$ are equal. By [8, Theorem 4.1] and Lemma 1, this is true when $\text{rank}(M) \leq 2$.

RNMF has the following geometric interpretation. For a dimension $\ell \in \mathbb{N}$, the *convex combination* of a set $\{v_1, \ldots, v_m\} \subset \mathbb{R}^\ell$ is a point $\lambda_1 v_1 + \cdots + \lambda_m v_m$ where $(\lambda_1, \ldots, \lambda_m)$ is a stochastic vector. The *convex hull* of $\{v_1, \ldots, v_m\}$, written as $\text{conv}\{v_1, \ldots, v_m\}$, is the set of all convex combinations of $\{v_1, \ldots, v_m\}$. We call $\text{conv}\{v_1, \ldots, v_m\}$ a *polytope spanned by* $v_1, \ldots, v_m$. A *polyhedron* is a set $\{\, x \in \mathbb{R}^\ell \mid Ax + b \geq 0 \,\}$ with $A \in \mathbb{R}^{n\times\ell}$ and $b \in \mathbb{R}^n$. A set is a polytope if and only if it is a bounded polyhedron. A polytope is *full-dimensional* (i.e., has volume) if the matrix $(A \;\; b) \in \mathbb{R}^{n\times(\ell+1)}$ has rank $\ell + 1$.

Nested polytope problem (NPP): Given $r, n \in \mathbb{N}$, let $A \in \mathbb{Q}^{n\times(r-1)}$ and $b \in \mathbb{Q}^n$ be such that $P = \{\, x \in \mathbb{R}^{r-1} \mid Ax + b \geq 0 \,\}$ is a full-dimensional polytope. Let $S \subseteq P$ be a full-dimensional polytope described by spanning points. The *nested polytope problem (NPP)* asks, given A, b, S and a number $k \in \mathbb{N}$, whether there exist k points that span a polytope Q with $S \subseteq Q \subseteq P$. Such a polytope Q is called *nested* between P and S.

The following proposition appears as Theorem 1 in [11].

▶ **Proposition 2.** *The RNMF problem and the NPP are interreducible in polynomial time.*

More specifically, the reductions are as follows.

1. Given a nonnegative matrix $M \in \mathbb{Q}_+^{n\times m}$ of rank r, one can compute in polynomial time $A \in \mathbb{Q}^{n\times(r-1)}$ and $b \in \mathbb{Q}^n$ such that $P = \{\, x \in \mathbb{R}^{r-1} \mid Ax + b \geq 0 \,\}$ is a full-dimensional polytope, and m rational points that span a full-dimensional polytope $S \subseteq P$ such that
   - **(a)** any d-dimensional RNMF (rational or irrational) of M determines d points that span a polytope Q with $S \subseteq Q \subseteq P$, and
   - **(b)** any d points (rational or irrational) that span a polytope Q with $S \subseteq Q \subseteq P$ determine a d-dimensional RNMF of M.
2. Let $A \in \mathbb{Q}^{n\times(r-1)}$ and $b \in \mathbb{Q}^n$ such that $P = \{\, x \in \mathbb{R}^{r-1} \mid Ax+b \geq 0 \,\}$ is a full-dimensional polytope. Let $S \subseteq P$ be a full-dimensional polytope spanned by $s_1, \ldots, s_m \in \mathbb{Q}^{r-1}$. Then matrix $M \in \mathbb{Q}^{n\times m}$ with $M^i = As_i + b$ for $i \in [m]$ satisfies (a) and (b).

Importantly, the correspondences (a) and (b) preserve rationality. In the full version we detail the reduction from point 2 above, thereby filling in a small gap in the proof of [11].

▶ **Example 3** ([11, Example 1])**.** Using the geometric interpretation of restricted nonnegative rank it follows easily that, in general, we may have $\operatorname{rank}(M) < \operatorname{rank}_+(M) < \operatorname{rrank}_+(M)$. Let *3D-cube* NPP be the NPP instance where the inner and outer polytope are the standard 3D cube, i.e., $P = S = \{\, x \in \mathbb{R}^3 \mid x_i \in [0,1],\ 1 \leq i \leq 3 \,\}$. The only nested polytope is $Q = P$. The corresponding restricted NMF problem consists of the following matrix $M \in \mathbb{R}_+^{6\times 8}$:

$$M = \begin{pmatrix} 0&0&0&0&1&1&1&1 \\ 1&1&1&1&0&0&0&0 \\ 0&0&1&1&0&0&1&1 \\ 1&1&0&0&1&1&0&0 \\ 0&1&0&1&0&1&0&1 \\ 1&0&1&0&1&0&1&0 \end{pmatrix}.$$

We have $\operatorname{rrank}_+(M) = 8$ and $\operatorname{rank}(M) = 4$. Since $\operatorname{rank}_+(M)$ is bounded above by the number of rows in M, we have $\operatorname{rank}_+(M) \leq 6$. It is shown in [11] that $\operatorname{rank}_+(M) = 6$.

3 Coverability of Labelled Markov Chains

In this section, we establish a connection between RNMF and the coverability relation for labelled Markov chains. We thereby answer an open question posed in 1971 by Paz [15] about the nature of minimal covering labelled Markov chains.

A *labelled Markov chain* (*LMC*) is a tuple $\mathcal{M} = (n, \Sigma, \mu)$ where $n \in \mathbb{N}$ is the number of states, Σ is a finite alphabet of labels, and function $\mu : \Sigma \to [0,1]^{n\times n}$ specifies the transition matrices and is such that $\sum_{\sigma\in\Sigma} \mu(\sigma)$ is a row-stochastic matrix. The intuitive behaviour of the LMC $\mathcal{M}$ is as follows: When $\mathcal{M}$ is in state $i \in [n]$, it emits label σ and moves to state j, with probability $\mu(\sigma)_{i,j}$.

We extend the function μ to words by defining $\mu(\varepsilon) := I_n$ and $\mu(\sigma_1 \ldots \sigma_k) := \mu(\sigma_1) \cdots \mu(\sigma_k)$ for all $k \in \mathbb{N}$, and all $\sigma_1, \ldots, \sigma_k \in \Sigma$. Observe that $\mu(xy) = \mu(x) \cdot \mu(y)$ for all words $x, y \in \Sigma^*$. We view $\mu(w)$ for a word $w \in \Sigma^*$ as follows: if $\mathcal{M}$ is in state $i \in [n]$, it emits w and moves to state j in $|w|$ steps, with probability $\mu(w)_{i,j}$.

For $i \in [n]$ and $w \in \Sigma^*$, we write $pr_i^{\mathcal{M}}(w) := e_i^\top \cdot \mu(w) \cdot \mathbf{1}^{(n)}$ for the probability that, starting in state i, $\mathcal{M}$ emits word w. For example, in Figure 1 we have $pr_0^{\mathcal{M}}(a_1 b_1) = \frac{1}{12}$. More generally, for a given *initial distribution* π on the set of states $[n]$ (viewed as a stochastic row vector), we write $pr_\pi^{\mathcal{M}}(w) := \pi \cdot \mu(w) \cdot \mathbf{1}^{(n)}$ for the probability that $\mathcal{M}$ emits word w starting from state distribution π.

We say that an LMC $\mathcal{M}$ is *covered by* an LMC $\mathcal{M}'$, written as $\mathcal{M}' \geq \mathcal{M}$, if for every initial distribution π for $\mathcal{M}$ there exists a distribution π' for $\mathcal{M}'$ such that $pr_\pi^{\mathcal{M}}(w) = pr_{\pi'}^{\mathcal{M}'}(w)$ for all words $w \in \Sigma^*$.

The *backward matrix* of $\mathcal{M}$ is a matrix $\operatorname{Back}\mathcal{M} \in \mathbb{R}_+^{[n]\times\Sigma^*}$ where $(\operatorname{Back}\mathcal{M})_{i,w} = pr_i^{\mathcal{M}}(w)$ for every $i \in [n]$ and $w \in \Sigma^*$. The *rank* of $\mathcal{M}$ is defined by $\operatorname{rank}(\mathcal{M}) = \operatorname{rank}(\operatorname{Back}\mathcal{M})$. (Matrix $\operatorname{Back}\mathcal{M}$ is infinite, but since it has n rows, its rank is at most n.) It follows easily from the definition (see also [15, Theorem 3.1]) that $\mathcal{M}' \geq \mathcal{M}$ if and only if there exists a row-stochastic matrix A such that $A \cdot \operatorname{Back}\mathcal{M}' = \operatorname{Back}\mathcal{M}$.

LMCs can be seen as a special case of stochastic sequential machines, a class of probabilistic automata introduced and studied by Paz [15]. More specifically, they are stochastic sequential machines with a singleton input alphabet and Σ as output alphabet. In his seminal 1971 textbook on probabilistic automata [15], Paz asks the following question:

▶ **Question 4** (Paz [15], p. 38)**.** *If an n-state LMC $\mathcal{M}$ is covered by an n'-state LMC $\mathcal{M}'$ where $n' < n$, is $\mathcal{M}$ necessarily covered by some n^*-state LMC $\mathcal{M}^*$, where $n^* < n$, such that $\mathcal{M}^*$ and $\mathcal{M}$ have the same rank?*

In 1974, a positive answer to this question was claimed [3, Theorem 13]. In fact, the author of [3] makes a stronger claim, namely that the answer to Question 4 is yes, even if the inequality $n^* < n$ in Question 4 is replaced by $n^* \leq n'$. To the contrary, we show:

▶ **Theorem 5.** *The answer to Question 4 is negative.*

Theorem 5 falsifies the claim in [3]. In the full version we discuss in detail the mistake in [3]. To prove Theorem 5 we establish a tight connection between NMF and LMC coverability:

▶ **Proposition 6.** *Given a nonnegative matrix $M \in \mathbb{Q}_+^{n\times m}$ of rank r, one can compute in polynomial time an LMC $\mathcal{M} = (m+2, \Sigma, \mu)$ of rank $r+2$ such that for all $d \in \mathbb{N}$:*

(a) *any d-dimensional NMF $M = W \cdot H$ determines an LMC $\mathcal{M}' = (d+2, \Sigma, \mu')$ with $\mathcal{M}' \geq \mathcal{M}$ and $\operatorname{rank}(\mathcal{M}') = \operatorname{rank}(W) + 2$, and*

(b) *any LMC $\mathcal{M}' = (d+2, \Sigma, \mu')$ with $\mathcal{M}' \geq \mathcal{M}$ determines a d-dimensional NMF $M = W \cdot H$ with $\operatorname{rank}(\mathcal{M}') = \operatorname{rank}(W) + 2$.*

In particular, for all $d \in \mathbb{N}$ the inequality $\operatorname{rrank}_+(M) \leq d$ holds if and only if $\mathcal{M}$ is covered by some $(d+2)$-state LMC $\mathcal{M}'$ such that $\mathcal{M}'$ and $\mathcal{M}$ have the same rank.

Assuming Proposition 6 we can prove Theorem 5:

Proof of Theorem 5. Let $M \in \{0,1\}^{6\times 8}$ be the matrix from Example 3. Let $\mathcal{M} = (10, \Sigma, \mu)$ be the associated LMC from Proposition 6. Since $M = I_6 \cdot M$ is an NMF with inner dimension 6, by Proposition 6 (a) there is an LMC $\mathcal{M}' = (8, \Sigma, \mu')$ with $\mathcal{M}' \geq \mathcal{M}$. Towards a contradiction, suppose the answer to Question 4 were yes. Then $\mathcal{M}$ is also covered by some n^*-state LMC $\mathcal{M}^*$, where $n^* \leq 9$, such that $\mathcal{M}^*$ and $\mathcal{M}$ have the same rank. The last sentence of Proposition 6 then implies that $\operatorname{rrank}_+(M) \leq 7$. But this contradicts the equality $\operatorname{rrank}_+(M) = 8$ from Example 3. Hence, the answer to Question 4 is no. ◀

To prove Proposition 6 we adapt a reduction from NMF to the trace-refinement problem in Markov decision processes [10].

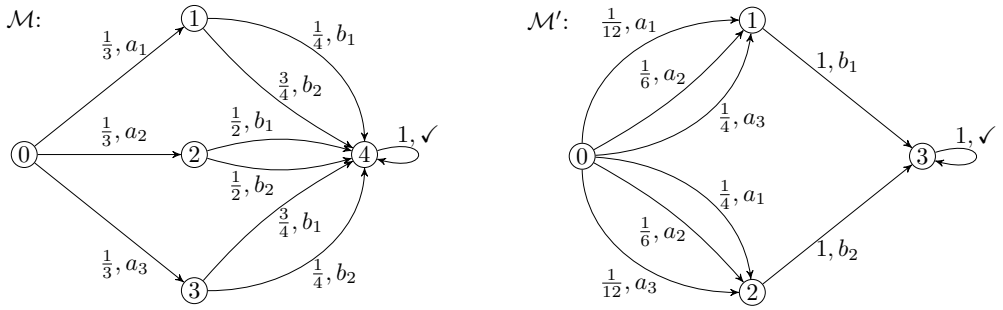


Figure 1 LMC $\mathcal{M}$ is constructed from matrix $M = \left(\begin{smallmatrix} 1/4 & 1/2 & 3/4 \\ 3/4 & 1/2 & 1/4 \end{smallmatrix}\right)$ whereas LMC $\mathcal{M}'$ is obtained by NMF $M = I_2 \cdot M$.

Proof sketch of Proposition 6. Let $M \in \mathbb{Q}_+^{n\times m}$ be a nonnegative matrix of rank r. As argued in Section 2.1, without loss of generality we may assume that M is column-stochastic and consider factorizations of M into column-stochastic matrices only.

We define an LMC $\mathcal{M} = (m+2, \Sigma, \mu)$ with $m+2$ states $\{0, 1, \ldots, m, m+1\}$. The alphabet is $\Sigma = \{a_1, \ldots, a_m\} \cup \{b_1, \ldots, b_n\} \cup \{\checkmark\}$ and the function μ, for all $i \in [m]$ and all $j \in [n]$, is defined by:

$$\mu(a_i)_{0,i} = \tfrac{1}{m}, \qquad \mu(b_j)_{i,m+1} = (M^\top)_{i,j} = M_{j,i}, \qquad \mu(\checkmark)_{m+1,m+1} = 1,$$

and all other entries of $\mu(a_i)$, $\mu(b_j)$, and $\mu(\checkmark)$ are 0. See Figure 1 for an example. We have:

$$\operatorname{Back}\mathcal{M} = \begin{array}{c} \begin{array}{ccccccccccc} \varepsilon & b_1 & \cdots & b_n & \checkmark & a_i & a_ib_j & b_1\checkmark & \cdots & b_n\checkmark & \checkmark^2 & \cdots \end{array} \\ \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \frac{1}{m} & \frac{1}{m}M_{j,i} & 0 & \cdots & 0 & 0 & \cdots \\ 1 & M_{1,1} & \cdots & M_{n,1} & 0 & 0 & 0 & M_{1,1} & \cdots & M_{n,1} & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \cdots \\ 1 & M_{1,m} & \cdots & M_{n,m} & 0 & 0 & 0 & M_{1,m} & \cdots & M_{n,m} & 0 & \cdots \\ 1 & 0 & \cdots & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 1 & \cdots \end{pmatrix} \end{array}.$$

Thus $\operatorname{rank}(\mathcal{M}) = \operatorname{rank}(\operatorname{Back}\mathcal{M}) \geq \operatorname{rank}(M) + 2$. The first $n+2$ columns (indexed by $\varepsilon, b_1, \ldots, b_n, \checkmark$) in $\operatorname{Back}\mathcal{M}$ span $\operatorname{Col}(\operatorname{Back}\mathcal{M})$. Therefore, $\operatorname{rank}(\mathcal{M}) = \operatorname{rank}(M) + 2 = r + 2$.

For $d \in \mathbb{N}$, let $M = W \cdot H$ for some column-stochastic matrices $W \in \mathbb{R}_+^{n\times d}$ and $H \in \mathbb{R}_+^{d\times m}$. Define an LMC $\mathcal{M}' = (d+2, \Sigma, \mu')$ where the states are $\{0, 1, \ldots, d, d+1\}$. The function μ', for all $i \in [m]$, $j \in [n]$, and $l \in [d]$, is defined by:

$$\mu'(a_i)_{0,l} = \tfrac{1}{m}H_{l,i}, \qquad \mu'(b_j)_{l,d+1} = W_{j,l}, \qquad \mu'(\checkmark)_{d+1,d+1} = 1,$$

and all other entries of $\mu'(a_i)$, $\mu'(b_j)$, and $\mu'(\checkmark)$ are 0. From the NMF $M = W \cdot H$ it follows that we can factor $\operatorname{Back}\mathcal{M}$ as follows:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & H_{1,1} & \cdots & H_{d,1} & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & H_{1,m} & \cdots & H_{d,m} & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \cdot \begin{array}{c} \begin{array}{ccccccccccc} \varepsilon & b_1 & \cdots & b_n & \checkmark & a_i & a_ib_j & b_j\checkmark & \checkmark^2 & \cdots \end{array} \\ \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \frac{1}{m} & \frac{1}{m}M_{j,i} & 0 & 0 & \cdots \\ 1 & W_{1,1} & \cdots & W_{n,1} & 0 & 0 & 0 & W_{j,1} & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \cdots \\ 1 & W_{1,d} & \cdots & W_{n,d} & 0 & 0 & 0 & W_{j,d} & 0 & \cdots \\ 1 & 0 & \cdots & 0 & 1 & 0 & 0 & 0 & 1 & \cdots \end{pmatrix} \end{array}$$

where the left factor is row-stochastic (as H is column-stochastic), and the right factor equals $\operatorname{Back}\mathcal{M}'$. It follows that $\mathcal{M}' \geq \mathcal{M}$. ◀

4 Restricted NMF of Rank-3 Matrices

In this section we consider rational matrices of rank at most 3. We show that for such matrices the restricted nonnegative ranks over $\mathbb{R}$ and $\mathbb{Q}$ are equal and we give a polynomial-time algorithm that computes a minimal-dimension RNMF over $\mathbb{Q}$.

▶ **Theorem 7.** *Given a matrix $M \in \mathbb{Q}_+^{n\times m}$ where $\operatorname{rank}(M) \leq 3$, there is a rational RNMF of M with inner dimension $\operatorname{rrank}_+(M)$ and it can be computed in polynomial time in the Turing model of computation.*

Using reduction 1 of Proposition 2, we can reduce in polynomial time the RNMF problem for rank-3 matrices to the 2-dimensional NPP, i.e., the nested polygon problem in the plane. As noted in Section 2.2, the correspondence between restricted nonnegative factorizations and nested polygons preserves rationality. Thus to prove Theorem 7 it suffices to prove:

▶ **Theorem 8.** *Given polygons $S \subseteq P \subseteq \mathbb{R}^2$ with rational vertices, there exists a minimum-vertex polygon Q nested between P and S that also has rational vertices. Moreover there is an algorithm that, given P and S, computes such a polygon in polynomial time in the Turing machine model.*

In fact, Aggarwal et al. [1] give an algorithm for the 2-dimensional NPP and prove that it runs in polynomial time in the RAM model with unit-cost arithmetic. However, they freely use trigonometric functions and do not address the rationality of the output of the algorithm nor its complexity in the Turing model. To prove Theorem 8 we show that, by adopting a suitable representation of the vertices of a nested polygon, the algorithm in [1] can be adapted so that it runs in polynomial time in the Turing model. We furthermore use this representation to prove that the minimum-vertex nested polygon identified by the resulting algorithm has rational vertices.

The remainder of the section is devoted to the proof of Theorem 8. We first recall some terminology from [1] and describe their algorithm.

A *supporting line segment* is a directed line segment, with its initial and final points on the boundary of the outer polygon P, that touches the inner polygon S on its left. A nested polygon with vertices on the boundary of P is said to be *supporting* if all but at most one of its edges are supporting line segments. A polygon nested between P and S is called *minimal* if it has the minimum number of vertices among all polygons nested between P and S. It is shown in [1, Lemma 4] that there is always a supporting polygon that is also minimal, and the algorithm given therein outputs such a polygon.

Let k be the number of vertices of a minimal nested polygon. Given a vertex v on the boundary of P, there is a uniquely defined supporting polygon Q_v with at most $k+1$ vertices. To determine Q_v one computes the supporting line segments $v_1v_2, \ldots, v_kv_{k+1}$, where $v_1 = v$; see Figure 2. Then Q_v is either the polygon with vertices $v_1, \ldots, v_k$ or the polygon with vertices $v_1, \ldots, v_{k+1}$. In the first case, Q_v is minimal. The idea behind the algorithm of [1] is to search along the boundary of P for an initial vertex v such that Q_v is minimal.

As a central ingredient for our proof of Theorem 8, we choose a convenient representation of the vertices of supporting polygons. To this end, we assume that the edges of P are oriented counter-clockwise, and we represent a vertex v on an edge pq of P by the unique $\lambda \in [0, 1]$ such that $v = (1 - \lambda)p + \lambda q$. We call this the *convex representation* of v.

Similar to [1], we associate with each supporting line segment uv a *ray function* r, such that if λ is the convex representation of u then $r(\lambda)$ is the convex representation of v. The same ray function applies for supporting line segments $u'v'$ with u' in a small enough interval containing u.

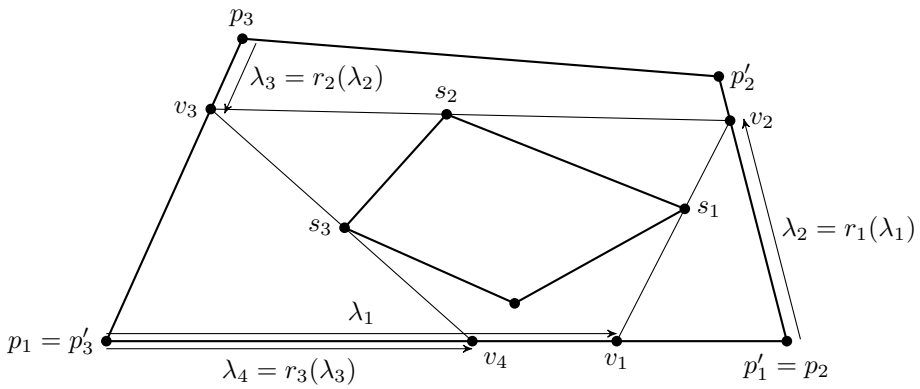


Figure 2 Supporting polygon Q_{v_1}. For every $i \in [3]$, vertex v_i lies on edge $p_i p_i'$ of P, and s_i is the point where the supporting line segment $v_i v_{i+1}$ touches the inner polygon S on its left.

In the following, when we say polynomial time, we mean polynomial time in the Turing model. To obtain a polynomial time bound, the key lemma is as follows:

▶ **Lemma 9.** *Consider bounded polygons $S \subseteq P \subseteq \mathbb{R}^2$ whose vertices are rational and of bit-length L. Then the ray function associated with a supporting line segment uv has the form $r(\lambda) = \frac{a\lambda+b}{c\lambda+d}$, where coefficients a, b, c, d are rational numbers with bit-length $O(L)$ that can be computed in polynomial time.*

Suppose that $v_1v_2, \ldots, v_kv_{k+1}$ is a sequence of k supporting line segments, with corresponding ray functions $r_1, \ldots, r_k$. Then $v_1, \ldots, v_k$ are the vertices of a minimal supporting polygon if and only if $(r_k \circ \ldots \circ r_1)(\lambda) \geq \lambda$, where λ is the convex representation of v_1.

It follows from [1] that, for each edge of P, one can compute in polynomial time a partition $\mathcal{I}$ of $[0, 1]$ into intervals with rational endpoints such that if λ_1, λ_2 are in the same interval $I \in \mathcal{I}$ then the points with convex representation λ_1 and λ_2 are associated with the same sequence of ray functions $r_1, \ldots, r_k$. Using Lemma 9 we can, for each interval $I \in \mathcal{I}$, compute these ray functions in polynomial time. Define the *slack function* $s(\lambda) = (r_k \circ \ldots \circ r_1)(\lambda) - \lambda$. In fact, this function has the form $s(\lambda) = \frac{a\lambda+b}{c\lambda+d} - \lambda$ for rational numbers a, b, c, d that are also computable in polynomial time. Then it is straightforward to check whether $s(\lambda) \geq 0$ has a solution $\lambda \in I$.

Next we show that if such a solution exists, then there exists a rational solution, which, moreover, can be computed in polynomial time. To this end, let $\lambda^* \in I$ be such that $s(\lambda^*) \geq 0$. If λ^* is on the boundary of I, then $\lambda^* \in \mathbb{Q}$. If λ^* is not on the boundary and is not an isolated solution, then there exists a rational solution in its neighbourhood. Lastly, let λ^* be an isolated solution not on the boundary. Then, λ^* is a root of both s and its derivative s'. For every $\lambda \in I$, we have

$$(c\lambda + d) \cdot s(\lambda) = a\lambda + b - \lambda \cdot (c\lambda + d).$$

Taking the derivative of the above equation with respect to λ, we get

$$c \cdot s(\lambda) + (c\lambda + d) \cdot s'(\lambda) = a - d - 2c\lambda. \tag{1}$$

Since $s(\lambda^*) = s'(\lambda^*) = 0$, from (1) we get $0 = a - d - 2c\lambda^*$. Note that $c \neq 0$ since otherwise $s \equiv 0$. Therefore, $\lambda^* = \frac{a-d}{2c} \in \mathbb{Q}$.

It follows that the vertex v represented by λ^* has rational coordinates computable in polynomial time. By computing $(r_i \circ \ldots \circ r_1)(\lambda^*)$ for $i \in [k]$, we can compute in polynomial time the convex representation of all vertices of the supporting polygon Q_v. Observe, in particular, that all vertices are rational. Hence we have proved Theorem 8.

5 Restricted NMF Requires Irrationality

Here we show that the restricted nonnegative ranks over $\mathbb{R}$ and $\mathbb{Q}$ are, in general, different.

▶ **Theorem 10.** *Let*

$$M = \begin{pmatrix} 1/8 & 1/2 & 17/22 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 3/4 & 7/12 \\ 3/4 & 3/4 & 3/11 & 2 & 1/2 & 1/6 \\ 1/4 & 1/4 & 8/11 & 1/4 & 19/8 & 55/24 \\ 1/2 & 1/8 & 1/11 & 1/8 & 15/16 & 17/16 \\ 11/16 & 5/16 & 7/44 & 1/16 & 7/32 & 43/96 \end{pmatrix} \in \mathbb{Q}_+^{6\times 6}.$$

The restricted nonnegative rank of M over $\mathbb{R}$ is 5. The restricted nonnegative rank of M over $\mathbb{Q}$ is 6.

Proof. Matrix M has an NMF $M = W \cdot H$ with inner dimension 5 with W, H as follows:

$$W = \begin{pmatrix} 0 & \frac{3+\sqrt{2}}{14} & \frac{11+\sqrt{2}}{14} & 0 & 0 \\ 0 & 0 & 0 & \frac{12-2\sqrt{2}}{17} & \frac{5}{7}+\frac{\sqrt{2}}{14} \\ 2-\sqrt{2} & 1 & \frac{3-\sqrt{2}}{7} & \frac{26+7\sqrt{2}}{17} & 0 \\ -1+\sqrt{2} & 0 & \frac{4+\sqrt{2}}{7} & \frac{21-12\sqrt{2}}{17} & \frac{39}{14}+\frac{5\sqrt{2}}{28} \\ \frac{\sqrt{2}}{2} & \frac{2}{7}-\frac{\sqrt{2}}{14} & 0 & \frac{7-4\sqrt{2}}{17} & \frac{33}{28}+\frac{\sqrt{2}}{56} \\ \frac{1}{2}+\frac{\sqrt{2}}{4} & \frac{15}{28}-\frac{\sqrt{2}}{14} & \frac{3-\sqrt{2}}{28} & 0 & \frac{3}{8}-\frac{\sqrt{2}}{16} \end{pmatrix},$$

$$H = \begin{pmatrix} \frac{1+\sqrt{2}}{4} & 0 & \frac{\sqrt{2}}{11} & \frac{1}{4}-\frac{\sqrt{2}}{8} & 0 & \frac{1}{6}+\frac{\sqrt{2}}{12} \\ \frac{3-\sqrt{2}}{4} & \frac{1}{2}+\frac{\sqrt{2}}{8} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2}-\frac{\sqrt{2}}{8} & 1-\frac{\sqrt{2}}{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{3}{4}+\frac{\sqrt{2}}{8} & \frac{13}{34}-\frac{7\sqrt{2}}{68} & 0 \\ 0 & 0 & 0 & 0 & \frac{21}{34}+\frac{7\sqrt{2}}{68} & \frac{5}{6}-\frac{\sqrt{2}}{12} \end{pmatrix}.$$

Since $\operatorname{rank}(M) = \operatorname{rank}(W) = 4$, the NMF $M = W \cdot H$ is restricted. This RNMF has been obtained by reducing, according to Proposition 2, an NPP instance, which we now describe.

Figure 3 shows the outer 3-dimensional polytope P with 6 faces. The polytope P is the intersection of the following half-spaces: $y \geq 0$ (blue), $z \geq 0$ (brown), $x \geq 0$ (pink), $-x + \frac{5}{2}z + 1 \geq 0$ (yellow), $-\frac{1}{2}x - y + \frac{1}{4}z + 1 \geq 0$ (green), $-\frac{1}{4}x - y - \frac{7}{8}z + 1 \geq 0$ (transparent front). The figure also indicates an interior polytope S spanned by 6 points (black dots): $s_1 = (\frac{3}{4}, \frac{1}{8}, 0)^\top$, $s_2 = (\frac{3}{4}, \frac{1}{2}, 0)^\top$, $s_3 = (\frac{3}{11}, \frac{17}{22}, 0)^\top$, $s_4 = (2, 0, \frac{1}{2})^\top$, $s_5 = (\frac{1}{2}, 0, \frac{3}{4})^\top$, $s_6 = (\frac{1}{6}, 0, \frac{7}{12})^\top$. In the following we discuss possible locations of 5 points q_1, q_2, q_3, q_4, q_5 that span a nested polytope Q. Since s_1, s_2, s_3 all lie on the (brown) face on the xy-plane, but not on a common line, at least 3 of the q_i must lie on the xy-plane. A similar statement holds for s_4, s_5, s_6 and the xz-plane. So at least one q_i, say q_1, must lie on the x-axis.

Suppose another q_i, say q_2, lies on the x-axis. Without loss of generality we can take $q_1 = (0, 0, 0)^\top$ and $q_2 = (1, 0, 0)^\top$, as all points in P on the x-axis are enclosed by these q_1, q_2.

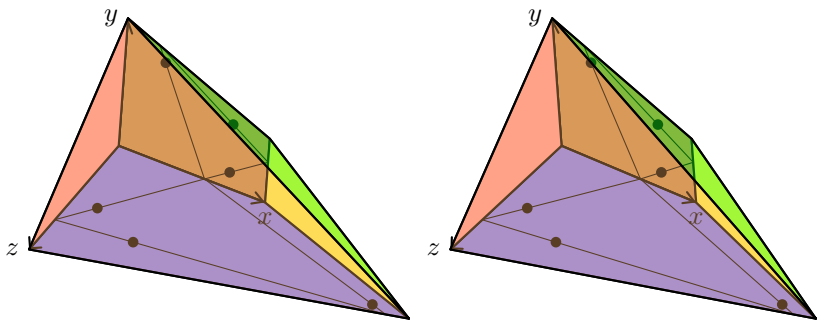


Figure 3 Instance of the nested polytope problem. The two images show orthogonal projections of a 3-dimensional outer polytope P. The black dots indicate 6 inner points (3 on the brown xy-face, and 3 on the blue xz-face) that span the interior polytope S. The two triangles on the xy-face and on the xz-face indicate the (unique) location of 5 points that span the nested polytope Q. The two slightly different projections are designed to create a 3-dimensional impression using stereoscopy. The "parallel-eye" technique should be used, see, e.g., [18]. See the full version for a "cross-eyed" variant.

Figure 4 provides a detailed view of the xy-plane. To enclose s_2, some $q \in \{q_3, q_4, q_5\}$ must also lie on the xy-plane and to the right of the line that connects $q_2 = (1,0,0)^\top$ and s_2. To enclose s_3, some $q' \in \{q_3, q_4, q_5\}$ must also lie on the xy-plane and to the left of the line that connects $q_1 = (0,0,0)^\top$ and s_3. If q and q' were identical then they would lie outside P – a contradiction. Hence 4 points (namely, q_1, q_2, q, q') are on the xy-plane. This leaves only one point, say q'', that is not on the xy-plane. To enclose s_4 (see the corresponding figure in the full version), point q'' must lie on the xz-plane and must lie to the right of the line that connects $q_2 = (1,0,0)^\top$ and s_4. To enclose s_6, point q'' must lie to the left of the line that connects $q_1 = (0,0,0)^\top$ and s_6. Hence q'' lies outside P – a contradiction.

Hence we have shown that only one point, say q_1, lies on the x-axis, and two points besides q_1, say q_2, q_3, lie on the xy-plane, and two points besides q_1, say q_4, q_5, lie on the xz-plane. Figure 4 indicates a possible location (q_1^*, q_2^*, q_3^*) of q_1, q_2, q_3. The figure illustrates that the x-coordinate of q_1^* must be at least $2 - \sqrt{2}$.

Figure 5 illustrates how to prove the same fact more formally, using the concept of a slack function (see Section 4): The slack function $s(\lambda)$ for the interval containing $2 - \sqrt{2}$ has a zero at $\lambda = 2 - \sqrt{2}$, with a sign change from negative to positive. An inspection of the intervals (of the partition $\mathcal{I}$ from Section 4) to the "left" of $2 - \sqrt{2}$ reveals that none of the corresponding slack functions $\tilde{s}$ satisfies $\tilde{s}(\lambda) \geq 0$ for $\lambda < 2 - \sqrt{2}$. Similarly, the x-coordinate of q_1^* must be *at most* $2 - \sqrt{2}$, see corresponding figures in the full version. Hence $q_1^* = (2 - \sqrt{2}, 0, 0)^\top$ is necessary. This uniquely (up to permutations) determines q_2^*, q_3^* and similarly the locations q_4^*, q_5^* of q_4, q_5. With the reduction from Proposition 2 this NPP solution determines the RNMF of M mentioned at the beginning of the proof. Since there is no 4-point solution of the NPP instance, we have $\operatorname{rrank}_+(M) = 5$. (Since $\operatorname{rank}(M) = 4$, Lemma 1 implies $\operatorname{rank}_+(M) = 5$.) Since there is no 5-point rational solution of the NPP instance, the restricted nonnegative rank of M over $\mathbb{Q}$ is 6. ◀

6 Conclusion and Future Work

We have shown that an optimal *restricted* nonnegative factorization of a rational matrix may require factors that have irrational entries. An outstanding open problem is whether the same holds for general nonnegative factorizations. An answer to this question will likely shed light on the issue of whether the nonnegative rank can be computed in NP.

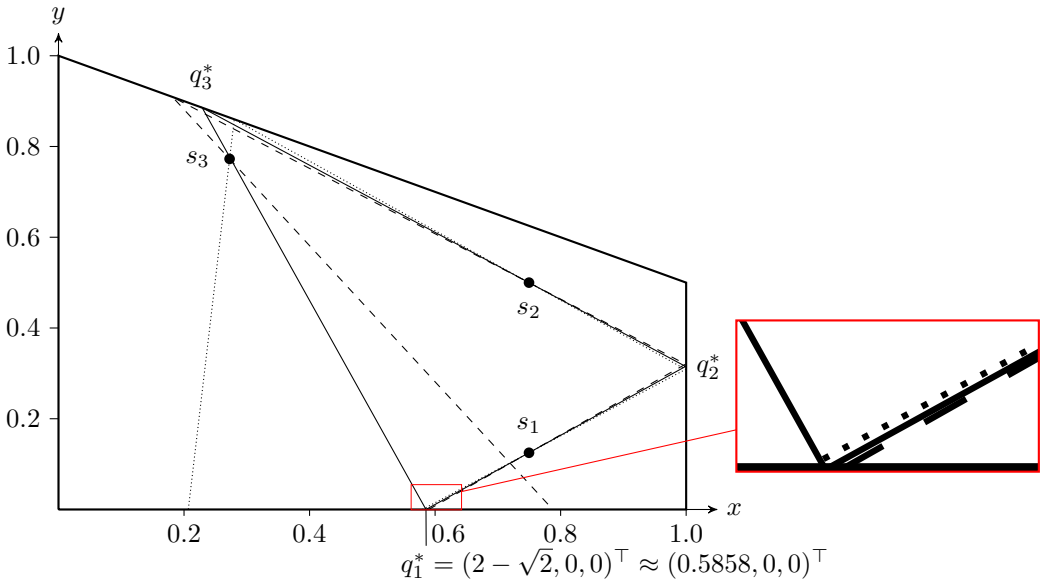


Figure 4 Detailed view of the xy-plane. The outer quadrilateral is one of 6 faces of P, the brown face in Figure 3. The points s_1, s_2, s_3 are among the 6 points that span the inner polytope S. The points q_1^*, q_2^*, q_3^* are among the 5 points that span the nested polytope Q. The area around q_1^* is zoomed in on the right-hand side. The picture illustrates that q_1^* cannot be moved left on the x-axis without increasing the number of vertices of the nested polytope: A dotted ray from a point slightly to the left of q_1^* is drawn through s_1. Its intersection with the line $x = 1$ is slightly below q_2^*. Following the algorithm of [1], the dotted ray is continued in a similar fashion, "wrapping around" s_2 and s_3, and ending on the x-axis at around $x \approx 0.2$, far left of the starting point. On the other hand, the dashed line illustrates that q_1^* could be moved right (considering only this face).

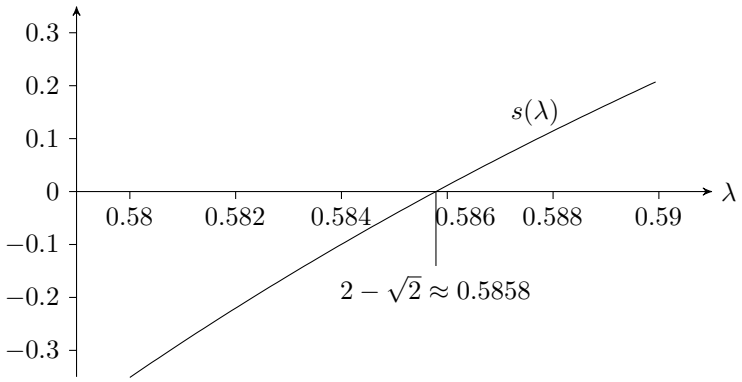


Figure 5 The slack function $s(\lambda) = \frac{52\lambda-30}{15\lambda-8} - \lambda$ corresponding to Figure 4. When $s(\lambda) < 0$, there is no nested triangle with vertex $(\lambda, 0, 0)$.

Another contribution of the paper has been to develop connections between nonnegative matrix factorization and probabilistic automata, thereby answering an old question concerning the latter. Pursuing this connection, and closely related to the above-mentioned open problem, one can ask whether, given a probabilistic automaton with rational transition probabilities, one can always find a minimal equivalent probabilistic automaton that also has rational transition probabilities.

Acknowledgements. The authors would like to thank Michael Benedikt for stimulating discussions, and anonymous referees for their helpful suggestions.

References

1 A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, and C. K. Yap. Finding minimal convex nested polygons. *Information and Computation*, 83(1):98–110, 1989.

2 S. Arora, R. Ge, R. Kannan, and A. Moitra. Computing a nonnegative matrix factorization – provably. In *Proceedings of the 44th Symposium on Theory of Computing (STOC)*, pages 145–162, 2012.

3 F. Bancilhon. A geometric model for stochastic automata. *IEEE Trans. Computers*, 23(12):1290–1299, 1974.

4 M. W. Berry, N. Gillis, and F. Glineur. Document classification using nonnegative matrix factorization and underapproximation. In *International Symposium on Circuits and Systems (ISCAS)*, pages 2782–2785. IEEE, 2009.

5 S. S. Bucak and B. Günsel. Video content representation by incremental non-negative matrix factorization. In *Proceedings of the International Conference on Image Processing (ICIP)*, pages 113–116. IEEE, 2007.

6 J. Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the 20th Annual Symposium on Theory of Computing (STOC)*, pages 460–467, 1988.

7 A. Cichocki, R. Zdunek, A. H. Phan, and S.-i. Amari. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley Publishing, 2009.

8 J. E. Cohen and U. G. Rothblum. Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 190:149–168, 1993.

9 D. L. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *Advances in Neural Information Processing Systems (NIPS)*, pages 1141–1148, 2003.

10 N. Fijalkow, S. Kiefer, and M. Shirmohammadi. Trace refinement in labelled Markov decision processes. In *Proceedings of the 19th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 9634 of *Lecture Notes in Computer Science*, pages 303–318. Springer, 2016.

11 N. Gillis and F. Glineur. On the geometric interpretation of the nonnegative rank. *Linear Algebra and its Applications*, 437(11):2685–2712, 2012.

12 N. Gillis and S. A. Vavasis. Semidefinite programming based preconditioning for more robust near-separable nonnegative matrix factorization. *SIAM Journal on Optimization*, 25(1):677–698, 2015.

13 A. Kumar, V. Sindhwani, and P. Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, page 231–239, 2013.

14 D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

15 A. Paz. *Introduction to probabilistic automata*. Academic Press, New York, 1971.

16 J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. Parts I–III. *Journal of Symbolic Computation*, 13(3):255–352, 1992.

17 Y. Shitov. Nonnegative rank depends on the field. Technical report, arxiv.org, 2015. Available at `http://arxiv.org/abs/1505.01893`.

18 D. Simanek. How to view 3D without glasses. `https://www.lhup.edu/~dsimanek/3d/view3d.htm`. Online, accessed in April 2016.

19 E. Tjioe, M. W. Berry, and R. Homayouni. Discovering gene functional relationships using FAUN (feature annotation using nonnegative matrix factorization). *BMC Bioinformatics*, 11(S-6):S14, 2010.

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

21 T. Yokota, R. Zdunek, A. Cichocki, and Y. Yamashita. Smooth nonnegative matrix and tensor factorizations for robust multi-way data analysis. *Signal Processing*, 113:234–249, 2015.

22 S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using nonnegative matrix factorization. In *Proceedings of the 6th SIAM International Conference on Data Mining*, pages 549–553. SIAM, 2006.

Proving the Herman-Protocol Conjecture*

Maria Bruna[1], Radu Grigore[2], Stefan Kiefer[†3], Joël Ouaknine[‡4], and James Worrell[5]

1 **Mathematical Institute, University of Oxford, Oxford, United Kingdom**
2 **School of Computing, University of Kent, Canterbury, United Kingdom**
3 **Department of Computer Science, University of Oxford, Oxford, United Kingdom**
4 **Department of Computer Science, University of Oxford, Oxford, United Kingdom**
5 **Department of Computer Science, University of Oxford, Oxford, United Kingdom**

Abstract

Herman's self-stabilization algorithm, introduced 25 years ago, is a well-studied synchronous randomized protocol for enabling a ring of N processes collectively holding any odd number of tokens to reach a stable state in which a single token remains. Determining the worst-case expected time to stabilization is the central outstanding open problem about this protocol. It is known that there is a constant h such that any initial configuration has expected stabilization time at most hN^2. Ten years ago, McIver and Morgan established a lower bound of $4/27 \approx 0.148$ for h, achieved with three equally-spaced tokens, and conjectured this to be the optimal value of h. A series of papers over the last decade gradually reduced the upper bound on h, with the present record (achieved in 2014) standing at approximately 0.156. In this paper, we prove McIver and Morgan's conjecture and establish that $h = 4/27$ is indeed optimal.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases randomized protocols, self-stabilization, Lyapunov function, expected time

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.104

1 Introduction

The notion of *self-stabilization* was introduced in a seminal paper of Dijkstra [11], and rose to prominence a decade later, following (among others) an invited talk of Lamport during which he pointed out that "self-stabilization [is] a very important concept in fault tolerance" [22]. Both self-stabilization and fault tolerance have since become central themes in distributed computing (see, e.g., [12]), as recently witnessed by the award of the 2015 Edsger W. Dijkstra Prize in Distributed Computing to Michael Ben-Or and Michael Rabin for "starting the field of fault-tolerant randomized distributed algorithms" in the early 1980s.

In this paper, we examine an early self-stabilization algorithm known as Herman's Protocol [19], whose exact mathematical analysis has proven remarkably challenging over the two-and-a-half decades since its inception. This algorithm considers a ring of N processes

* Full version at `http://arxiv.org/abs/1504.01130`.
† Stefan Kiefer is supported by a University Research Fellowship of the Royal Society and by EPSRC grant EP/M003795/1.
‡ Joël Ouaknine is supported by ERC grant AVS-ISS (648701).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 104; pp. 104:1–104:12


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

(or nodes), where each process either holds or doesn't hold a token. Starting from any initial configuration of K tokens, where K is required to be odd, Herman's algorithm proceeds as follows: at each time step, every process that holds a token either keeps it or passes it to its clockwise neighbor with probability 1/2. All updates happen synchronously, and if a process finds itself with two tokens (having simultaneously kept one and received one from its counterclockwise neighbor) then both tokens are annihilated. It is straightforward to see that, starting from an odd number of tokens and following this procedure, almost surely only one token eventually remains, at which point the ring is said to have *stabilized*.

Herman's original paper [19] presents the algorithm in a form amenable to implementation. Each process possesses a bit, which the process can read and write. Each process can also read the bit of its counterclockwise neighbor. In this representation, having the same bit as one's counterclockwise neighbor is interpreted as having a token. At each time step, each process compares its bit with the bit of its counterclockwise neighbor; if the bits differ, the process keeps its bit, whereas if the bits are the same, the process flips its bit with probability 1/2 and keeps it with probability 1/2. It is straightforward to verify that the bit-flipping version is an implementation of the token-passing version: in particular, a process flipping its bit corresponds to passing its token to its clockwise neighbor. If the number of processes is odd, by construction this bit representation forces the number of tokens to be odd as well, which justifies the assumption that K, the number of tokens, is always odd. In this paper we make no assumption about the parity of the number of *processes*, as we abstract from the bit implementation, and simply assume that the number of tokens is odd throughout.

Herman's original paper [19] showed that the expected time (number of synchronous steps) to stabilization is $O(N^2 \log N)$. The same paper also mentions an improved upper bound of $O(N^2)$ due to Dolev, Israeli, and Moran, without giving a proof or a further reference. In 2004, Fribourg et al. [16] established an upper bound of $2N^2$, and the following year Nakata [24] gave a tighter upper bound of $0.936N^2$ and exhibited an initial configuration with expected stablization time $\Omega(N^2)$. At the same time and independently, McIver and Morgan showed in [23] that the initial configuration consisting of three equally-spaced tokens has an expected stabilization time of exactly $\frac{4}{27}N^2$, and conjectured that this value is an upper bound on the expected time to stabilization starting from *any* initial configuration with *any* (odd) number of tokens. The conjecture is intriguing since increasing the initial number of tokens might be thought to lengthen the expected time to stabilization, due to the larger number of collisions required to achieve stabilization.

Nevertheless, McIver and Morgan's Herman-Protocol Conjecture is supported by considerable amount of experimental evidence [5], and in the intervening years a series of papers have gradually reduced the upper bound on the constant h such that stabilization from any initial configuration takes expected time at most hN^2: upper bounds of approximately 0.64, 0.521, 0.167, and 0.156 are given respectively in [21, 13, 14, 18], the last one provided last year by Haslegrave, and coming relatively close to McIver and Morgan's lower bound of $4/27 \approx 0.148$.

In this paper, we prove McIver and Morgan's conjecture and establish that $h = 4/27$ is indeed optimal. Writing T_z for the stabilization time starting from an initial configuration z, we seek to prove that $\mathbb{E}T_z \leq \frac{4}{27}N^2$. To this end, one of the key ideas is to work with a Lyapunov function $V(z)$ in lieu of the (more complicated) function $\mathbb{E}T_z$. The domain of the function V is *continuous*: a domain element describes a configuration in terms of the distances between adjacent tokens. Combinatorial arguments exploiting the highly symmetrical structure of $V(z)$ enable us to establish that, for an arbitrary configuration z,

we have $\mathbb{E}T_z \leq V(z)$, with equality holding for all three-token configurations. Finally, in what constitutes the most technically challenging part of this paper, we combine induction on the number of tokens with analytical techniques to show that V is bounded by $\frac{4}{27}N^2$. Taken together, we obtain $\mathbb{E}T_z \leq \frac{4}{27}N^2$, entailing the Herman-Protocol Conjecture.

The case of there being an *even* number K of tokens is equally natural from a mathematical point of view, although it does not correspond to a concrete bit-flipping protocol. It was established in [14] that the worst-case configuration in this variant is the equidistant *two*-token configuration, with an expected stabilization time of $\frac{1}{2}N^2$; the analysis underlying that result is considerably simpler than what is required in case the number of tokens is odd, as in the present paper.

Herman's protocol is also related to the notion of *coalescing random walks* [2, 8, 1]. There, one considers multiple independent random walks on $\mathbb{Z}^d$ (or on the vertices of a connected graph). When two walks meet, they coalesce into a new random walk. A protocol for self-stabilizing mutual exclusion based on such random walks was proposed in [20]. The expected coalescence time was studied in [7, 25, 6].

It is interesting to note that Herman's ring is closely related to widely-studied models of random walks and Brownian motion in statistical physics. Observe that by a simple modification of the formalism, one may equivalently view Herman's model as a ring in which tokens randomly move in discrete step in *any* direction, with pairwise collisions leading to annihilation; this precisely corresponds to Fisher's *vicious drunks* model [15] (with periodic boundary conditions). Similar models have been studied in chemical physics [10, 3, 28] and statistical mechanics [17, 26, 27], among others.

The rest of the paper is organized as follows. In Section 2 we review previous results in the literature that are relevant to our proof. In Section 3 we outline the structure of our proof, identifying two key lemmas, Lemma 8 and Lemma 9. Those are proved in [4] and Section 4, respectively.

Another solution of the conjecture, using different techniques, is independently shown in [9].

2 Relevant Previous Results

For the rest of the paper we fix the number N of processes. We assume that the number K of tokens is odd, and both N and K are at least 3.

Processes are numbered from 1 to N, clockwise, according to their position in the ring. A configuration with K tokens is formalized as a function $z : \{1, \ldots, K\} \to \{1, \ldots, N\}$ with $z(1) < \cdots < z(K)$, where the ith token ($i \in \{1, \ldots, K\}$) is held by the processor with the number $z(i)$. We write Z_K for the set of configurations with K tokens, and Z for the set of all possible configurations, that is, $Z = Z_1 \cup Z_3 \cup Z_5 \cup \ldots$

For a fixed initial configuration $z = z_0$ we write $(z_t)_{t\geq 0}$ for the stochastic process of configurations emanating from z. The *stabilization time* T_z is the smallest $t \geq 0$ such that $z_t \in Z_1$, i.e., the time until only one token is left. In this paper we focus on the expectation $\mathbb{E}T_z$. It is shown in [23] that if N is odd and a multiple of 3, then there is a configuration $z \in Z_3$ (with the 3 tokens maximally separated in an equilateral triangle) such that $\mathbb{E}T_z = \frac{4}{27}N^2$.

In this paper we show:

▶ **Theorem 1.** *We have $\mathbb{E}T_z \leq \frac{4}{27}N^2$ for all $z \in Z$.*

Equivalently, the Herman conjecture states that for all odd $K \geq 3$ and all $z \in Z_K$ we have $\mathbb{E}T_z \leq \frac{4}{27}N^2$. Only the case $K = 3$ was previously known [23].

The following proposition has been used in a similar form in various papers on Herman's protocol, for instance in [23, Lemma 5]. It bounds the stabilization time by a Lyapunov function V.

▶ **Proposition 2** (Bound by a Lyapunov function). *Given $z \in Z$, denote by $z' \in Z$ the random successor configuration of z. Let $V : Z \to \mathbb{R}$ be a function with*

$$\mathbb{E}(V(z') \mid z) \;\leq\; V(z) - 1 \qquad \text{for all } z \in Z \setminus Z_1, \text{ and} \tag{1}$$

$$0 \;\leq\; V(z) \qquad \text{for all } z \in Z_1. \tag{2}$$

Then $\mathbb{E}T_z \leq V(z)$ for all $z \in Z$. In particular, $V(z) \geq 0$ for all $z \in Z$.

Although this result is not new, we give a short proof based on a martingale argument. The proof is inspired by [18], and may provide some intuition.

Proof. Let $z \in Z$. Consider the stochastic process $(z_t)_{t\geq 0}$ of configurations emanating from $z = z_0$. Define $W_t := V(z_t) + t$. By (1) the process $(W_t)_{t\geq 0}$ is a supermartingale. The stabilization time $T_z = T_{z_0}$ is a stopping time with finite expectation, and the differences $|W_{t+1} - W_t|$ are bounded as the Markov chain reachable from z has finitely many states. Hence, the optional stopping theorem applies, yielding $\mathbb{E}W_{T_z} \leq \mathbb{E}W_0 = V(z)$. By definition of W_t we have $\mathbb{E}W_{T_z} = \mathbb{E}V(z_{T_z}) + \mathbb{E}T_z$. Since $z_{T_z} \in Z_1$, we have $\mathbb{E}T_z \leq \mathbb{E}W_{T_z}$ by (2). By combining the previous two inequalities, we obtain $\mathbb{E}T_z \leq V(z)$. ◀

Following [14, 18] we associate with a configuration $z \in Z_K$ the *gap vector* $\mathbf{g}(z) = (g_0, \ldots, g_{K-1}) \in \mathbb{N}^K$ by setting $g_0 := N + z(1) - z(K)$, and $g_i := z(i+1) - z(i)$ for $i \in \{1, \ldots, K-1\}$. Then $\mathbf{g}(z)/N$ lives in the so-called standard $(K-1)$-simplex $D^{(K)}$, defined by

$$D^{(K)} := \left\{ \mathbf{x} = (x_0, \ldots, x_{K-1}) \in [0,1]^K \;\mid\; x_0 + \cdots + x_{K-1} = 1 \right\}.$$

Towards a suitable Lyapunov function V we define the cubic polynomial $f_3^{(K)} : D^{(K)} \to [0, \infty)$ by

$$f_3^{(K)}(\mathbf{x}) := \sum_{\substack{0 \leq i_0 < i_1 < i_2 < K \\ i_2 - i_1,\ i_1 - i_0 \text{ odd}}} x_{i_0} x_{i_1} x_{i_2}.$$

For instance, we have $f_3^{(5)}(\mathbf{x}) = x_0x_1x_2 + x_0x_1x_4 + x_0x_3x_4 + x_1x_2x_3 + x_2x_3x_4$.

The following lemma was implicitly proved in previous works:

▶ **Lemma 3** (Lyapunov function V_3 [14, Page 240, Proof of Theorem 1] and [18, Theorem 4]). *Let $V_3 : Z \to [0, \infty)$ be defined by $V_3(z) := 4N^2 f_3^{(K)}(\mathbf{g}(z)/N)$ for $z \in Z_K$. Denote by $z' \in Z_1 \cup Z_3 \cup \ldots \cup Z_K$ the random successor configuration of $z \in Z_K$. Then $\mathbb{E}(V_3(z') \mid z) = V_3(z) - \frac{K-1}{2}$ for all $z \in Z_K$. Hence, by Proposition 2, $\mathbb{E}T_z \leq 4N^2 f_3^{(K)}(\mathbf{g}(z)/N)$.*

For $K = 3$ Lemma 3 gives $\mathbb{E}T_z \leq 4N^2 f_3^{(K)}(\mathbf{g}(z)/N) = \frac{4}{N} g_0 g_1 g_2$. In fact, for $K = 3$ it was shown before in [23] that $\mathbb{E}T_z$ is identically equal to $\frac{4}{N} g_0 g_1 g_2$, providing an exact formula for the expected stabilization time of configurations with three tokens. Lemma 3 suggests analyzing f_3:

▶ **Lemma 4** (Maximum of f_3 [14, Proof of Theorem 2], [18, Theorem 3]). *For all $K \geq 3$ odd we have*

$$\max_{\mathbf{x} \in D} f_3^{(K)}(\mathbf{x}) \;=\; f_3^{(K)}\left(\frac{1}{K}, \ldots, \frac{1}{K}\right) \;=\; \frac{1}{24}\left(1 - \frac{1}{K^2}\right).$$

By combining Lemmas 3 and 4 one obtains $\mathbb{E}T_z \leq \frac{N^2}{6}(1 - \frac{1}{K^2})$, which is the bound obtained in [14]. A slightly better bound is given in [18].

3 Proof of the Herman Conjecture

The function V_3 from Lemma 3 leaves room for improvement since $\mathbb{E}(V_3(z') \mid z) = V_3(z) - \frac{K-1}{2}$, which is strictly less than $V_3(z) - 1$ for $K > 3$. The idea for obtaining an optimal bound is to decrease the gap between $\frac{K-1}{2}$ and 1, by decreasing the Lyapunov function V. One could think that the scaled function $\frac{2}{K-1}V_3$ is also a Lyapunov function satisfying (1), but this is not true; in particular, note that the number of tokens K might be different for a configuration z and its successor z'. Since scaling does not work, we decrease the Lyapunov function by subtracting a quintic polynomial, as follows. Define a quintic polynomial $f_5^{(K)} : D^{(K)} \to [0, \infty)$, similar to $f_3^{(K)}$:

$$f_5^{(K)}(\mathbf{x}) = \sum_{\substack{0 \le i_0 < i_1 < \cdots < i_4 < K \\ i_4 - i_3, \ldots, i_1 - i_0 \text{ odd}}} x_{i_0} x_{i_1} x_{i_2} x_{i_3} x_{i_4}$$

For instance, $f_5^{(3)}(\mathbf{x}) = 0$, $f_5^{(5)}(\mathbf{x}) = x_0x_1x_2x_3x_4$, and $f_5^{(7)}(\mathbf{x}) = x_0x_1x_2x_3x_4 + x_0x_1x_2x_3x_6 + x_0x_1x_2x_5x_6 + x_0x_1x_4x_5x_6 + x_0x_3x_4x_5x_6 + x_1x_2x_3x_4x_5 + x_2x_3x_4x_5x_6$. We also define a polynomial $f^{(K)} : D^{(K)} \to [0, \infty)$:

$$f^{(K)}(\mathbf{x}) := f_3^{(K)}(\mathbf{x}) - \alpha f_5^{(K)}(\mathbf{x}) \qquad \text{with } \alpha := 24 \tag{3}$$

For example, $f^{(5)}(\mathbf{x}) = x_0x_1x_2 + x_0x_1x_4 + x_0x_3x_4 + x_1x_2x_3 + x_2x_3x_4 - \alpha x_0x_1x_2x_3x_4$. Throughout the paper we use α in the expression of $f^{(K)}$ for notational convenience. From now onwards we may drop the superscript K from the domain $D^{(K)}$ of the functions $f_3^{(K)}$, $f_5^{(K)}$ and $f^{(K)}$ to avoid notational clutter when K is understood.

The following properties of f are fundamental:

▶ **Lemma 5** (Symmetry and continuity properties). *The function f has the following properties.*

(a) *It is symmetric with respect to rotation:*

$$f(x_0, \ldots, x_{K-1}) = f(x_1, \ldots, x_{K-1}, x_0)$$

(b) *It is continuous: For $K \ge 5$ we have*

$$f^{(K)}(x_0, 0, x_2, x_3, \ldots, x_{K-1}) = f^{(K-2)}(x_0 + x_2, x_3, \ldots, x_{K-1}).$$

Analogous properties were shown for f_3 in [14]. Their proof carries over to f_5 and hence to f. The following lemma uses f to define a tighter Lyapunov function.

▶ **Lemma 6** (Lyapunov function V). *Define $V : Z \to [0, \infty)$ by $V(z) := 4N^2 f(\mathbf{g}(z)/N)$. Let $z \in Z$ and denote by z' the random successor configuration of z. Then $\mathbb{E}(V(z') \mid z) \le V(z) - 1$. Hence, by Proposition 2, $\mathbb{E}T_z \le 4N^2 f(\mathbf{g}(z)/N)$.*

We remark that a similar Lyapunov function has been investigated in [14, Equation (15)], but did not lead to a proof of the Herman conjecture. It seems that $V(z)$ needs to be chosen with great care, since even slight variations do not work.

Lemma 6 suggests analyzing f:

▶ **Lemma 7** (Maximum of f). *For all $K \ge 3$ odd we have*

$$\max_{\mathbf{x} \in D} f^{(K)}(\mathbf{x}) = \frac{1}{27}.$$

With this in hand our main result follows:

Proof of Theorem 1. Immediate by combining Lemmas 6 and 7. ◀

It remains to prove Lemmas 6 and 7.

3.1 Proof of Lemma 6

Towards Lemma 6 we show:

▶ **Lemma 8** (Lyapunov function V_5). *Define $V_5 : Z \to [0,\infty)$ by $V_5(z) := 4N^2 f_5(\mathbf{g}(z)/N)$. Let $K \geq 5$ and $z \in Z$ and denote by z' the random successor configuration of z. Then*

$$\mathbb{E}(V_5(z') \mid z) = V_5(z) + \frac{1}{32}\frac{(K-1)(K-3)}{N^2} - \frac{1}{2}(K-3) f_3\left(\frac{\mathbf{g}(z)}{N}\right).$$

The proof in [4] requires an analysis of correlations among the changes in gaps between tokens in each step of the protocol. Using Lemma 8 one can readily prove Lemma 6:

Proof of Lemma 6. For $K = 3$ the statement follows from Lemma 3. For $K \geq 5$ we have:

$$\begin{aligned}
\mathbb{E}(V(z') \mid z) &= \mathbb{E}((V_3(z') - 24V_5(z')) \mid z) && \text{by the definitions}\\
&= \mathbb{E}(V_3(z') \mid z) - 24\mathbb{E}(V_5(z') \mid z) && \text{linearity of expectation}\\
&= V_3(z) - \frac{K-1}{2} - 24V_5(z) - \frac{3}{4}\frac{(K-1)(K-3)}{N^2}\\
&\quad + 12(K-3) f_3\left(\frac{\mathbf{g}(z)}{N}\right) && \text{Lemmas 3 and 8}\\
&\leq V(z) - \frac{K-1}{2} + 12(K-3) f_3\left(\frac{\mathbf{g}(z)}{N}\right) && \text{since } K \geq 3\\
&\leq V(z) - \frac{K-1}{2} + \frac{K-3}{2} && \text{Lemma 4}\\
&= V(z) - 1
\end{aligned}$$

◀

3.2 Proof of Lemma 7

Towards Lemma 7 we show:

▶ **Lemma 9** (Local maxima of f). *Let $K \geq 5$ and odd. There is no $\mathbf{v} \in D^{(K)}$ in the interior of $D^{(K)}$ such that $\mathbf{v}$ is a local maximum and $f^{(K)}(\mathbf{v}) > \frac{1}{27}$.*

The proof in Section 4 involves a combinatorial analysis of inequalities arising from conditions on the derivatives of $f^{(K)}$. Using Lemma 9 one can readily prove Lemma 7:

Proof of Lemma 7. We proceed by induction on K. For the induction base we have $K = 3$. It is straightforward to check that the maximum of $f^{(3)}(\mathbf{x}) = f_3^{(3)}(\mathbf{x}) = x_0x_1x_2$ is $f^{(3)}(\frac{1}{3},\frac{1}{3},\frac{1}{3}) = \frac{1}{27}$.

For the induction step we have $K \geq 5$. Let $\mathbf{v} \in D^{(K)}$ with $f^{(K)}(\mathbf{v}) = \max_{\mathbf{x}\in D^{(K)}} f^{(K)}(\mathbf{x})$. If $\mathbf{v}$ is in the interior of $D^{(K)}$, then by Lemma 9 we have $f^{(K)}(\mathbf{v}) \leq \frac{1}{27}$. If $\mathbf{v}$ is at the boundary of $D^{(K)}$, then $v_i = 0$ for some i. By Lemma 5(a) we can assume that $v_1 = 0$. Using Lemma 5(b) the statement follows from the induction hypothesis. ◀

4 Proof of Lemma 9

In this section we prove Lemma 9. In Section 4.1 we state several properties that an interior local maximum of $f^{(K)}$ would have to satisfy. In Section 4.2 we prove Lemma 9 for $K = 5$ for a first taste of the general argument. In Section 4.3 we prove Lemma 9 for $K = 7$ to illustrate some fine points that occur only for larger values of K. In Section 4.4 we state some combinatorial facts needed for the general case. Finally, in Section 4.5 we prove Lemma 9.

4.1 Properties of an Interior Local Maximum

The following lemma is obtained by considering first and second derivatives of f evaluated at an interior local maximum.

▶ **Lemma 10.** *Let* $\mathbf{v}$ *be a local maximum of* $f^{(K)}$ *in the interior of* $D^{(K)}$ *and define* $c \in \mathbb{R}$ *by*

$$c \;=\; \sum_{\substack{1<i_2<K \\ i_2 \text{ even}}} v_{i_2} \;-\; \alpha \sum_{\substack{1<i_2<i_3<i_4<K \\ i_2,\, i_4 \text{ even} \\ i_3 \text{ odd}}} v_{i_2} v_{i_3} v_{i_4}\,. \tag{4}$$

This expression holds for the same value of c *if the indices are rotated by an arbitrary* k*: for all* j *the index* i_j *becomes* $(i_j + k) \bmod K$*. Further, we have*

$$\sum_{\substack{3\le i_3<i_4<K \\ i_3 \text{ odd} \\ i_4 \text{ even}}} v_{i_3} v_{i_4} \le \frac{1}{\alpha}\,. \tag{5}$$

Again, this inequality also holds when indices are rotated.

For example, for $K = 7$ we have $c = v_2 + v_4 + v_6 - \alpha(v_2v_3v_4 + v_2v_3v_6 + v_2v_5v_6 + v_4v_5v_6) = v_1 + v_3 + v_5 - \alpha(v_1v_2v_3 + v_1v_2v_5 + v_1v_4v_5 + v_3v_4v_5)$.

Proof of Lemma 10. The idea of the proof is as follows. We pick a particular direction in $D^{(K)}$, namely $\mathbf{d} = (-1, 0, 1, 0, 0, \ldots, 0)$, and consider the function $f(\mathbf{v} + \epsilon\mathbf{d})$ as a univariate function of ϵ. Since $\mathbf{v}$ is a local maximum, the first derivative must be zero and the second derivative must be nonpositive. Exploiting the fact that $v_i > 0$ for all i holds in the interior, we obtain (4) and (5), respectively. See [4] for the detailed proof. ◀

Let $S_j^{(K)}(\mathbf{x})$ denote the scalar product of $\mathbf{x}$ with a copy of itself rotated j times:

$$S_j^{(K)}(\mathbf{x}) := \sum_{i=0}^{K-1} x_i x_{i+j}$$

In all formulas it will be the case that the subscript of S is odd. Also, the superscript will be omitted when unimportant or understood from context.

▶ **Corollary 11.** *Let* $\mathbf{v}$ *be a local maximum of* $f^{(K)}$ *in the interior of* $D^{(K)}$*. Then the following inequality holds:*

$$\sum_{\substack{1\le i<K-2 \\ i \text{ odd}}} \frac{K-i-2}{2} S_i(\mathbf{v}) \quad\le\quad \frac{K}{\alpha}\,.$$

For example, for $K = 11$ we have $4S_1(\mathbf{v}) + 3S_3(\mathbf{v}) + 2S_5(\mathbf{v}) + S_7(\mathbf{v}) \le 11/\alpha$.

▶ **Lemma 12** (Bound for f_5)**.** *Suppose that* $\mathbf{v} \in D^{(K)}$ *satisfies* $f^{(K)}(\mathbf{v}) > \frac{1}{27}$*. Then* $\alpha f_5(\mathbf{v}) < \frac{1}{216}$*.*

Proof. By Lemma 4 we have $f_3(\mathbf{v}) \le \frac{1}{24}$ and hence $\alpha f_5(\mathbf{v}) = f_3(\mathbf{v}) - f(\mathbf{v}) < \frac{1}{24} - \frac{1}{27} = \frac{1}{216}$. ◀

4.2 Proof of Lemma 9 for $K = 5$

Let $K = 5$. Then

$$\begin{aligned} f(\mathbf{x}) &= f_3(\mathbf{x}) - \alpha f_5(\mathbf{x}) \\ &= x_0x_1x_2 + x_0x_1x_4 + x_0x_3x_4 + x_1x_2x_3 + x_2x_3x_4 - \alpha x_0x_1x_2x_3x_4 \end{aligned}$$

Towards a contradiction, suppose that there is a local maximum $\mathbf{v}$ with $f(\mathbf{v}) > \frac{1}{27}$ in the interior of D. By (4), the value

$$c = v_2 + v_4 - \alpha v_2v_3v_4 \tag{6}$$

is invariant under rotations. Indeed, $v_{2+k} + v_{4+k} - \alpha v_{2+k}v_{3+k}v_{4+k} \equiv c$ for all k, but we shall avoid explicitly mentioning rotations, for notational simplicity. Summing (6) over all K rotations we obtain:

$$5c = 2 - \alpha f_3(\mathbf{v}) \tag{7}$$

By (6) we have $v_0v_1c = v_0v_1v_2 + v_0v_1v_4 - \alpha f_5(\mathbf{v})$ and, summing over all K rotations,

$$cS_1(\mathbf{v}) = 2f(\mathbf{v}) - 3\alpha f_5(\mathbf{v}) \tag{8}$$

Moreover,

$$cS_1(\mathbf{v}) \overset{\text{Cor. 11}}{\leq} \frac{5c}{\alpha} \overset{(7)}{=} \frac{2}{\alpha} - f_3(\mathbf{v}) = \frac{2}{\alpha} - f(\mathbf{v}) - \alpha f_5(\mathbf{v}).$$

Combining this with (8) gives:

$$\frac{2}{\alpha} \geq 3f(\mathbf{v}) - 2\alpha f_5(\mathbf{v}) \overset{\text{Lemma 12}}{\geq} \frac{3}{27} - 2 \cdot \frac{1}{216}$$

This implies $\alpha \leq 216/11 \approx 19.6$, which is a contradiction as required (since $\alpha = 24$). ◀

4.3 Proof of Lemma 9 for $K = 7$

Let $K = 7$. Towards a contradiction, we suppose again that there is a local maximum $\mathbf{v}$ with $f(\mathbf{v}) > \frac{1}{27}$ in the interior of D. By (4), all K rotations of the following hold with the same $c \in \mathbb{R}$:

$$c = v_2 + v_4 + v_6 - \alpha(v_2v_3v_4 + v_2v_3v_6 + v_2v_5v_6 + v_4v_5v_6) \tag{9}$$

Summing (9) over K rotations we obtain:

$$7c = 3 - 2\alpha f_3(\mathbf{v}) \tag{10}$$

By (9) we have

$$v_0v_1c = v_0v_1v_2 + v_0v_1v_4 + v_0v_1v_6 - \alpha(v_0v_1v_2v_3v_4 + v_0v_1v_2v_3v_6 + v_0v_1v_2v_5v_6 + v_0v_1v_4v_5v_6) \tag{11}$$

and

$$\begin{aligned} v_0v_3c &= v_0v_3v_4 + v_0v_3v_6 - \alpha v_0v_3v_4v_5v_6 + v_0v_2v_3(1 - \alpha(v_3v_4 + v_3v_6 + v_5v_6)) \\ &\geq v_0v_3v_4 + v_0v_3v_6 - \alpha v_0v_3v_4v_5v_6 \end{aligned} \tag{12}$$

where the last inequality is by (5). Summing (11) and (12) over K rotations we obtain:

$$c\big(2S_1(\mathbf{v}) + S_3(\mathbf{v})\big) \;\geq\; 4f_3(\mathbf{v}) - 9\alpha f_5(\mathbf{v}) = 4f(\mathbf{v}) - 5\alpha f_5(\mathbf{v}) \tag{13}$$

Further we have:

$$c\big(2S_1(\mathbf{v}) + S_3(\mathbf{v})\big) \overset{\text{Cor. 11}}{\leq} \frac{7c}{\alpha} \overset{(10)}{=} \frac{3}{\alpha} - 2f_3(\mathbf{v}) \;=\; \frac{3}{\alpha} - 2f(\mathbf{v}) - 2\alpha f_5(\mathbf{v})$$

Combining this with (13) gives:

$$\frac{3}{\alpha} \;\geq\; 6f(\mathbf{v}) - 3\alpha f_5(\mathbf{v}) \overset{\text{Lemma 12}}{\geq} \frac{6}{27} - 3\cdot\frac{1}{216}$$

This leads to $\alpha \leq 14.4$, which is a contradiction as desired. ◀

4.4 Combinatorial Lemmas

In order to generalize the proofs from Sections 4.2 and 4.3 to any odd K, we state some combinatorial lemmas in this subsection. They are proved in [4].

In order to generalize (7) and (10) we show the following lemma:

▶ **Lemma 13.** *We have:*

$$\sum_{k=0}^{K-1} \sum_{\substack{1<i'_0<i'_1<i'_2<K \\ i'_0,\ i'_2\ even \\ i'_1\ odd}} x_{i'_0+k}x_{i'_1+k}x_{i'_2+k} = \frac{K-3}{2} \sum_{\substack{0\leq i_0<i_1<i_2<K \\ i_2-i_1,\ i_1-i_0\ odd}} x_{i_0}x_{i_1}x_{i_2} = \frac{K-3}{2} f_3^{(K)}(\mathbf{x})\,.$$

For example, if $K = 5$, then we obtain that summing the 5 rotations of $x_2x_3x_4$ gives $f_3^{(5)}(\mathbf{x})$. As another example, if $K = 7$, then we obtain that summing the 7 rotations of $x_2x_3x_4 + x_2x_3x_6 + x_2x_5x_6 + x_4x_5x_6$ gives $2f_3^{(7)}(\mathbf{x})$. These two instances of Lemma 13 help establish (7) and (10).

In order to generalize the inequality in (12) we need the following lemma:

▶ **Lemma 14.** *Let* $\mathbf{v}$ *be a local maximum of* $f^{(K)}$ *in the interior of* $D^{(K)}$*. If* i_1 *is odd and* $0 < i_1 < K$*, then the following inequality holds:*

$$v_0v_{i_1}\Bigg(\sum_{\substack{1<i_2<K \\ i_2\ even}} v_{i_2} - \sum_{\substack{1<i_2<i_3<i_4<K \\ i_2,i_4\ even \\ i_3\ odd}} \alpha v_{i_2}v_{i_3}v_{i_4}\Bigg) \geq v_0v_{i_1}\Bigg(\sum_{\substack{i_1<i_2<K \\ i_2\ even}} v_{i_2} - \sum_{\substack{i_1<i_2<i_3<i_4<K \\ i_2,i_4\ even \\ i_3\ odd}} \alpha v_{i_2}v_{i_3}v_{i_4}\Bigg)\,.$$

The inequality says that if we drop those terms that do not occur in $f_3^{(K)}$ or $f_5^{(K)}$, then we obtain a lower bound. The proof groups those terms that are not in either of $f_3^{(K)}$ or $f_5^{(K)}$, and then invokes (5) to show that their sum is nonnegative.

In order to generalize (8) and (13) we need Corollary 16 below, which is a consequence of the following lemma:

▶ **Lemma 15.** *Let* l *be an odd, positive integer. Then:*

$$\sum_{k=0}^{K-1} \sum_{\substack{1\leq i'_1<K-2 \\ i'_1\ odd}} \frac{K-i'_1-2}{2} \sum_{\substack{i'_1<i'_2<\cdots<i'_{l-1}<K \\ \forall j,\ i'_j\equiv j \pmod 2}} x_k x_{i'_1+k} \prod_{1<j<l} x_{i'_j+k} =$$

$$= \Big(\frac{l-1}{2}K - l\Big) \sum_{\substack{0\leq i_0<\cdots<i_{l-1}<K \\ i_j-i_{j-1}\ odd\ for\ 0<j<l}} \prod_{j=0}^{l-1} x_{i_j}$$

For example, if $K = 5$ and $l = 3$, then we have that summing 5 rotations of $x_0x_1x_2 + x_0x_1x_4$ gives $2f_3^{(5)}(\mathbf{x})$. As another example, if $K = 9$ and $l = 3$, then summing 9 rotations of $3x_0x_1(x_2 + x_4 + x_6 + x_8) + 2x_0x_3(x_4 + x_6 + x_8) + x_0x_5(x_6 + x_8)$ gives $6f_3^{(9)}(\mathbf{x})$.

▶ **Corollary 16.** *We have:*

$$\sum_{k=0}^{K-1} \sum_{\substack{1 \le i_1 < K-2 \\ i_1 \text{ odd}}} \frac{K - i_1 - 2}{2} \sum_{\substack{i_1 < i_2 < K \\ i_2 \text{ even}}} x_{0+k}x_{i_1+k}x_{i_2+k} = (K-3)f_3^{(K)}(\mathbf{x})$$

and also

$$\sum_{k=0}^{K-1} \sum_{\substack{1 \le i_1 < K-2 \\ i_1 \text{ odd}}} \frac{K - i_1 - 2}{2} \sum_{\substack{i_1 < i_2 < i_3 < i_4 < K \\ i_2, i_4 \text{ even} \\ i_3 \text{ odd}}} x_{0+k}x_{i_1+k}x_{i_2+k}x_{i_3+k}x_{i_4+k} = (2K-5)f_5^{(K)}(\mathbf{x})\,.$$

Proof. Instantiate Lemma 15 with $l = 3$ and, respectively, $l = 5$. ◀

4.5 Proof of Lemma 9

Towards a contradiction, suppose that there is a local maximum $\mathbf{v}$ with $f(\mathbf{v}) > \frac{1}{27}$ in the interior of D, i.e., $v_i > 0$ for all $i \in \{0, \ldots, K-1\}$. Summing up the K rotations of (4) and using Lemma 13, we obtain:

$$Kc \;=\; \frac{K-1}{2} - \frac{K-3}{2}\alpha f_3(\mathbf{v}) \tag{14}$$

Multiplying (4) on both sides by $\sum_{\substack{1 \le i_1 < K-2 \\ i_1 \text{ odd}}} \frac{K-i_1-2}{2} v_0v_{i_1}$ we obtain:

$$\begin{aligned} c \sum_{\substack{1 \le i_1 < K-2 \\ i_1 \text{ odd}}} \frac{K - i_1 - 2}{2} v_0v_{i_1} &= \sum_{\substack{1 \le i_1 < K-2 \\ i_1 \text{ odd}}} \frac{K - i_1 - 2}{2} v_0v_{i_1} \left(\sum_{\substack{1 < i_2 < K \\ i_2 \text{ even}}} v_{i_2} - \sum_{\substack{1 < i_2 < i_3 < i_4 < K \\ i_2,\, i_4 \text{ even} \\ i_3 \text{ odd}}} \alpha v_{i_2}v_{i_3}v_{i_4} \right) \\ &\ge \sum_{\substack{1 \le i_1 < K-2 \\ i_1 \text{ odd}}} \frac{K - i_1 - 2}{2} v_0v_{i_1} \left(\sum_{\substack{i_1 < i_2 < K \\ i_2 \text{ even}}} v_{i_2} - \sum_{\substack{i_1 < i_2 < i_3 < i_4 < K \\ i_2,\, i_4 \text{ even} \\ i_3 \text{ odd}}} \alpha v_{i_2}v_{i_3}v_{i_4} \right) \end{aligned}$$

using Lemma 14. Summing K rotations of this inequality yields:

$$\begin{aligned} c \sum_{\substack{1 \le i_1 < K-2 \\ i_1 \text{ odd}}} \frac{K - i_1 - 2}{2} S_{i_1}(\mathbf{v}) \;&\ge\; (K-3)f_3(\mathbf{v}) - (2K-5)\alpha f_5(\mathbf{v}) \\ &= (K-3)f(\mathbf{v}) - (K-2)\alpha f_5(\mathbf{v}) \end{aligned} \tag{15}$$

using Corollary 16. Further we have:

$$c \sum_{\substack{1 \le i_1 < K-2 \\ i_1 \text{ odd}}} \frac{K - i_1 - 2}{2} S_{i_1}(\mathbf{v}) \overset{\text{Cor. } 11}{\le} \frac{Kc}{\alpha} \overset{(14)}{=} \frac{K-1}{2\alpha} - \frac{K-3}{2} f_3(\mathbf{v})\,.$$

Combining this with (15) gives:

$$\frac{K-1}{2\alpha} \;\ge\; \frac{3K-9}{2} f(\mathbf{v}) - \frac{K-1}{2}\alpha f_5(\mathbf{v}) \overset{\text{Lemma } 12}{\ge} \frac{K-3}{2} \cdot \frac{1}{9} - \frac{K-1}{2} \cdot \frac{1}{216}\,.$$

This implies

$$\alpha \leq \frac{216(K-1)}{23K-71} < 19.7\,.$$

Since $\alpha = 24$, this leads to a contradiction as desired. ◀

5 Conclusions

In this paper we have proved the Herman-Protocol Conjecture formulated by McIver and Morgan in [23] a decade ago, which says that the worst-case initial configuration consists of three maximally-separated tokens, for N multiple of 3. This follows from our result that the worst-case self-stabilization time is at most $\frac{4}{27}N^2$, for any number of processes N and any odd number of tokens K.

The proof uses a Lyapunov function approach. To do so, we first find a suitable Lyapunov function and then show that its maximum is $\frac{4}{27}N^2$. Then we show that this function gives an upper bound for the self-stabilization time for *each* possible configuration in Herman's algorithm.

References

1 D. Aldous and J. A. Fill. Reversible Markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/~aldous/RWG/book.html`.

2 R. Arratia. Limiting point processes for rescalings of coalescing and annihilating random walks on Z^d. *The Annals of Probability*, 9(6):909–936, 1981.

3 D. Balding. Diffusion-reaction in one dimension. *J. Appl. Prob.*, 25:733–743, 1988.

4 M. Bruna, R. Grigore, S. Kiefer, J. Ouaknine, and J. Worrell. Proving the Herman-Protocol Conjecture. Technical report, arxiv.org, 2015. Available at `http://arxiv.org/abs/1504.01130`.

5 PRISM case studies. Randomised self-stabilising algorithms. `http://www.prismmodelchecker.org/casestudies/self-stabilisation.php`.

6 C. Cooper, R. Elsässer, H. Ono, and T. Radzik. Coalescing random walks and voting on graphs. In *Proc. PODC*, pages 47–56. ACM, 2012.

7 D. Coppersmith, P. Tetali, and P. Winkler. Collisions among random walks on a graph. *SIAM Journal on Discrete Mathematics*, 6(3):363–374, 1993.

8 J.T. Cox. Coalescing random walks and voter model consensus times on the torus in Z^d. *The Annals of Probability*, 17(4):1333–1366, 1989.

9 E. Csóka and S. Mészáros. Generalized solution for the Herman protocol conjecture. Technical report, arxiv.org, 2015. Available at `http://arxiv.org/abs/1504.06963`.

10 P.-G. de Gennes. Soluble model for fibrous structures with steric constraints. *J. Chem. Phys.*, 48(5):2257–2259, 1968.

11 E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Comm. ACM*, 17(11):643–644, 1974.

12 S. Dolev. *Self-Stabilization*. MIT Press, 2000.

13 Y. Feng and L. Zhang. A Tighter Bound for the Self-Stabilization Time in Herman's Algorithm. *Inf. Process. Lett.*, 113(13):486–488, 2013.

14 Y. Feng and L. Zhang. A nearly optimal upper bound for the self-stabilization time in Herman's algorithm. *Dist. Comp.*, pages 1–12, 2015.

15 M. E. Fisher. Walks, walls, wetting, and melting. *J. Stat. Phys.*, 34(5-6):667–729, 1984.

16 L. Fribourg, S. Messika, and C. Picaronny. Coupling and self-stabilization. *Dist. Comp.*, 18:221–232, 2005.

17 S. Y. Grigoriev and V. B. Priezzhev. Random walk of annhilating particles on the ring. *Theor. Math. Phys.*, 146(3):411–420, 2006.

18 J. Haslegrave. Bounds on Herman's algorithm. *Theoretical Computer Science*, 550:100–06, 2014.

19 T. Herman. Probabilistic self-stabilization. *Inf. Process. Lett.*, 35(2):63–67, 1990.

20 A. Israeli and M. Jalfon. Token management schemes and random walks yield self-stabilizing mutual exclusion. In *Proc. PODC*, pages 119–131. ACM, 1990.

21 S. Kiefer, A. Murawski, J. Ouaknine, J. Worrell, and L. Zhang. On stabilization in Herman's algorithm. In *Proc. ICALP*, volume 6756 of *LNCS*. Springer, 2011.

22 L. Lamport. Solved problems, unsolved problems and non-problems in concurrency. In *Proc. PODC*, pages 1–11. ACM, 1984.

23 A. McIver and C. Morgan. An elementary proof that Herman's ring is $\Theta(N^2)$. *Inf. Process. Lett.*, 94(2):79–84, 2005.

24 T. Nakata. On the expected time for Herman's probabilistic self-stabilizing algorithm. *Theor. Comput. Sci.*, 349(3):475–483, 2005.

25 R.I. Oliveira. On the coalescence time of reversible random walks. *Trans. Amer. Math. Soc.*, 364(4):2109–2128, 2012.

26 J. Rambeau and G. Schehr. Distribution of the time at which N vicious walkers reach their maximal height. *Phys. Rev. E*, 83, 2011.

27 G. Schehr, S. N. Majumdar, A. Comtet, and P. J. Forrester. Reunion probability of N vicious walkers: Typical and large fluctuations for large N. *J. Stat. Phys.*, 150:491–530, 2013.

28 M. Warner. Aggregation in dense solutions of rods. *J. Chem. Soc. Faraday. Trans.*, 87(6):861–867, 1991.

A Polynomial-Time Algorithm for Reachability in Branching VASS in Dimension One

Stefan Göller[*1], **Christoph Haase**[†2], **Ranko Lazić**[‡3], **and Patrick Totzke**[§4]

1 **LSV, CNRS & ENS Cachan, Université Paris-Saclay, Paris, France**
goeller@lsv.ens-cachan.fr
2 **LSV, CNRS & ENS Cachan, Université Paris-Saclay, Paris, France**
haase@lsv.ens-cachan.fr
3 **DIMAP, Department of Computer Science, University of Warwick, Warwick, United Kingdom**
r.s.lazic@warwick.ac.uk
4 **DIMAP, Department of Computer Science, University of Warwick, Warwick, United Kingdom**
p.totzke@warwick.ac.uk

Abstract

Branching VASS (BVASS) generalise vector addition systems with states by allowing for special branching transitions that can non-deterministically distribute a counter value between two control states. A run of a BVASS consequently becomes a tree, and reachability is to decide whether a given configuration is the root of a reachability tree. This paper shows P-completeness of reachability in BVASS in dimension one, the first decidability result for reachability in a subclass of BVASS known so far. Moreover, we show that coverability and boundedness in BVASS in dimension one are P-complete as well.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases branching vector addition systems, reachability, coverability, boundedness

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.105

1 Introduction

Vector addition systems with states (VASS), equivalently known as Petri nets, are a fundamental model of computation which comprise a finite-state controller with a finite number of counters ranging over the naturals. The number of counters is usually refereed to as the dimension of the VASS. A configuration $q(\boldsymbol{n})$ of a VASS in dimension d consists of a control state q and a valuation $\boldsymbol{n} \in \mathbb{N}^d$ of the counters. A transition of a VASS can increment and decrement counters and is enabled in a configuration whenever the resulting counter values are all non-negative, otherwise the transition is disabled. Consequently, VASS induce an infinite transition system. Three of the most fundamental decision problems for VASS are reachability, coverability and boundedness. Given a target configuration $q(\boldsymbol{n})$ and some

* Supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS.
† Supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS.
‡ Supported by the EPSRC, grants EP/M011801/1 and EP/M027651/1.
§ Supported by the EPSRC, grants EP/M011801/1 and EP/M027651/1.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 105; pp. 105:1–105:13

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

initial configuration, reachability is to decide whether starting in the initial configuration there exists a path ending in $q(\boldsymbol{n})$ in the induced infinite transition system. Coverability asks whether some configuration $q(\boldsymbol{n}')$ can be reached for some $\boldsymbol{n}' \geq \boldsymbol{n}$, where $\geq$ is defined component-wise. Boundedness is the problem to decide whether there are infinitely many different configurations reachable from a given starting configuration. Those decision problems find a plethora of applications, for instance in the verification of concurrent programs. Coverability can, for example, be used in order to validate mutual exclusion properties of shared-memory concurrent programs [6]; reachability is a key underlying decision problem in the verification of liveness properties of finite-data asynchronous programs [5]. Even though the complexity of coverability and boundedness are well-understood and known to be EXPSPACE-complete [12, 14], the precise complexity of reachability remains a major unsolved problem; a non-primitive recursive upper bound ($\mathbf{F}_{\omega^3}$) has only recently been established [11] and the best known lower bound is EXPSPACE [12].

The situation is even more dissatisfying when considering *branching extensions* of VASS. Such *branching VASS (BVASS)* are additionally equipped with special branching transitions of the form (q, p, p'). When in a configuration $q(\boldsymbol{n})$, a BVASS can simultaneously non-deterministically branch into configurations $p(\boldsymbol{m})$ and $p'(\boldsymbol{m}')$ such that $\boldsymbol{n} = \boldsymbol{m} + \boldsymbol{m}'$. Reachability of a configuration $q(\boldsymbol{n})$ then is to decide whether there exists a proof tree whose root is labelled with $q(\boldsymbol{n})$ and whose leaves are all labelled with designated target control states in which all counters have value zero; coverability and boundedness are defined analogously as above. While coverability and boundedness are known to be 2-EXPTIME-complete [3], reachability in BVASS is not known to be decidable, not even in any fixed dimension. Recently, non-elementary lower bounds for reachability in BVASS have been obtained [10]. Reachability in BVASS is closely related and in fact equivalent to decidability of the multiplicative-exponential fragment of linear logic [2], and also an underlying decision problem in various other applications for instance in computational linguistics, cryptographic protocol verification, data logics and concurrent program verification; see [10] for more details.

The primary contribution of this paper is to provide a polynomial-time algorithm for reachability in BVASS in dimension one (BVASS_1) and to show that reachability is in fact P-complete. To the best of our knowledge, we give the first decidability result for reachability in a fragment of BVASS. Let us remark that a decidability result, in particular with such low complexity is actually quite surprising. On the one hand, due to the infinite state space of BVASS_1 it is not immediate that reachability is decidable. In particular, the emptiness problem for conjunctive grammars over a unary alphabet, which can be seen as a slight generalisation of BVASS_1 with special alternating transitions that can simultaneously branch into two control states while retaining the same counter value (known as ABVASS_1), is undecidable [9]. On the other hand, if we disallow branching rules in ABVASS_1 and thus obtain AVASS_1 then reachability is PSPACE-complete [15, 8].

Due to the presence of only one single counter, it is possible to establish a small-model property and to show that if a configuration is reachable in a BVASS_1 then there exists a so-called reachability tree of exponential size. What causes a main challenge when establishing a polynomial-time algorithm is that this bound is optimal in the sense that, as we show in Section 3, there exist families of BVASS_1 whose reachability trees are inherently of exponential size, and which also contain an exponential number of different counter values. Consequently, reachability cannot be witnessed in polynomial time by explicitly constructing a witnessing reachability tree. Instead, in Section 4 we show that polynomial-time computable certificates for the reachability of a configuration suffice. These certificates have two parts: the first is

a table that, for certain $d > 0$ contains those pairs of control states q and residue classes r modulo d such that $q(n)$ is reachable for some sufficiently large n with $n \equiv r \bmod d$. This is called residue reachability and described in Section 4.1. The second part, described in Section 4.2, is a compressed collection of incomplete small reachability trees, so-called expandable partial reachability trees, whose leaves are either accepting configurations or have some ancestor node with the same control state and a strictly smaller counter. In the latter case, the corresponding subtree can be repeated arbitrarily often, which leaves some configuration with an arbitrarily large counter value in a certain residue class. This eventually enables us to witness the existence of a reachability tree via residue reachability.

In Section 5, we show that coverability and boundedness are P-complete for BVASS_1. For coverability, the upper bound follows easily via a reduction to reachability. For boundedness, this is not the case and we require a specifically tailored argument.

Due to space constraints, the proofs of some statements can be found in the technical report accompanying this article [7].

2 Preliminaries

We write $\mathbb{Z}$ and $\mathbb{N}$ for the sets of integers and non-negative integers, respectively, and define $[i,j] \stackrel{\text{def}}{=} \{i, i+1, \ldots, j-1, j\}$, for given integers $i < j$. For $d \geq 1$ we define $\mathbb{Z}_d \stackrel{\text{def}}{=} [0, d-1]$.

The set of finite words over alphabet A is denoted by A^* and the length of a word $w \in A^*$ is written as $|w|$. For two words $u, v \in A^*$, we say u is a *prefix* of v (written as $u \preceq v$) if $v = uw$ for some $w \in A^*$. It is a *strict prefix* ($u \prec v$) if $u \preceq v$ and $u \neq v$. We say u and v are *incomparable* if neither $u \preceq v$ nor $v \preceq u$. A set $U \subseteq A^*$ is *prefix-closed* if for all $u \in U$ and all $v \in A^*$ we have that $v \preceq u$ implies $v \in U$.

Let Σ be a set. A Σ-*labelled (finite) tree* is a mapping $T \colon U \to \Sigma$ where $U \subseteq A^*$ is a non-empty finite prefix-closed set of *nodes* for some finite set A. For $V \subseteq U$, we define $T(V) \stackrel{\text{def}}{=} \{T(v) \mid v \in V\}$. A *leaf* of T is a node $u \in U$ such that there is no $v \in U$ with $u \prec v$; every node of T that is not a leaf is called *inner node*. A node u is an *ancestor* (resp. *descendant*) of a node v if $u \preceq v$ (resp. $v \preceq u$) and a *strict ancestor* (resp. *strict descendant*) if $u \prec v$ (resp. $v \prec u$). For any node u we define the *subtree of T rooted at u* as $T^{\downarrow u} \colon u^{-1}U \to \Sigma$, where $u^{-1}U \stackrel{\text{def}}{=} \{x \in A^* \mid ux \in U\}$ and $T^{\downarrow u}(x) \stackrel{\text{def}}{=} T(ux)$. Note that $u^{-1}U$ is a prefix-closed subset of A. We define $h(u) \stackrel{\text{def}}{=} \max\{|x| \mid x \in u^{-1}U\}$ to be the *height* of the subtree rooted at u and and define $h(T) \stackrel{\text{def}}{=} h(\varepsilon)$. Note that $h(u) = 0$ if, and only if, u is a leaf. We say T is *binary* if $U \subseteq \{0,1\}^*$; in this case if for some node $u \in U$ we have that $u0 \in U$, then $u0$ the *left child* of u and if $u1 \in U$ we say that $u1$ is the *right child* of u.

2.1 Branching Vector Addition Systems

In the following, $\boldsymbol{n}$ and $\boldsymbol{z}$ will denote elements from $\mathbb{N}^k$ and $\mathbb{Z}^k$, respectively; addition on $\mathbb{Z}^k$ is defined component-wise.

▶ **Definition 1.** Let $k \geq 1$. A *k-dimensional branching vector addition system with states (BVASS_k)* is a tuple $\mathcal{B} = (Q, \Delta, F)$ where Q is a finite set of *control states*, $\Delta \subseteq Q^3 \cup (Q \times \{-1,0,1\}^k \times Q)$ is a finite set of *transitions*, and $F \subseteq Q$ is a set of *final states*. The *size* $|\mathcal{B}|$ of a BVASS is defined as $|\mathcal{B}| \stackrel{\text{def}}{=} |Q| + k \cdot |\Delta|$.

The semantics of BVASS is given in terms of reachability trees. A *partial reachability tree* of a BVASS_k $\mathcal{B}$ is a $Q \times \mathbb{N}^k$-labelled binary tree $T \colon U \to Q \times \mathbb{N}^k$, where each inner node $u \in U$ with $T(u) = (q, \boldsymbol{n})$ satisfies exactly one of the following conditions:

Figure 1 Illustration of the BVASS$_1$ $\mathcal{B}_n$. The reachability set of the control state q_n is the singleton set $\{2^n\}$, and a reachability tree for $q(0)$ contains all counter values between 0 and 2^n.

- $u0, u1 \in U$, and if $T(u0) = (p, \boldsymbol{n}_0)$ and $T(u1) = (p', \boldsymbol{n}_1)$, then $\boldsymbol{n} = \boldsymbol{n}_0 + \boldsymbol{n}_1$ and $(q, p, p') \in \Delta$; or
- $u0 \in U, u1 \notin U$, and if $T(u0) = (p, \boldsymbol{n}_0)$, then $\boldsymbol{n}_0 = \boldsymbol{n} + \boldsymbol{z}$ and $(q, \boldsymbol{z}, p) \in \Delta$.

Note that in the second condition, counter values can be seen as being propagated top down. A *reachability tree* is a partial reachability tree T where $T(u) \in F \times \{0\}^k$ for all leaves u of T. We call these nodes *accepting* nodes. For each $j \in \mathbb{N}$ we say that a partial reachability tree T is *j-bounded* if $T(u) \in Q \times [0, j]^k$ for all $u \in U$. We call $Q \times \mathbb{N}^k$ the set of *configurations* of $\mathcal{B}$ and for the sake of readability often write its elements $(q, \boldsymbol{n})$ as $q(\boldsymbol{n})$. We say that a configuration $q(\boldsymbol{n})$ is *reachable* if there exists a reachability tree T with $T(\varepsilon) = q(\boldsymbol{n})$. Note that in particular every configuration in $F \times \{0\}^k$ is reachable. The reachability set *reach*(q) of a control state q is defined as $\mathit{reach}(q) \stackrel{\text{def}}{=} \{\boldsymbol{n} \in \mathbb{N} \mid q(\boldsymbol{n}) \text{ is reachable}\}$. The decision problem that we mainly focus on in this paper is *reachability*, defined as follows:

Reachability in BVASS$_k$

INPUT: A BVASS$_k$ $\mathcal{B} = (Q, \Delta, F)$, a control state q and $\boldsymbol{n} \in \mathbb{N}^k$ encoded in unary.
QUESTION: Is $q(\boldsymbol{n})$ reachable?

Our main result is that reachability is P-complete in dimension one.

▶ **Theorem 2.** *Reachability in BVASS$_1$ is* P*-complete.*

3 Lower Bounds

As a warm-up exercise and in order to familiarise ourselves with BVASS$_1$, we begin with proving a couple of lower bounds for the reachability problem. First, it is not difficult to see that the reachability problem is P-hard via a reduction from the monotone circuit value problem (MCVP) [13]. By simulating $\vee$-gates of a Boolean by non-deterministic branching and $\wedge$-gates by splitting transitions, the following statement can easily be obtained.

▶ **Proposition 3.** *Let $\mathcal{C}$ be a Boolean circuit. There exists a logspace computable BVASS$_1$ $\mathcal{B}$ with a control state q such that $q(0)$ is reachable if, and only if, $\mathcal{C}$ evaluates to true.*

A challenging aspect when providing a polynomial-time upper bound for reachability in BVASS$_1$ is that reachability trees may be of exponential size and may contain an exponential number of nodes labelled with distinct counter values. To see this, consider the family $(\mathcal{B}_n)_{n \geq 0}$ of BVASS$_1$, where $\mathcal{B}_n \stackrel{\text{def}}{=} (Q_n, \Delta_n, F)$ and where $Q_n \stackrel{\text{def}}{=} \{q, q_f\} \cup \{q_0, \ldots, q_n\}$, $\Delta_n \stackrel{\text{def}}{=} \{(q, +1, q), (q, 0, q_n)\} \cup \{(q_i, q_{i-1}, q_{i-1}) \mid 0 < i \leq n\} \cup \{(q_0, -1, q_f)\}$ and $F \stackrel{\text{def}}{=} \{q_f\}$. The construction is illustrated in Figure 1. It is easily seen that $q_i(N)$ is reachable if, and only if, $N = 2^i$. Observe that $\mathit{reach}(q) = \{0, \ldots, 2^n\}$ is finite and that the reachability tree of $q(0)$ contains all counter values between 0 and 2^n. In particular, this allows us to obtain the following hardness result in which the updates of the BVASS$_1$ are from $\{-1, 0, +1\}$ (i.e. encoded in unary), but the initial configuration is given in binary, via a straight-forward reduction from the NP-complete Subset Sum problem [13].

▶ **Proposition 4.** *Reachability in $BVASS_1$ is* NP*-hard if the initial configuration $q(n)$ is given in binary.*

It is worth mentioning that the previous lemma enables us to derive as a corollary an NP-lower bound for reachability in BVASS_2. This is in contrast to VASS where there is no difference between the NL-completeness of reachability in dimensions one and two [16, 4].

▶ **Corollary 5.** *Reachability in $BVASS_2$ is* NP*-hard.*

4 Reachability in BVASS_1

Here, we show that reachability in BVASS_1 is decidable in polynomial time, thereby establishing the P upper bound claimed in Theorem 2. In the first part, we consider a variation of the reachability problem in which we are only interested in reaching configurations that are sufficiently large and lie in a certain residue class. Subsequently, we will apply this intermediate result for showing that reachability can be witnessed by small partial reachability trees. Finally, we put everything together in order to obtain a polynomial-time algorithm.

4.1 The Residue Reachability Problem

A cornerstone of our algorithm for reachability in BVASS_1 is the polynomial-time decidability of the following variant of the reachability problem for BVASS_1:

Residue Reachability for BVASS_1

INPUT: A BVASS_1 $\mathcal{B} = (Q, \Delta, F)$, a configuration $q_0(n_0)$ and $d \geq 1$, where n_0 and d are given in unary.

QUESTION: Does there exist some $n \geq n_0$ such that $q_0(n)$ is reachable and $n \equiv n_0 \bmod d$?

The main result of this section is that residue reachability for BVASS_1 is decidable in polynomial time. Notice that setting $d = 1$ allows for checking whether there exists some $n \geq n_0$ such that $q(n)$ is reachable. We first introduce some auxiliary definitions that allow us to abstract away concrete counter values of reachability trees. A *partial d-residue tree* is a binary tree $T\colon U \to Q \times \mathbb{Z}_d$, where each inner node $u \in U$ with $T(u) = (q, n)$ satisfies precisely one of the following conditions:

- **(i)** $u0, u1 \in U$, and if $T(u0) = (p, m_0)$ and $T(u1) = (p', m_1)$ then $n \equiv m_0 + m_1 \bmod d$ and $(q, p, p') \in \Delta$;
- **(ii)** $u0 \in U, u1 \notin U$, and if $T(u0) = (p, m)$ then $m = n + z \bmod d$ and $(q, z, p) \in \Delta$.

We call a configuration from $Q \times \mathbb{Z}_d$ a *residue configuration*. Given a set of configurations S, its *residue* is $S/\mathbb{Z}_d \stackrel{\text{def}}{=} \{(q, n \bmod d) \in Q \times \mathbb{Z}_d \mid q(n) \in S\}$. Likewise, given a partial reachability tree $T\colon U \to Q \times \mathbb{N}$, the *residue* $T/\mathbb{Z}_d$ *of* T is $T/\mathbb{Z}_d\colon U \to Q \times \mathbb{Z}_d$, where $T/\mathbb{Z}_d(u) \stackrel{\text{def}}{=} T(u)/\mathbb{Z}_d$ for all $u \in U$. Clearly, $T/\mathbb{Z}_d$ is a partial residue tree.

For the remainder of this section, fix some BVASS_1 $\mathcal{B} = (Q, \Delta, F)$, some configuration $q_0(n_0)$ and some $d \geq 1$, where n_0 and d are given in unary. In order to decide residue reachability, one might be tempted to start with an initial configuration and then to repeatedly apply transitions of $\mathcal{B}$ modulo d until the desired residue configuration is discovered. Such an approach would, however, not be sound as it may lead to residue configurations that, informally speaking, can only be obtained by forcing the counter to drop below zero. Also, the simple alternative of constructing a sufficiently large reachability tree is futile as it may be of exponential size, cf. Section 3. In order to balance between those two extremes, we introduce reachability trees in which all nodes except of the root are required to be bounded

by some value $j \in \mathbb{N}$: a partial reachability tree $T\colon U \to Q \times \mathbb{N}$ is *almost j-bounded* if $T(u) \in Q \times [0, j]$ for all $u \in U \setminus \{\varepsilon\}$. Note that every j-bounded partial reachability tree is almost j-bounded. The following constant will be particularly useful:

$$N \stackrel{\text{def}}{=} |Q| \cdot d.$$

Moreover, by S we denote the set of configurations for which there exists an (n_0+N)-bounded reachability tree and define for $i < j$:

$$S \stackrel{\text{def}}{=} \{(q, m) \in Q \times \mathbb{N} \mid q(m) \text{ has an } (n_0 + N)\text{-bounded reachability tree}\}$$
$$S[i, j] \stackrel{\text{def}}{=} S \cap Q \times [i, j].$$

▶ **Lemma 6.** *The set S is computable in polynomial time.*

For any set of residue configurations (modulo d) $V, W \subseteq Q \times \mathbb{Z}_d$, we define the following sets that contain the result of an application of a transition of $\mathcal{B}$ modulo d:

$$\Delta(V) \stackrel{\text{def}}{=} \{(q, r - z \bmod d) \mid (q, z, p) \in \Delta, (p, r) \in V\}$$
$$\Delta(V, W) \stackrel{\text{def}}{=} \{(q, r_0 + r_1 \bmod d) \mid (q, p_0, p_1) \in \Delta, (p_0, r_0) \in V, (p_1, r_1) \in W\}.$$

Next, we inductively define a sequence of sets $R_i \subseteq Q \times \mathbb{Z}_d$ for $i \geq 0$ whose fixed point will allow for deciding residue reachability. The set R_0 consists of those pairs of control states and residue classes that can be witnessed by a reachability tree that is almost $(n_0 + N)$-bounded and whose root has a counter value at least $n_0 + N$, and the R_i for $i > 0$ are obtained by application of Δ :

$$R_0 \stackrel{\text{def}}{=} \{(q, n \bmod d) \in Q \times \mathbb{Z}_d \mid n \geq n_0 + N,\ q(n) \text{ has an almost } (n_0 + N)\text{-bounded reachability tree}\}$$
$$R_{i+1} \stackrel{\text{def}}{=} R_i \cup \Delta(R_i) \cup \Delta(R_i, S/\mathbb{Z}_d) \cup \Delta(S/\mathbb{Z}_d, R_i) \cup \Delta(R_i, R_i).$$

Since the cardinality of each R_i is at most N, it is easily seen that the sequence $(R_i)_{i\geq 0}$ reaches a fixed point which can be computed in polynomial time.

▶ **Lemma 7.** *The fixed point $R \stackrel{def}{=} \bigcup_{i\geq 0} R_i$ equals R_N and is computable in polynomial time.*

In particular, R together with S yields the whole residue reachability set.

▶ **Lemma 8.** *The set $X \stackrel{def}{=} R \cup S[n_0, n_0+N]/\mathbb{Z}_d$ is computable in polynomial time. Moreover,*

$$X = \{(q, n \bmod d) \mid q \in Q, n \in \mathit{reach}(q), n \geq n_0\}.$$

Proof (sketch). Polynomial-time computability of X follows immediately from Lemmas 6 and 7. The proof of the stated equality is quite technical though not too difficult and can be found in the technical report [7]. The crucial part for the inclusion "$\subseteq$" is to show that for every $i \in [0, N]$ and each $(q, r) \in R_i$ there exists some $n \in \mathit{reach}(q)$ with $n \geq n_0 + N - i$ and $n \equiv r \bmod d$ by induction on i. For the converse inclusion the only interesting case is when a potential reachability tree T is not $(n_0 + N)$-bounded. One first shows that all $\prec$-maximal nodes u in T with $T(u) \notin S$ satisfy $T(u)/\mathbb{Z}_d \in R_0$ and uses the fact that $\Delta(R, R) \subseteq R$ and $\Delta(R) \subseteq R$ to conclude $T(\varepsilon)/\mathbb{Z}_d \in R$. ◀

The main result of this section now follows directly from Lemma 8.

▶ **Theorem 9.** *Residue reachability for $BVASS_1$ is decidable in polynomial time.*

4.2 Expandable Partial Reachability Trees

We now employ our result on residue reachability to show that small partial reachability trees suffice in order to witness reachability. The key idea is to identify branches of partial reachability trees that end in a leaf and which could, informally speaking, be copied or pumped an arbitrary number of times, thus achieving a counter value in the leaf that is large enough and lies in a certain residue class of some modulus. Residue reachability then witnesses that such a leaf could be completed in order to yield a reachability tree. For the remainder of this section, fix some BVASS_1 $\mathcal{B} = (Q, \Delta, F)$.

Let us first introduce a couple of auxiliary definitions. Given a partial reachability tree $T\colon U \to Q \times \mathbb{N}$ and $v, w \in U$, the *lowest common ancestor* of $v, w \in U$ is defined as

$$\mathit{lca}(v, w) \stackrel{\text{def}}{=} \max\{u \in U \mid u \preceq v \text{ and } u \preceq w\},$$

where the maximum is taken with respect to $\preceq$. Let $T(u) = q(n)$, we define functions $\mathit{state}(u) \stackrel{\text{def}}{=} q$ and $\mathit{counter}(u) \stackrel{\text{def}}{=} n$ that allow us to access the control state and the counter value at u, respectively.

▶ **Definition 10.** A node $v \in U$ is *increasing* if there is a proper ancestor $u \prec v$ such that $\mathit{state}(u) = \mathit{state}(v)$ and $\mathit{counter}(u) < \mathit{counter}(v)$; the maximal such u is called the *anchor* of v. We say that T is *exclusive* if the least common ancestor of any two distinct increasing leaves is a proper ancestor of at least one of their anchors. Finally, we call T *expandable* if

- T is exclusive,
- every leaf v of T is either accepting or an increasing leaf,
- every increasing leaf v with anchor u such that $T(v) = q(n)$ and $T(u) = q(m)$ induces a valid instance of the residue reachability problem, i.e., $q(l)$ is reachable for some $l \geq n$ and $l \equiv n \bmod (n - m)$.

A node u is said to be *exclusive* resp. *expandable* if $T^{\downarrow u}$ is.

Observe that nodes cannot be both accepting and increasing because increasing nodes have strictly positive counter values and accepting nodes must have counter value zero. Exclusive and non-exclusive partial reachability trees are illustrated in Figure 2(a).

The next lemma states a useful fact that directly follows from the pigeon-hole principle: whenever the counter increases on a branch by a certain amount then the branch contains an increasing node and its anchor.

▶ **Lemma 11.** *Let u and v be nodes of a partial reachability tree such that $u \prec v$ and $\mathit{counter}(u) + |Q| \leq \mathit{counter}(v)$. Then there exists an increasing node v' with anchor u' such that $u \preceq u' \prec v' \preceq v$.*

The following lemma shows that every reachability tree gives rise to an expandable reachability tree whose nodes have counter values bounded polynomially in $|\mathcal{B}|$.

▶ **Lemma 12.** *Suppose $q(n)$ is reachable and let $B \stackrel{def}{=} 2 \cdot |Q| + n$. Then there exists an expandable B-bounded partial reachability tree with root $q(n)$.*

Proof. Let T be a reachability tree with $T(\varepsilon) = q(n)$. We call a node w of T *large* if $\mathit{counter}(w) = B$. We obtain a partial reachability tree T' from T as follows. By Lemma 11, every large node w gives rise to at least one pair of nodes (u, v) such that $u \prec v \preceq w$ and v is an increasing node with anchor u. For every large node w that is minimal with respect to $\preceq$, we assign the maximal such pair $\mathit{pair}(w) \stackrel{\text{def}}{=} (u, v)$ with respect to the lexicographical ordering on nodes (more precisely, $(u, v) \preceq (u', v')$ if either, $u \prec u'$, or $u = u'$ and $v \preceq v'$).

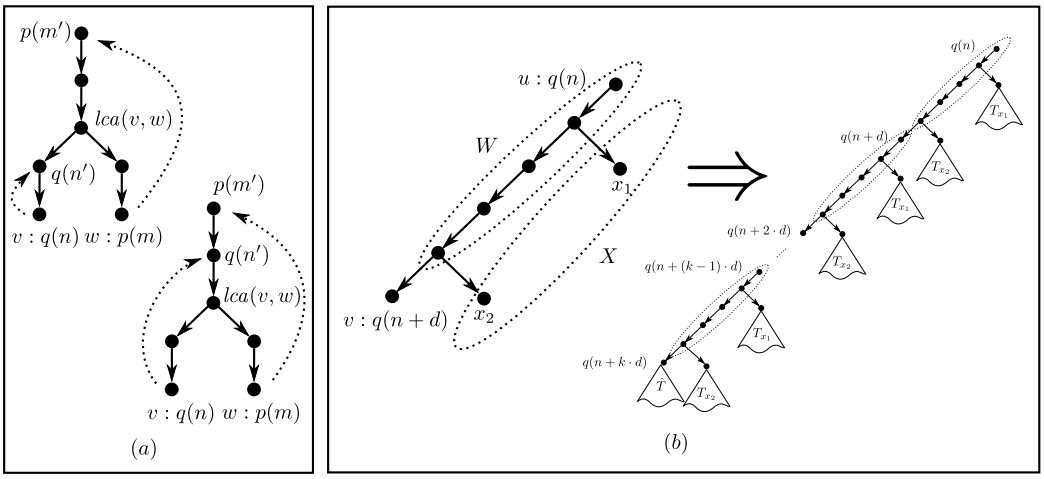


Figure 2 (a) Illustration an exclusive (top) and a non-exclusive (bottom) partial reachability tree. Here, v and w are pumping nodes and anchor relationships are depicted as dashed arrows. (b) Illustration of the pumping argument in Lemma 14.

Let $T' \colon U' \to Q \times \mathbb{N}$ denote the tree that one obtains from T by replacing all subtrees of T that are rooted at some node v such that $pair(w) = (u, v)$ for some minimal (with respect to $\preceq$) large node w in T by $\{v\}$ itself, i.e. such nodes v become leaves. We now prove that T' is B-bounded and exclusive:

- T' is B-bounded since the w above are chosen minimal with respect to $\preceq$ and hence $counter(u) \leq B$ for all nodes $u \in U'$.
- T' is exclusive, which can be seen as follows. Striving for a contradiction, suppose that T' is not exclusive. Then there are distinct increasing nodes v, v' with anchors u, u' such that $u, u' \preceq w \stackrel{\text{def}}{=} lca(v, v')$. Since $counter(w) = counter(w0) + counter(w1) \leq B$, we have $counter(w0) \leq B/2$ or $counter(w1) \leq B/2$, and assume without loss of generality that $counter(w0) \leq B/2$. Since $B - B/2 \geq |Q|$, by Lemma 11 there is another increasing node v'' with anchor u'' such that $w0 \preceq u'' \prec v''$, contradicting the assumed maximality of (u, v).
- Every leaf is accepting or increasing, by definition of T'.
- Finally, every increasing leaf u in T' induces a positive residue-reachability instance. Since T is a reachability tree, we have that $T(u)$ is reachable and thus $T'(u)$ is reachable. So in particular, it is reachable modulo $d = 1$, i.e. if $T'(u) = q(n)$, then we can choose $(q(n), 1)$ as the required valid instance of residue reachability. ◀

We now turn towards the converse direction and show that every expandable tree witnesses reachability. We first state an auxiliary lemma about structural properties of nodes in exclusive trees whose proof can be found in the technical report [7].

▶ **Lemma 13.** *For every node u of an expandable partial reachability tree the following hold:*

(i) *If u is the anchor of an increasing leaf v then u is expandable and all nodes w such that $u \prec w \preceq v$ are not expandable.*

(ii) *u has at most one child that is not expandable.*

The previous lemma enables us to show that an expandable partial reachability tree implies the existence of a reachability tree.

▶ **Lemma 14.** *Let $T\colon U \to Q \times \mathbb{N}$ be an expandable partial reachability tree. Then for all $u \in U$, $T(u)$ is reachable or u is not expandable.*

Proof. We prove the lemma by induction on $h(u)$. For the induction base, assume $h(u) = 0$, hence u is a leaf. Then u is either accepting and thus $T(u)$ is reachable, or u is not accepting and therefore an increasing leaf and so $T^{\downarrow u}$ is not expandable by Lemma 13(i).

For the induction step, suppose u is expandable. We distinguish two cases:

- *All children of u are expandable.* We only treat the case when u has two children, the case when u has one child follows as a special case. Since the children $u0$ and $u1$ of u are expandable, by the induction hypothesis there are reachability trees $T_0\colon U_0 \to Q \times \mathbb{N}$ and $T_1\colon U_1 \to Q \times \mathbb{N}$ with $T_0(\varepsilon) = T(u0)$ and $T_1(\varepsilon) = T(u1)$. We define the following tree $T_u\colon V \to Q \times \mathbb{N}$, where $V \stackrel{\text{def}}{=} \{0\}U_0 \cup \{1\}U_1 \cup \{\varepsilon\}$, $T_u(\varepsilon) \stackrel{\text{def}}{=} T(u)$ and $T_u(iv) \stackrel{\text{def}}{=} T_i(v)$ for all $i \in \{0,1\}$. Now T_u is a reachability tree, hence $T_u(\varepsilon) = T(u)$ is reachable.
- *Some child of u is not expandable.* For simplicity of presentation, let $u = \varepsilon$, the cases when $u \neq \varepsilon$ can be proven analogously. Moreover, let us assume that $T(u) = q(n)$. By Lemma 13(ii) there is at most one such child, without loss of generality let $u0 = 0$ be this child. Moreover, since u is expandable and $u0$ is not expandable it must hold that u is the anchor of some unique increasing leaf v, we may assume without loss of generality $v = u0^\ell$ for some $\ell \geq 1$. We must have $T(v) = q(n+d)$ for some $d \geq 1$. Let $W = \{0^i \mid i \in [0, \ell-1]\}$ be the set all nodes in T "on the path from u to v" without v. Let $X \stackrel{\text{def}}{=} \{0^i 1 \in U \mid i \in [0, \ell-1]\}$ be the set of all right children of nodes in W.
  By Lemma 13(i), all nodes in $\{0^i \mid i \in [1, \ell]\}$ are not expandable and consequently, Lemma 13(ii) implies that all nodes in X are expandable. Hence by induction hypothesis, for every $x \in X$ there is a reachability tree $T_x : U_x \to Q \times \mathbb{N}$ such that $T_x(\varepsilon) = T(x)$.
  It remains to show that $T(u) = q(n)$ is reachable. Since T is expandable there exists some $m \geq n+d$ such that $q(m)$ is reachable and $m \equiv n \bmod d$. Let us assume $m = n+d+k\cdot d$ for some $k \geq 0$ and let $\widehat{T}\colon Z \to Q \times \mathbb{N}$ be a reachability tree for $q(m)$.
  We construct the following reachability tree T' (formal definition below) for $q(n)$ as the tree one obtains from T by replacing the leaf v by the tree T repeatedly exactly k times and by adding to the counter values of the resulting nodes from 0^* in the i-th copy the counter value $i \cdot d$. This procedure is illustrated in Figure 2(b). Note that this process yields a partial reachability tree in which every leaf is accepting except for the leaf $0^{(k+1)\cdot\ell}$; therefore we replace this leaf by the tree $\widehat{T}\colon Z \to Q \times \mathbb{N}$. Recall that $T_x\colon U_x \to Q \times \mathbb{N}$ is a reachability tree for $T(x) = T_x(\varepsilon)$. Formally, we define $T'\colon \left(0^{(k+1)\cdot\ell}Z \cup \bigcup_{i=0}^{k} 0^{i\cdot\ell}(W \cup \bigcup\{xU_x \mid x \in X\})\right) \to Q \times \mathbb{N}$, where
  - $T'(0^{(k+1)\cdot\ell}z) \stackrel{\text{def}}{=} \widehat{T}(z)$ for all $z \in Z$,

  and for all $i \in [0,k]$ we put
  - $T'(0^{i\cdot\ell}w) \stackrel{\text{def}}{=} i\cdot d + T(w)$ for all $w \in W$, and
  - $T'(0^{i\cdot\ell}xy) \stackrel{\text{def}}{=} T_x(y)$ for all $x \in X$ and all $y \in U_x$.

  It easily checked that the result is a reachability tree for $T'(\varepsilon) = q(n)$. ◀

A consequence of the previous lemma is that in particular $T(\varepsilon)$ is reachable for every expandable partial reachability tree T. By combining Lemmas 12 and 14, we obtain the following characterisation of reachability in BVASS_1.

▶ **Proposition 15.** *A node $q(n)$ is reachable if, and only if, there exists an expandable B-bounded partial reachability tree T with $T(\varepsilon) = q(n)$, where $B \stackrel{def}{=} 2\cdot|Q| + n$.*

Algorithm 1 An alternating logspace procedure for reachability in BVASS_1.

```
1: procedure Reach(q(n))
2:     if n ∉ [0, B] then return false
3:     if q(n) ∈ F × {0} then return true
4:     else non-deterministically guess t ∈ Δ ∩ ({q} × Q × Q ∪ {q} × {−1, 0, 1} × Q)
5:         if t = (q, p1, p2) ∈ Q^3 then
6:             non-deterministically guess m1, m2 ∈ [0, B] s.t. n = m1 + m2
7:             return (Reach(p1(m1)) and Reach(p2(m2)))
8:                 or (AnchorReach(q(n), p1(m1)) and Reach(p2(m2)))
9:                 or (AnchorReach(q(n), p2(m2)) and Reach(p1(m1)))
10:        else let  t = (q, z, p) ∈ Q × {−1, 0, 1} × Q
11:            return Reach(p(n + z)) or AnchorReach(q(n), p(n + z))

12: procedure AnchorReach(q(n), p(m))
13:    if {n, m} ⊈ [0, B] then return false
14:    if p = q and m > n and ResidueReach(q(n), m − n) then return true
15:    else non-deterministically guess t ∈ Δ ∩ ({p} × Q × Q ∪ {p} × {−1, 0, 1} × Q)
16:        if t = (p, p1, p2) ∈ Q^3 then
17:            non-deterministically guess m1, m2 ∈ [0, B] s.t. m = m1 + m2
18:            return AnchorReach(q(n), p1(m1)) and Reach(p2(m2))
19:                or AnchorReach(q(n), p2(m2)) and Reach(p1(m1))
20:        else let  t = (p, z, p′) ∈ Q × {−1, 0, 1} × Q
21:            return AnchorReach(q(n), p′(m + z))
```

4.3 The Algorithm

In this section, we provide an alternating logspace procedure for reachability in BVASS_1. This shows that reachability in BVASS_1 is decidable in deterministic polynomial time since alternating logspace equals deterministic polynomial time [1]. We employ the characterisation of reachability in BVASS_1 in terms of expandable B-bounded partial reachability of Proposition 15. First, by Theorem 9 we may assume the existence of an alternating logspace procedure for residue reachability in BVASS_1, i.e., an alternating logspace procedure $\textsc{ResidueReach}(q(n_0), d)$ that has an accepting computation if, and only if, $q(n)$ is reachable for some $n \geq n_0$ and $n \equiv n_0 \bmod d$. By application of this procedure, we show that one can construct an alternating logspace procedure $\textsc{Reach}(q(n))$ that takes a configuration $q(n)$ as input and that has an accepting computation if, and only if, there exists an expandable B-bounded partial reachability tree T with $T(\varepsilon) = q(n)$.

The idea is to simply to guess an expandable B-bounded partial reachability tree T in a top-down manner. The procedure Reach is defined above in Algorithm 1. First in Line 2, Reach rejects whenever the counter value n is not in $[0, B]$ and accepts if $q(n)$ is an accepting configuration (Line 3). Thus, subsequently we may assume that $n \in [0, B]$. In Line 4, we non-deterministically choose a transition $t \in \Delta$. If $t = (q, p_1, p_2) \in Q^3$ is a branching rule, we non-deterministically guess how n can be decomposed as $n = m_1 + m_2$. Moreover, we non-deterministically guess whether the currently processed inner node of T labelled by $q(n)$ will be an anchor of some pumping leaf "below." If not then we simply recursively call $\textsc{Reach}(p_1(m_1))$ and $\textsc{Reach}(p_2(m_2))$ (Line 7). Otherwise, $q(n)$ will be the anchor of some pumping leaf that is either in the subtree "rooted at" $p_1(m_1)$ (Line 8) or in the subtree "rooted at" $p_2(m_2)$ (Line 9). Speaking in terms of Lemma 13, either the inner node corresponding to configuration $p_1(m_1)$ is not exclusive or the one for $p_2(m_2)$ is not exclusive. Suppose $p_1(m_1)$ is not exclusive, we then call a procedure $\textsc{AnchorReach}(q(n), p(m_1))$ that takes two configurations as arguments and tacitly assumes the first argument $q(n)$ is the anchor and the second argument $p_1(m_1)$ corresponds to some inner node that lies between the anchor and the pumping leaf it will eventually correspond to.

In more detail, analogously to Reach the procedure AnchorReach first checks whether the counter values of its inputs both lie in $[0, B]$ (Line 13). If so it checks whether $p(m)$ corresponds to a valid pumping leaf of $q(n)$, i.e., it induces a positive instance of the residue reachability problem by invoking ResidueReach$(q(n), m - n)$ (Line 14). If not then a rule $t \in \Delta$ is non-deterministically chosen (Line 15), and in case t is a branching rule, it is non-deterministically chosen which "child" of $p(m)$ is not exclusive, the other child is simply checked for reachability by invoking procedure Reach (Lines 18 and 19).

Obviously, Reach and AnchorReach can be implemented in alternating logspace since the involved counter values lie in the interval $[-1, B + 1]$ and can hence be stored using a logarithmic number of bits.

5 Coverability and Boundedness

In this section, we show that the coverability and boundedness problem for BVASS_1 are also P-complete. The two problems are defined as follows:

Coverability and Boundedness in BVASS_1

INPUT: A BVASS_1 $\mathcal{B} = (Q, \Delta, F)$, a control state q and $n \in \mathbb{N}$ encoded in unary.
QUESTION: *Coverability:* Is there $m \geq n$ such that $q(m)$ is reachable?
Boundedness: Is $\mathit{reach}(q)$ finite?

If $q(n)$ is a positive instance of coverability then we call the configuration $q(n)$ *coverable*. A state q is *unbounded* whenever $\mathit{reach}(q)$ is unbounded (i.e. infinite).

Hardness for P is in both cases easily seen and similar to the P-hardness reduction from MCVP in Proposition 3.

Moreover, the P upper bound for coverability follows easily from the P upper bound for residue reachability since $q(n)$ is coverable if, and only if, the pair $(q(n), 1)$ is a positive instance of the residue reachability problem.

▶ **Theorem 16.** *Coverability in BVASS_1 is P-complete.*

The P upper bound for boundedness, however, cannot be derived immediately. In particular, as discussed in Section 3, there exists a family of BVASS_1 $(\mathcal{B}_n)_{n \geq 0}$ with some control state q such that $\mathit{reach}(q)$ is finite but of cardinality 2^n.

For the remainder of this section, fix some BVASS_1 $\mathcal{B} = (Q, \Delta, F)$. We first provide sufficient and necessary criteria that witness that a control state is unbounded. Call a node v in a reachability tree *decreasing* if there is an ancestor $u \prec v$ with $\mathit{state}(u) = \mathit{state}(v)$ and $\mathit{counter}(u) > \mathit{counter}(v)$. The following lemma, whose proof can be found in the technical report [7], shows that a reachability tree that contains some decreasing node witnesses that the control state at its root is unbounded.

▶ **Lemma 17.** *If a reachability tree T with $T(\varepsilon) = q(n)$ contains a decreasing node then q is unbounded.*

Conversely, the next lemma shows that a reachability tree whose root is labelled with a configuration with a sufficiently large counter value gives rise to a reachability tree which contains a decreasing node, informally speaking, shortly below its root.

▶ **Lemma 18.** *Suppose $n > 2^{|Q|}$ with $n \in \mathit{reach}(q)$. There exists a reachability tree $T \colon U \to Q \times \mathbb{N}$ for $q(n')$ where $n' \geq n$, and which contains a decreasing node v with $|v| \leq |Q|$.*

A consequence of the two previous lemmas is that q is unbounded if, and only if, $\mathit{reach}(q)$ contains some $n > 2^{|Q|}$. Even though the reachability trees in Lemma 18 are sufficient witnesses for unboundedness, they still contain much more information than necessary and are potentially of exponential size. In order to verify the existence of such a tree, exact counter values and in fact the subtrees rooted in v as well as all incomparable nodes can be abstracted away, as shown in the lemma below.

Let us write $\mathit{src}(t) \stackrel{\text{def}}{=} q$, $\mathit{trg}(t) \stackrel{\text{def}}{=} \{p, p'\}$ and $\mathit{eff}(t) \stackrel{\text{def}}{=} 0$, for the *source* and *target states* and the *effect* of a branching transition $t = (q, p, p') \in Q^3$, respectively. Similarly, for $t = (q, z, p)$ define $\mathit{src}(t) \stackrel{\text{def}}{=} q$, $\mathit{trg}(t) \stackrel{\text{def}}{=} \{p\}$ and $\mathit{eff}(t) \stackrel{\text{def}}{=} z$.

▶ **Lemma 19.** *A control state p_0 is unbounded if, and only if, there is a sequence of control states and transitions $p_0 t_1 p_1 t_2 \cdots t_k p_k$ with $k \leq |Q|$ and some index $j < k$ such that*

- **(i)** *$p_{i-1} = \mathit{src}(t_i)$ and $p_i \in \mathit{trg}(t_i)$ for all $1 \leq i \leq k$;*
- **(ii)** *$p_k = p_j$ and $p_i \neq p_j$ for all $0 \leq i < j$;*
- **(iii)** *$p(0)$ is coverable for every $p \in \bigcup_{i=1}^{k} \mathit{trg}(t_i)$; and*
- **(iv)** *for every $j < i \leq k$, there exists $n_i \leq |Q| + 1$ such that*
  1. *if $t_i = (p_{i-1}, p_i, p'_i) \in Q^3$ or $t_i = (p_{i-1}, p'_i, p_i) \in Q^3$ then $p'_i(n_i)$ is coverable, else $n_i = 0$,*
  2. *$\sum_{i=j+1}^{k} n_i > \sum_{i=j+1}^{k} \mathit{eff}(t_i)$.*

The last condition (iv) expresses that the cyclic suffix is consistent with the transition relation and guarantees a decreasing node.

Proof. If p_0 is unbounded, then by Lemma 18 we can take a reachability tree T containing a short decreasing node v, i.e., with $|v| \leq |Q|$. This decreasing node provides the claimed sequence: Conditions (i) and (ii) are immediate; for condition (iii) notice that for each mentioned state p some configuration $p(n)$ is reachable, as guaranteed by the respective subtree of T. This means in particular that $p(0)$ is coverable.

For (iv), first notice that the combined effect $\sum_{i=j+1}^{k} \mathit{eff}(t_i)$ of those transitions used between v (where $\mathit{state}(v) = p_k$) and its anchor (with state $p_j = p_k$) is bounded by $|v| = k \leq |Q|$. Secondly, as for condition (iii), we can assume that for all p'_i such that either $t_i = (p_{i-1}, p'_i, p_i) \in Q^3$ or $t_i = (p_{i-1}, p_i, p'_i) \in Q^3$, some configuration $p'_i(m_i)$ is reachable. For those $i \leq k$ where $t_i \notin Q^3$, let $m_i \stackrel{\text{def}}{=} 0$. Now, for all $j < i \leq k$, define $n_i \stackrel{\text{def}}{=} \min\{|Q| + 1, m_i\}$.

Case (iv)(a) holds immediately by definition of the n_i. To show Case (iv)(b) we distinguish two cases. In case $m_i \geq |Q| + 1$ for some such i it follows that $n_i = |Q| + 1$ and hence $\sum_{i=j+1}^{k} n_i \geq |Q| + 1 > \sum_{i=j+1}^{k} \mathit{eff}(t_i)$. Otherwise, if all $m_i < |Q| + 1$ then for all i it holds that $n_i = m_i$ and so $\sum_{i=j+1}^{k} m_i \leq \sum_{i=j+1}^{k} \mathit{eff}(t_i)$ contradicts that v is a decreasing node.

For the converse direction, assume a sequence as claimed above. Conditions (i)-(iii) imply the existence of a reachability tree for some $p_0(n)$. Condition (iv) ensures that there is such a tree with a decreasing node. We conclude by Lemma 17. ◀

Lemma 19 provides a characterisation of unbounded states that directly translates into an alternating logspace algorithm for the boundedness problem, similar to Algorithm 1, which yields the P upper bound. In particular, observe that a witnessing sequence satisfying Conditions (i) and (ii), as well as the numbers $n_i \leq |Q|+1$ can be guessed non-deterministically in logarithmic space. Moreover, Conditions (iii) and (iv) are decidable in polynomial time by Theorem 16.

▶ **Theorem 20.** *Boundedness in $BVASS_1$ is* P*-complete.*

References

1 A.K. Chandra, D. Kozen, and L.J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981. doi:10.1145/322234.322243.

2 Ph. de Groote, B. Guillaume, and S. Salvati. Vector addition tree automata. In *Logic in Computer Science, LICS*, pages 64–73. IEEE Computer Society, 2004. doi:10.1109/LICS.2004.1319601.

3 S. Demri, M. Jurdziński, O. Lachish, and R. Lazić. The covering and boundedness problems for branching vector addition systems. *J. Comput. Syst. Sci.*, 79(1):23–38, 2013. doi:10.1016/j.jcss.2012.04.002.

4 M. Englert, R. Lazić, and P. Totzke. Reachability in two-dimensional unary vector addition systems with states is NL-complete. In *Logic in Computer Science, LICS*, 2016. To appear.

5 P. Ganty and R. Majumdar. Algorithmic verification of asynchronous programs. *ACM Trans. Program. Lang. Syst.*, 34(1):6, 2012. doi:10.1145/2160910.2160915.

6 S.M. German and A.P. Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992. doi:10.1145/146637.146681.

7 S. Göller, C. Haase, R. Lazić, and P. Totzke. A polynomial-time algorithm for reachability in branching VASS in dimension one. *CoRR*, abs/1602.05547, 2016. URL: http://arxiv.org/abs/1602.05547.

8 P. Jančar and Z. Sawa. A note on emptiness for alternating finite automata with a one-letter alphabet. *Inf. Process. Lett.*, 104(5):164–167, 2007. doi:10.1016/j.ipl.2007.06.006.

9 A. Jez and A. Okhotin. Conjunctive grammars over a unary alphabet: Undecidability and unbounded growth. *Theory Comput. Syst.*, 46(1):27–58, 2010. doi:10.1007/s00224-008-9139-5.

10 R. Lazić and S. Schmitz. Nonelementary complexities for branching VASS, MELL, and Extensions. *ACM Trans. Comput. Log.*, 16(3):20, 2015. doi:10.1145/2733375.

11 J. Leroux and S. Schmitz. Demystifying reachability in vector addition systems. In *Logic in Computer Science, LICS*, pages 56–67. IEEE, 2015. doi:10.1109/LICS.2015.16.

12 R.J. Lipton. The reachability problem requires exponential space. *Yale University*, Technical Report 62, 1976.

13 C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

14 C. Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comput. Sci.*, 6:223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

15 O. Serre. Parity games played on transition graphs of one-counter processes. In *Foundations of Software Science and Computation Structures, FoSSaCS*, pages 337–351, 2006. doi:10.1007/11690634_23.

16 L.G. Valiant and M. Paterson. Deterministic one-counter automata. *J. Comput. Syst. Sci.*, 10(3):340–350, 1975. doi:10.1016/S0022-0000(75)80005-5.

Reachability in Networks of Register Protocols under Stochastic Schedulers*†

Patricia Bouyer[1], Nicolas Markey[2], Mickael Randour‡[3], Arnaud Sangnier[4], and Daniel Stan[5]

1 **LSV, CNRS & ENS de Cachan, Cachan Cedex, France, and University Paris-Saclay, Paris, France**
2 **LSV, CNRS & ENS de Cachan, Cachan Cedex, France, and University Paris-Saclay, Paris, France**
3 **Computer Science Department, Université Libre de Bruxelles, Brussels, Belgium**
4 **IRIF, University Paris Diderot & CNRS, Paris, France**
5 **LSV, CNRS & ENS de Cachan, Cachan Cedex, France, and University Paris-Saclay, Paris, France**

Abstract

We study the almost-sure reachability problem in a distributed system obtained as the asynchronous composition of N copies (called processes) of the same automaton (called protocol), that can communicate via a shared register with finite domain. The automaton has two types of transitions: write-transitions update the value of the register, while read-transitions move to a new state depending on the content of the register. Non-determinism is resolved by a stochastic scheduler. Given a protocol, we focus on almost-sure reachability of a target state by one of the processes. The answer to this problem naturally depends on the number N of processes. However, we prove that our setting has a cut-off property: the answer to the almost-sure reachability problem is constant when N is large enough; we then develop an EXPSPACE algorithm deciding whether this constant answer is positive or negative.

1998 ACM Subject Classification F.1.1 Models of Computation, F.3.1 Specifying and Verifying and Reasoning about Programs, C.2.2 Network Protocols

Keywords and phrases Networks of Processes, Parametrized Systems, Stochastic Scheduler, Almost-sure Reachability, Cut-Off Property

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.106

1 Introduction

Verification of systems with many identical processes. It is a classical pattern in distributed systems to have a large number of identical components running concurrently (a.k.a. networks of processes). In order to verify the correctness of such systems, a naive option consists in fixing an upper bound on the number of processes, and applying classical verification techniques on the resulting system. This has several drawbacks, and in particular it gives no information whatsoever about larger systems. Another option is to

* A full version of the paper is available on Arxiv as [7].

† This work has been partly supported by ERC Starting grant EQualIS (FP7-308087), by European FET project Cassting (FP7-601148), and by the ANR research program PACS (ANR-14-CE28-0002).

‡ F.R.S.-FNRS Postdoctoral Researcher.

use parameterized-verification techniques, taking as a parameter the number of copies of the protocol in the system being considered. In such a setting, the natural question is to find and characterize the set of parameter values for which the system is correct. Not only the latter approach is more general, but it might also turn out to be easier and more efficient, since it involves orthogonal techniques.

Different means of communication lead to different models. A seminal paper on parameterized verification of such distributed systems is the work of German and Sistla [17]. In this work, the authors consider networks of processes all following the same finite-state automaton; the communication between processes is performed thanks to *rendez-vous* communication. Various related settings have been proposed and studied since then, which mainly differ by the way the processes communicate. Among those, let us mention broadcast communication [15, 10], token-passing [8, 2], message passing [6], shared register with ring topologies [1], or shared memory [16]. In his nice survey on such parameterized models [14], Esparza shows that minor changes in the setting, such as the presence of a controller in the system, might drastically change the complexity of the verification problems. The relative expressiveness of some of those models has been studied recently in [3], yielding several reductions of the verification problems for some of those classes of models.

Asynchronous shared-memory systems. We consider a communication model where the processes asynchronously access a shared register, and where read and write operations on this register are performed non-atomically. A similar model has been proposed by Hague in [18], where the behavior of processes is defined by a pushdown automaton. The complexity of some reachability and liveness problems for shared-memory models have then been established in [16] and [11], respectively. These works consider networks in which a specific process, called the leader, runs a different program, and address the problem whether, for some number of processes, the leader can satisfy a given reachability or liveness property. In the case where there is no leader, and where processes are finite-state, the parameterized control-state reachability problem (asking whether one of the processes can reach a given control state) can be solved in polynomial time, by adapting the approach of [9] for lossy broadcast protocols.

Fairness and cut-off properties. In this work, we further insert fairness assumptions in the model of parameterized networks with asynchronous shared memory, and consider reachability problems in this setting. There are different ways to include fairness in parameterized models. One approach is to enforce fairness expressed as a temporal-logic properties on the executions (e.g., any action that is available infinitely often must be performed infinitely often); this is the option chosen for parameterized networks with rendez-vous [17] and for systems with disjunctive guards (where processes can query the states of other processes) in [4]. We follow another choice, by equipping our networks with a stochastic scheduler that, at each step of the execution, assigns the same probability to the available actions of all the processes. From a high-level perspective, both forms of fairness are similar. However, expressing fairness via temporal logic allows for very regular patterns (e.g., round-robin execution of the processes), whereas the stochastic approach leads to consider all possible interleavings with probability 1. Under this stochastic scheduler assumption, we focus on almost-sure reachability of a given control state by any of the processes of the system. More specifically, as in [4], we are interested in determining the existence of a *cut-off*, i.e., an integer k such that networks with more than k processes almost-surely reach the target state. Deciding the existence and computing such cut-offs is important for at least two aspects: first, it ensures that the

system is correct for arbitrarily large networks; second, if we are able to derive a bound on the cut-off, then using classical verification techniques we can find the exact value of the cut-off and exactly characterize the sizes of the networks for which the behavior is correct.

Our contributions. We prove that for finite-state asynchronous shared-memory protocols with a stochastic scheduler, and for almost-sure reachability of some control state by some process of the network, there always exists a positive or negative cut-off; positive cut-offs are those above which the target state is reached with probability 1, while negative cut-offs are those above which the target state is reached with probability strictly less than 1. Notice that both cut-offs are not complement of one another, so that our result is not trivial.

We then prove that the "sign" (positive or negative) of a cut-off can be decided in EXPSPACE, and that this problem is PSPACE-hard. Finally, we provide lower and upper bounds on the values of the cut-offs, exhibiting in particular protocols with exponential (negative) cut-off. Notice how these results contrast with classical results in related areas: in the absence of fairness, reachability can be decided in polynomial time, and in most settings, when cut-offs exist, they generally have polynomial size [4, 13, 12].

2 Presentation of the model and of the considered problem

2.1 Preliminaries

Let S be a finite set. A multiset over S is a mapping $\mu\colon S \to \mathbb{N}$. The cardinality of a multiset μ is $|\mu| = \sum_{s\in S} \mu(s)$. The support $\overline{\mu}$ of μ is the subset $\nu \subseteq S$ s.t. for all $s \in S$, it holds $s \in \nu$ if, and only if, $\mu(s) > 0$. For $k \in \mathbb{N}$, we write $\mathbb{N}_k^S$ for the set of multisets of cardinality k over S, and $\mathbb{N}^S$ for the set of all multisets over S. For any $s \in S$ and $k \in \mathbb{N}$, we write s^k for the multiset where $s^k(s) = k$ and $s^k(s') = 0$ for all $s' \neq s$. We may write s instead of s^1 when no ambiguity may arise. A multiset μ is included in a multiset μ', written $\mu \sqsubseteq \mu'$, if $\mu(s) \leq \mu'(s)$ for all $s \in S$. Given two multisets μ and μ', their union $\mu \oplus \mu'$ is still a multiset s.t. $(\mu \oplus \mu')(s) = \mu(s) + \mu'(s)$ for all $s \in S$. Assuming $\mu \sqsubseteq \mu'$, the difference $\mu' \ominus \mu$ is still a multiset s.t. $(\mu' \ominus \mu)(s) = \mu'(s) - \mu(s)$.

A quasi-order $\langle A, \preceq\rangle$ is a *well quasi-order* (wqo for short) if for every infinite sequence of elements $a_1, a_2, \ldots$ in A, there exist two indices $i < j$ such that $a_i \preceq a_j$. For instance, for $n > 0$, $\langle \mathbb{N}^n, \leq\rangle$ (with lexicographic order) is a wqo. Given a set A with an ordering $\preceq$ and a subset $B \subseteq A$, the set B is said to be *upward closed* in A if for all $a_1 \in B$ and $a_2 \in A$, in case $a_1 \preceq a_2$, then $a_2 \in B$. The *upward-closure* of a set B (for the ordering $\preceq$), denoted by $\uparrow_{\preceq}(B)$ (or sometimes $\uparrow(B)$ when the ordering is clear from the context), is the set $\{a \in A \mid \exists b \in B \text{ s.t. } b \preceq a\}$. If $\langle A, \preceq\rangle$ is a wqo and B is an upward closed set in A, there exists a finite set of minimal elements $\{b_1, \ldots, b_k\}$ such that $B = \uparrow\{b_1, \ldots, b_k\}$.

2.2 Register protocols and associated distributed system

We focus on systems that are defined as the (asynchronous) product of several copies of the same protocol. Each copy communicates with the others through a single register that can store values from a finite alphabet.

▶ **Definition 1.** A *register protocol* is given by $\mathcal{P} = \langle Q, D, q_0, T\rangle$, where Q is a finite set of control locations, D is a finite alphabet of data for the shared register, $q_0 \in Q$ is an initial location, $T \subseteq Q \times \{R, W\} \times D \times Q$ is the set of transitions of the protocol. Here R means *read* the content of the shared register, while W means *write* in the register.

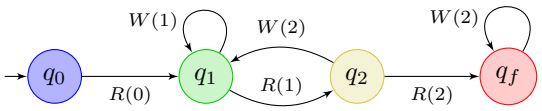


Figure 1 Example of a register protocol with $D = \{0, 1, 2\}$.

In order to avoid deadlocks, it is required that each location has at least one outgoing transition. We also require that whenever some R-transition (q, R, d', q') appears in T, then for all $d \in D$, there exists at least one $q_d \in Q$ such that $(q, R, d, q_d) \in T$. The size of the protocol $\mathcal{P}$ is given by $|Q| + |T|$.

▶ **Example 1.a.** *Figure 1 displays a small register protocol with four locations, over an alphabet of data* $D = \{0, 1, 2\}$. *In this figure (and in the sequel), omitted* R*-transitions (e.g., transitions* $R(1)$ *and* $R(2)$ *from* q_0*) are assumed to be self-loops. When the register contains* 0*, this protocol may move from initial location* q_0 *to location* q_1*. From there it can write* 1 *in the register, and then move to* q_2*. From* q_2*, as long as the register contains* 1*, the process can either stay in* q_2 *(with the omitted self-loop* $R(1)$*), or write* 2 *in the register and jump back to* q_1*. It is easily seen that if this process executes alone, it cannot reach state* q_f*.*

We now present the semantics of distributed systems associated with our register protocols. We consider the *asynchronous* composition of several copies of the protocol (the number of copies is not fixed a priori and can be seen as a parameter). We are interested in the behavior of such a composition under a fair scheduler. Such distributed systems involve two sources of non-determinism: first, register protocols may be non-deterministic; second, in any configuration, all protocols have at least one available transition, and non-determinism arises from the asynchronous semantics. In the semantics associated with a register protocol, non-determinism will be solved by a randomized scheduler, whose role is to select at each step which process will perform a transition, and which transition it will perform among the available ones. Because we will consider qualitative objectives (almost-sure reachability), the exact probability distributions will not really matter, and we will pick the uniform one (arbitrary choice). Note that we assume non-atomic read/write operations on the register, as in [18, 16, 11]. More precisely, when one process performs a transition, then all the processes that are in the same state are allowed to also perform the same transition just after, in fact write are always possible, and if a process performs a read of a specific value, since this read does not alter the value of the register, all processes in the same state can perform the same read (until one process performs a write). We will see later that dropping this hypothesis has a consequence on our results. We now give the formal definition of such a system.

The configurations of the distributed system built on register protocol $\mathcal{P} = \langle Q, D, q_0, T\rangle$ belong to the set $\Gamma = \mathbb{N}^Q \times D$. The first component of a configuration is a multiset characterizing the number of processes in each state of Q, whereas the second component provides the content of the register. For a configuration $\gamma = \langle \mu, d\rangle$, we denote by $st(\gamma)$ the multiset μ in $\mathbb{N}^Q$ and by $data(\gamma)$ the data d in D. We overload the operators defined over multisets; in particular, for a multiset δ over Q, we write $\gamma \oplus \delta$ for the configuration $\langle \mu \oplus \delta, d\rangle$. Similarly, we write $\overline{\gamma}$ for the support of $st(\gamma)$.

A configuration $\gamma' = \langle \mu', d'\rangle$ is a *successor* of a configuration $\gamma = \langle \mu, d\rangle$ if, and only if, there is a transition $(q, \mathsf{op}, d'', q') \in T$ such that $\mu(q) > 0$, $\mu' = \mu \ominus q \oplus q'$ and either $\mathsf{op} = R$ and $d = d' = d''$, or $\mathsf{op} = W$ and $d' = d''$. In that case, we write $\gamma \to \gamma'$. Note that since $\mu(q) > 0$ and $\mu' = \mu \ominus q \oplus q'$, we have necessarily $|\mu| = |\mu'|$. In our system, we assume that there is no creation or deletion of processes during an execution, hence the size of

configurations (i.e., $|st(\gamma)|$) remains constant along transitions. We write Γ_k for the set of configurations of size k. For any configuration $\gamma \in \Gamma_k$, we denote by $\mathrm{Post}(\gamma) \subseteq \Gamma_k$ the set of successors of γ, and point out that such a set is finite and non-empty.

Now, the *distributed system* $\mathcal{S}_\mathcal{P}$ associated with a register protocol $\mathcal{P}$ is a discrete-time Markov chain $\langle \Gamma, Pr \rangle$ where $Pr \colon \Gamma \times \Gamma \to [0,1]$ is the transition probability matrix defined as follows: for all γ and $\gamma' \in \Gamma$, we have $Pr(\gamma, \gamma') = \frac{1}{|\mathrm{Post}(\gamma)|}$ if $\gamma \to \gamma'$, and $Pr(\gamma, \gamma') = 0$ otherwise. Note that Pr is well defined: by the restriction imposed on the transition relation T of the protocol, we have $0 < |\mathrm{Post}(\gamma)| < \infty$ for all configuration γ, and hence we also get $\Sigma_{\gamma' \in \Gamma} Pr(\gamma, \gamma') = 1$. For a fixed integer k, we define the distributed system of size k associated with $\mathcal{P}$ as the finite-state discrete-time Markov chain $\mathcal{S}^k_\mathcal{P} = \langle \Gamma_k, \Pr_k \rangle$, where $\Pr_k$ is the restriction of Pr to $\Gamma_k \times \Gamma_k$.

We are interested in analyzing the behavior of the distributed system for a large number of participants. More precisely, we are interested in determining whether almost-sure reachability of a specific control state holds when the number of processes involved is large. We are therefore seeking a *cut-off* property, which we formalize in the following.

A finite path in the system $\mathcal{S}_\mathcal{P}$ is a finite sequence of configurations $\gamma_0 \to \gamma_1 \ldots \to \gamma_k$. In such a case, we say that the path starts in γ_0 and ends in γ_k. We furthermore write $\gamma \to^* \gamma'$ if, and only if, there exists a path that starts in γ and ends in γ'. Given a location q_f, we denote by $[\![\Diamond q_f]\!]$ the set of paths of the form $\gamma_0 \to \gamma_1 \ldots \to \gamma_k$ for which there is $i \in [0; k]$ such that $st(\gamma_i)(q_f) > 0$. Given a configuration γ, we denote by $\mathbb{P}(\gamma, [\![\Diamond q_f]\!])$ the probability that some paths starting in γ belong to $[\![\Diamond q_f]\!]$ in $\mathcal{S}_\mathcal{P}$. This probability is well-defined since the set of such paths is measurable (see e.g., [5]). Given a register protocol $\mathcal{P} = \langle Q, D, q_0, T \rangle$, an initial register value d_0, and a target location $q_f \in Q$, we say that q_f is almost-surely reachable for k processes if $\mathbb{P}(\langle q_0^k, d_0 \rangle, [\![\Diamond q_f]\!]) = 1$.

▶ **Example 1.b.** *Consider again the protocol depicted in Fig. 1, with initial register content* 0*. As we explained already, for $k = 1$, the final state is not reachable at all, for any scheduler (here as $k = 1$, the scheduler only has to solve non-determinism in the protocol).*

When $k = 2$, one easily sees that the final state is reachable: it suffices that both processes go to q_2 together, from where one process may write value 2 *in the register, which the other process can read and go to q_f. Notice that this does not ensure that q_f is reachable almost-surely for this k (and actually, it is not; see Example 1.c).*

We aim here at finding cut-offs for almost-sure reachability, i.e., we seek the existence of a threshold such that almost-sure reachability (or its negation) holds for all larger values.

▶ **Definition 2.** Fix a protocol $\mathcal{P} = \langle Q, D, q_0, T \rangle$, $d_0 \in D$, and $q_f \in Q$. An integer $k \in \mathbb{N}$ is a *cut-off for almost-sure reachability* (shortly a *cut-off*) for $\mathcal{P}$, d_0 and q_f if one of the following two properties holds:

- for all $h \geq k$, we have $\mathbb{P}(\langle q_0^h, d_0 \rangle, [\![\Diamond q_f]\!]) = 1$. In this case k is a *positive* cut-off;
- for all $h \geq k$, we have $\mathbb{P}(\langle q_0^h, d_0 \rangle, [\![\Diamond q_f]\!]) < 1$. Then k is a *negative* cut-off.

An integer k is a *tight* cut-off if it is a cut-off and $k - 1$ is not.

Notice that from the definition, cut-offs need not exist for a given distributed system. Our main result precisely states that cut-offs always exist, and that we can decide their nature.

▶ **Theorem 3.** *For any protocol $\mathcal{P}$, any initial register value d_0 and any target location q_f, there always exists a cut-off for almost-sure reachability, whose value is at most doubly-exponential in the size of $\mathcal{P}$. Whether it is a positive or a negative cut-off can be decided in* EXPSPACE*, and is* PSPACE*-hard.*

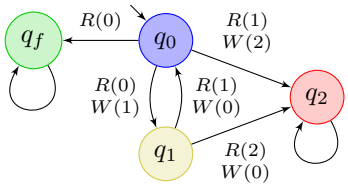


Figure 2 Example of a register protocol with atomic read/write operations.

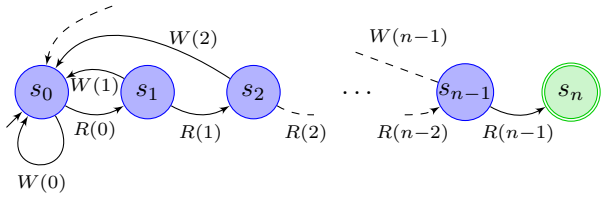


Figure 3 A "filter" protocol $\mathcal{F}_n$ for $n > 0$.

▶ Remark. When dropping the condition on non-atomic read/write operations, and allowing transitions with atomic read/write operations (i.e., one process is ensured to perform a read and a write operation without to be interrupted by another process), the existence of a cut-off (Theorem 3) is not ensured. This is demonstrated with the protocol of Fig. 2: one easily checks (e.g., inductively on the number of processes, since processes that end up in q_2 play no role anymore) that state q_f is reached with probability 1 if, and only if, the number of processes is odd.

3 Properties of register protocols

3.1 Example of a register protocol

We illustrate our model with a family of register protocols $(\mathcal{F}_n)_{n>0}$, depicted in Fig. 3. For a fixed n, protocol $\mathcal{F}_n$ has $n+1$ states and n different data; intuitively, in order to move from s_i to s_{i+1}, two processes are needed: one writes i in the register and goes back to s_0, and the second process can proceed to s_{i+1} by reading i. Since backward transitions to s_0 are always possible and since states can always exit s_0 by writing a 0 and reading it afterwards, no deadlock can ever occur so the main question remains to determine if s_n is reachable by one of the processes as we increase the number of initial processes. As shown in Lemma 4, the answer is positive: $\mathcal{F}_n$ has a tight linear positive cut-off; it actually behaves like a "filter", that can test if at least n processes are running together. We exploit this property later in Section 4.4.

▶ **Lemma 4.** *Fix $n \in \mathbb{N}$. The "filter" protocol $\mathcal{F}_n$, depicted in Fig. 3, with initial register value 0 and target location s_n, has a tight positive cut-off equal to n.*

3.2 Basic results

In this section, we consider a register protocol $\mathcal{P} = \langle Q, D, q_0, T\rangle$, its associated distributed system $\mathcal{S}_\mathcal{P} = \langle \Gamma, Pr\rangle$, an initial register value $d_0 \in D$ and a target state $q_f \in Q$. We define a partial order $\preceq$ over the set Γ of configurations as follows: $\langle \mu, d\rangle \preceq \langle \mu', d'\rangle$ if, and only if, $d = d'$ and $\overline{\mu} = \overline{\mu'}$ and $\mu \sqsubseteq \mu'$. Note that with respect to the classical order over multisets, we require here that the supports of μ and μ' be the same (we add in fact a finite information to hold for the comparison). We know from Dickson's lemma that $\langle \mathbb{N}^Q, \sqsubseteq\rangle$ is a wqo and since Q, D and the supports of multisets in $\mathbb{N}^Q$ are finite, we can deduce the following lemma.

▶ **Lemma 5.** *$\langle \Gamma, \preceq\rangle$ is a wqo.*

We will give some properties of register protocols, but first we introduce some further notations. Given a set of configuration $\Delta \subseteq \Gamma$, we define $\mathrm{Pre}^*(\Delta)$ and $\mathrm{Post}^*(\Delta)$ as follows:

$$\mathrm{Pre}^*(\Delta) = \{\gamma \in \Gamma \mid \exists \gamma' \in \Delta . \gamma \to^* \gamma'\} \qquad \mathrm{Post}^*(\Delta) = \{\gamma' \in \Gamma \mid \exists \gamma \in \Delta . \gamma \to^* \gamma'\}$$

We also define the set $[\![q_f]\!]$ of configurations we aim to reach as $\{\gamma \in \Gamma \mid st(\gamma)(q_f) > 0\}$. It holds that $\gamma \in \mathrm{Pre}^*([\![q_f]\!])$ if, and only if, there exists a path in $[\![\Diamond q_f]\!]$ starting in γ.

As already mentioned, when $\langle \mu, d\rangle \to \langle \mu', d'\rangle$ in $\mathcal{S}_{\mathcal{P}}$, the multisets μ and μ' have the same cardinality. This implies that given $k > 0$, the set $\mathrm{Post}^*(\{\langle q_0^k, d_0\rangle\})$ is finite (remember that Q and D are finite). As a consequence, for a fixed k, checking whether $\mathbb{P}(\langle q_0^k, d_0\rangle, [\![\Diamond q_f]\!]) = 1$ can be easily achieved by analyzing the finite-state discrete-time Markov chain $\mathcal{S}_{\mathcal{P}}^k$ [5].

▶ **Lemma 6.** *Let $k \geq 1$. Then $\mathbb{P}(\langle q_0^k, d_0\rangle, [\![\Diamond q_f]\!]) = 1$ if, and only if, $\mathrm{Post}^*(\{\langle q_0^k, d_0\rangle\}) \subseteq \mathrm{Pre}^*([\![q_f]\!])$.*

The difficulty here precisely lies in finding such a k and in proving that, once we have found one correct value for k, all larger values are correct as well (to get the cut-off property). Characteristics of register protocols provide us with some tools to solve this problem. We base our analysis on reasoning on the set of configurations reachable from initial configurations in $\uparrow\{\langle q_0, d_0\rangle\}$ (the upward closure of $\{\langle q_0, d_0\rangle\}$ w.r.t. $\preceq$), remember that since the order $\langle \Gamma, \preceq\rangle$ requires equality of support for elements to be comparable, we have that $\uparrow\{\langle q_0, d_0\rangle\} = \bigcup_{k\geq 1}\{\langle q_0^k, d_0\rangle\}$. We begin by showing that this set of reachable configurations and the set of configurations from which $[\![q_f]\!]$ is reachable are both upward-closed. Thanks to Lemma 5, they can be represented as upward closures of finite sets. To show that $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ is upward-closed, we prove that register protocols enjoy the following monotonicity property. A similar property is given in [11] and derives from the non-atomicity of operations.

▶ **Lemma 7.** *Let γ_1, γ_2, and γ_2' be configurations in Γ. If $\gamma_1 \to^* \gamma_2$ and $\gamma_2 \preceq \gamma_2'$, then there exists $\gamma_1' \in \Gamma$ such that $\gamma_1' \to^* \gamma_2'$ and $\gamma_1 \preceq \gamma_1'$.*

$\mathrm{Pre}^*([\![q_f]\!])$ is also upward-closed, since if $[\![q_f]\!]$ can be reached from some configuration γ, it can also be reached by a larger configuration by keeping the extra copies idle. Thus:

▶ **Lemma 8.** $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ *and* $\mathrm{Pre}^*([\![q_f]\!])$ *are upward-closed sets in* $\langle \Gamma, \preceq\rangle$.

3.3 Existence of a cut-off

From Lemma 8, and from the fact that $\langle \Gamma, \preceq\rangle$ is a wqo, there must exist two finite sequences of configurations $(\theta_i)_{1\leq i\leq n}$ and $(\eta_i)_{1\leq i\leq m}$ such that $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\}) = \uparrow\{\theta_1, \ldots, \theta_n\}$ and $\mathrm{Pre}^*([\![q_f]\!]) = \uparrow\{\eta_1, \ldots, \eta_m\}$. By analyzing these two sequences, we now prove that any register protocol has a cut-off (for any initial register value and any target location).

We let $\Delta, \Delta' \subseteq \Gamma$ be two upward-closed sets (for $\preceq$). We say that Δ *is included in* Δ' *modulo single-state incrementation* whenever for every $\gamma \in \Delta$, for every $q \in \overline{\gamma}$, there is some $k \in \mathbb{N}$ such that $\gamma \oplus q^k \in \Delta'$. Note that this condition can be checked using only comparisons between minimal elements of Δ and Δ'. In particular, we have the following lemma.

▶ **Lemma 9.** $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ *is included in* $\mathrm{Pre}^*([\![q_f]\!])$ *modulo single-state incrementation if, and only if, for all $i \in [1; n]$, and for all $q \in \overline{\theta_i}$, there exists $j \in [1; m]$ such that $data(\theta_i) = data(\eta_j)$ and $\overline{\theta_i} = \overline{\eta_j}$ and $st(\eta_j)(q') \leq st(\theta_i)(q')$ for all $q' \in Q \setminus \{q\}$.*

Using the previous characterization of inclusion modulo single-state incrementation for $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ and $\mathrm{Pre}^*([\![q_f]\!])$ together with the result of Lemma 6, we are able to provide a first characterization of the existence of a negative cut-off.

▶ **Lemma 10.** *If* $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ *is not included in* $\mathrm{Pre}^*([\![q_f]\!])$ *modulo single-state incrementation, then* $\max_{1\leq i\leq n}(|st(\theta_i)|)$ *is a negative cut-off.*

We now prove that if the condition of Lemma 10 fails to hold, then there is a positive cut-off.In order to make our claim precise, for every $i \in [1; n]$ and for any $q \in \overline{\theta_i}$, we let $d_{i,q} = \max\{(|st(\eta_j)(q) - st(\theta_i)(q)|) \mid 1 \leq j \leq m \text{ and } \overline{\theta_i} = \overline{\eta_j}\}$.

▶ **Lemma 11.** *If* $\mathrm{Post}^*(\uparrow\{\langle q_0, d_0\rangle\})$ *is included in* $\mathrm{Pre}^*([\![q_f]\!])$ *modulo single-state incrementation, then* $\max_{1\leq i\leq n}(|st(\theta_i)| + \sum_{q\in\overline{\theta_i}} d_{i,q})$ *is a positive cut-off.*

The last two lemmas entail our first result:

▶ **Theorem 12.** *Any register protocol admits a cut-off (for any given initial register value and target state).*

4 Detecting negative cut-offs

We develop an algorithm for deciding whether a distributed system associated with a register protocol has a negative cut-off. Thanks to Theorem 12, this can also be used to detect the existence of a positive cut-off. Our algorithm relies on the construction and study of a *symbolic graph*, as we define below: for any given protocol $\mathcal{P}$, the symbolic graph has bounded size, but can be used to reason about *arbitrarily large* distributed systems built from $\mathcal{P}$. It will store sufficient information to decide the existence of a negative cut-off.

4.1 k-bounded symbolic graph

In this section, we consider a register protocol $\mathcal{P} = \langle Q, D, q_0, T\rangle$, its associated distributed system $\mathcal{S}_\mathcal{P} = \langle \Gamma, Pr\rangle$, an initial register value $d_0 \in D$, and a target location $q_f \in Q$ of $\mathcal{P}$. With $\mathcal{P}$, we associate a finite-state graph, called *symbolic graph of index* k, which for k large enough contains enough information to decide the existence of a negative cut-off.

▶ **Definition 13.** Let k be an integer. The *symbolic graph of index* k associated with $\mathcal{P}$ and d_0 is the transition system $\mathcal{G} = \langle V, v_0, E\rangle$ where

- $V = \mathbb{N}_k^Q \times 2^Q \times D$ contains triples made of a multiset of states of Q of size k, a subset of Q, and the content of the register; the multiset (called *concrete part*) is used to exactly keep track of a fixed set of k processes, while the subset of Q (the *abstract part*) encodes the support of the arbitrarily many remaining processes;
- $v_0 = \langle q_0^k, \{q_0\}, \{d_0\}\rangle$;
- transitions are of two types, depending whether they involve a process in the concrete part or a process in the abstract part. Formally, there is a transition $\langle \mu, S, d\rangle \to \langle \mu', S', d'\rangle$ whenever there is a transition $(q, O, d'', q') \in T$ such that $d = d' = d''$ if $O = R$ and $d' = d''$ if $O = W$, and one of the following two conditions holds:
  - either $S' = S$ and $q \sqsubseteq \mu$ (that is, $\mu(q) > 0$) and $\mu' = \mu \ominus q \oplus q'$;
  - or $\mu = \mu'$ and $q \in S$ and $S' \in \{S \setminus \{q\} \cup \{q'\}, S \cup \{q'\}\}$.

The symbolic graph of index k can be used as an abstraction of distributed systems made of at least $k + 1$ copies of $\mathcal{P}$: it keeps full information of the states of k processes, and only gives the support of the states of the other processes. In particular, the symbolic graph of index 0 provides only the states appearing in each configuration of the system.

▶ **Example 1.c.** *Consider the protocol depicted in Fig. 1. Its symbolic graph of index* 0 *is depicted in Fig. 4. Notice that the final state (representing all configurations containing* q_f*) is reachable from any state of this symbolic graph. However, our original protocol* $\mathcal{P}$ *of Fig. 1 does not have a positive cut-off (assuming initial register value* 0*): indeed, with*

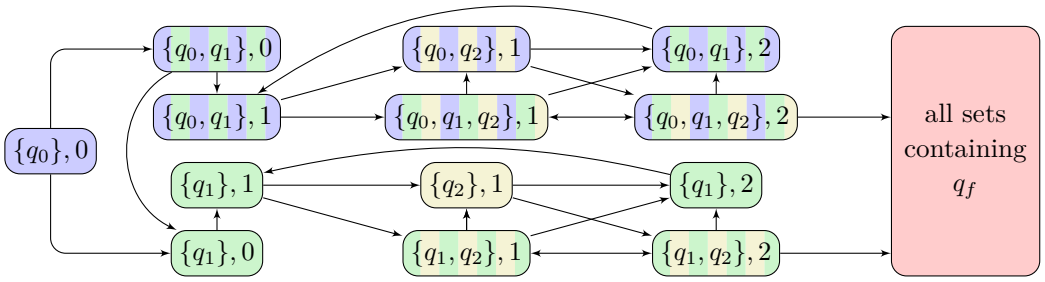


Figure 4 Symbolic graph (of index 0) of the protocol of Fig. 1 (self-loops omitted).

positive probability, a single process will go to q_1 and immediately write 1 in the register, thus preventing any other process to leave q_0; then one may check that the process in q_1 alone cannot reach q_f, so that the probability of reaching q_f from q_0^k is strictly less than 1, for any $k > 0$. This livelock is not taken into account in the symbolic graph of index 0, because from any configuration with support $\{q_0, q_1\}$ and register data equal to 1, the symbolic graph has a transition to the configuration with support $\{q_0, q_1, q_2\}$, which only exists in the concrete system when there are at least two processes in q_1. As we prove in the following, analyzing the symbolic graph for a sufficiently large index guarantees to detect such a situation.

For any index k, the symbolic graph achieves the following correspondence:

▶ **Lemma 14.** *Given two states $\langle \mu, S, d\rangle$ and $\langle \mu', S', d'\rangle$, there is a transition from $\langle \mu, S, d\rangle$ to $\langle \mu', S', d'\rangle$ in the symbolic graph $\mathcal{G}$ of index k if, and only if, there exist multisets δ and δ' with respective supports S and S', and such that $\langle \mu \oplus \delta, d\rangle \to \langle \mu' \oplus \delta', d'\rangle$ in $\mathcal{S}_\mathcal{P}$.*

4.2 Deciding the existence of a negative cut-off

We now explain how the symbolic graph can be used to decide the existence of a negative cut-off. Since $\mathrm{Pre}^*(\llbracket q_f \rrbracket)$ is upward-closed in $\langle \Gamma, \preceq\rangle$, there is a finite set of configurations $\{\eta_i = \langle \mu_i, d_i\rangle \mid 1 \le i \le m\}$ such that $\mathrm{Pre}^*(\llbracket q_f \rrbracket) = \uparrow\{\eta_i \mid 1 \le i \le m\}$. We let $K = \max\{st(\eta_i)(q) \mid q \in Q,\ 1 \le i \le m\}$, and show that for our purpose, it is enough to consider the symbolic graph of index $K \cdot |Q|$; we provide a bound on K in the next section.

▶ **Lemma 15.** *There is a negative cut-off for $\mathcal{P}$, d_0 and q_f if, and only if, there is a node in the symbolic graph of index $K \cdot |Q|$ that is reachable from $\langle q_0^{K\cdot|Q|}, \{q_0\}, d_0\rangle$ but from which no configuration involving q_f is reachable.*

Proof. We begin with the converse implication, assuming that there is a state $\langle \mu, S, d\rangle$ in the symbolic graph of index $K \cdot |Q|$ that is reachable from $(q_0^{K\cdot|Q|}, \{q_0\}, d_0)$ and from which no configuration in $\llbracket q_f \rrbracket$ is reachable. Applying Lemma 14, there exist multisets $\delta_0 = q_0^N$ and δ, with respective supports $\{q_0\}$ and S, such that $\langle \mu \oplus \delta, d\rangle$ is reachable from $\langle q_0^{K\cdot|Q|} \oplus \delta_0, d_0\rangle$. If location q_f was reachable from $\langle \mu \oplus \delta, d\rangle$ in the distributed system, then there would exist a path from $\langle \mu, S, d\rangle$ to a state involving q_f in the symbolic graph, which contradicts our hypothesis. By Lemma 7, it follows that such a configuration $\langle \mu \oplus \delta', d\rangle$ — which cannot reach q_f — can be reached from $\langle q_0^{K\cdot|Q|} \oplus q_0^{N'}, d_0\rangle$ for any $N' \ge N$: hence it cannot be the case that q_f is reachable almost-surely for any $N' \ge N$. Therefore there cannot be a positive cut-off, which implies that there is a negative one (from Theorem 12).

Conversely, if there is a negative cut-off, then for some $N > K \cdot |Q|$, the distributed system $\mathcal{S}_\mathcal{P}^N$ with N processes has probability less than 1 of reaching $\llbracket q_f \rrbracket$ from q_0^N. This system

being finite, there must exist a reachable configuration $\langle \mu, d \rangle$ from which q_f is not reachable [5]. Hence $\langle \mu, d \rangle \notin \mathrm{Pre}^*([\![q_f]\!])$, and for all $i \leq m$, there is a location q^i such that $\mu(q^i) < \mu_i(q^i) \leq K$. Then there must exist a reachable state $\langle \kappa, S, d \rangle$ of the symbolic graph of index $K \cdot |Q|$ for which $\kappa(q^i) = \mu(q^i)$ and $q^i \notin S$, for all $1 \leq i \leq m$: it indeed suffices to follow the path from $\langle q_0^N, d_0 \rangle$ to $\langle \mu, d \rangle$ while keeping track of the processes that end up in some q^i in the concrete part; this is possible because the concrete part has size at least $K \cdot |Q|$.

It remains to be proved that no state involving q_f is reachable from $\langle \kappa, S, d \rangle$ in the symbolic graph. If it were the case, then by Lemma 14, there would exist δ with support S such that $[\![q_f]\!]$ is reachable from $\langle \kappa \oplus \delta, d \rangle$ in the distributed system. Then $\langle \kappa \oplus \delta, d \rangle \in \mathrm{Pre}^*([\![q_f]\!])$, so that for some $1 \leq i \leq m$, $(\kappa \oplus \delta)(q^i) \geq \mu_i(q^i)$, which is not possible as $\kappa(q^i) < \mu_i(q^i)$ and q^i is not in the support S of δ. This contradiction concludes the proof. ◀

▶ Remark. Besides the existence of a negative cut-off, this proof also provides us with an upper bound on the tight cut-off, as we shall see in Section 5.

4.3 Complexity of the algorithm

We now consider the complexity of the algorithm that can be deduced from Lemma 15. Using results by Rackoff on the coverability problem in Vector Addition Systems [19], we can bound K – and consequently the size of the needed symbolic graph – by a *double-exponential* in the size of the protocol. Therefore, it suffices to solve a reachability problem in NLOGSPACE [20] on this doubly-exponential graph: this boils down to NEXPSPACE with regard to the protocol's size, hence EXPSPACE by Savitch's theorem [20].

▶ **Theorem 16.** *Deciding the existence of a negative cut-off is in* EXPSPACE.

4.4 PSPACE-hardness for deciding cut-offs

▶ **Theorem 17.** *Deciding the existence of a negative cut-off is* PSPACE*-hard.*

Our proof is based on the encoding of a linear-bounded Turing machine [20]: we build a register protocol for which there is a negative cut-off if, and only if, the machine reaches its final state q_{halt} with the tape head reading the last cell of the tape. Write n for the size of the tape of the Turing machine. We assume (without loss of generality) that the machine is deterministic, and that it accepts only if it ends in its halting state q_{halt} while reading the last cell of the tape. Our reduction works as follows: some processes of our network will first be assigned an index i in $[1; n]$ indicating the cell of the tape they shall encode during the simulation. The other processes are stuck in the initial location, and will play no role. The state q and position j of the head of the Turing machine are stored in the register. During the simulation phase, when a process is scheduled to play, it checks in the register whether the tape head is on the cell it encodes, and in that case it performs the transition of the Turing machine. If the tape head is not on the cell it encodes, the process moves to the target location (which we consider as the target for the almost-sure reachability problem). Finally, upon seeing (q_{halt}, n) in the register, all processes move to a $(n+1)$-filter protocol $\mathcal{F}_{n+1}$ (similar to that of Fig. 3) whose last location s_{n+1} is the aforementioned target location.

If the Turing machine halts, then the corresponding run can be mimicked with exactly one process per cell, thus giving rise to a finite run of the distributed system where n processes end up in the $(n+1)$-filter (and the other processes are stuck in the initial location); from there s_{n+1} cannot be reached. If the Turing machine does not halt, then assume that there is

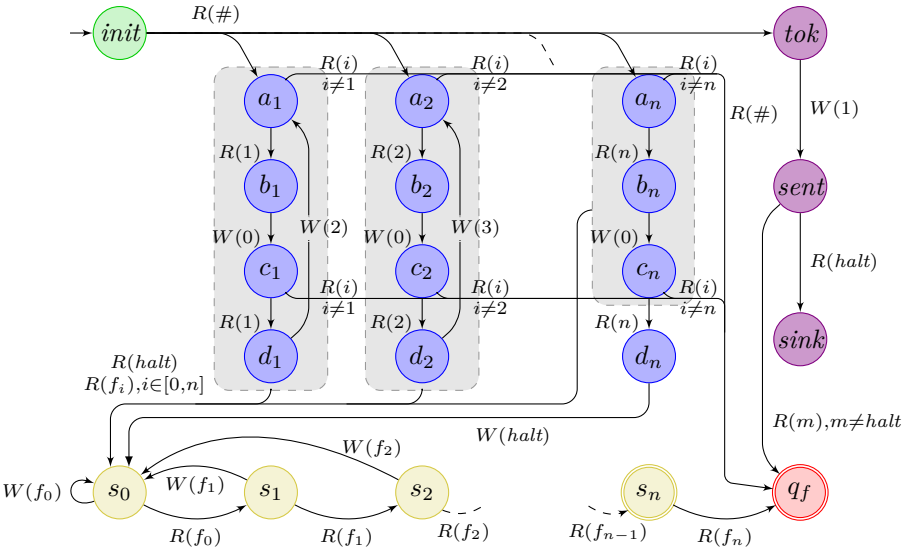


Figure 5 Simulating an exponential counter: grey boxes contain the nodes used to encode the bits of the counter; yellow nodes at the bottom correspond to the filter module from Fig. 3; purple nodes *tok*, *sent* and *sink* correspond to the second part of the protocol, and are used to produce tokens. Missing *read* edges are assumed to be self-loops.

an infinite run of the distributed system never reaching the target location. This run cannot get stuck in the simulation phase forever, because it would end up in a strongly connected component from which the target location is reachable. Thus this run eventually reaches the $(n+1)$-filter, which requires that at least $n+1$ processes participate in the simulation (because with n processes it would simulate the exact run of the machine, and would not reach q_{halt}, while with fewer processes the tape head could not go over cells that are not handled by a process). Thus at least $n+1$ processes would end up in the $(n+1)$-filter, and with probability 1 the target location should be reached.

5 Bounds on cut-offs

5.1 Existence of exponential tight negative cut-offs

We exhibit a family of register protocols that admits negative cut-off exponential in the size of the protocol. The construction reuses ideas from the PSPACE-hardness proof. Our register protocol has two parts: one part simulates a counter over n bits, and requires a *token* (a special value in the register) to perform each step of the simulation. The second part is used to generate the tokens (i.e., writing 1 in the register). Figure 5 depicts our construction. We claim that this protocol, with # as initial register value and q_f as target location, admits a negative tight cut-off larger than 2^n: in other terms, there exists $N > 2^n$ such that the final state will be reached with probability strictly less than 1 in the distributed system made of at least N processes (starting with # in the register), while the distributed system with 2^n processes will reach the final state almost-surely. In order to justify this claim, we explain now the intuition behind this protocol.

We first focus on the first part of the protocol, containing nodes named a_i, b_i, c_i, d_i and s_i. This part can be divided into three phases: the initialization phase lasts as long as the register contains #; the counting phase starts when the register contains *halt* for the first time; the simulation phase is the intermediate phase.

During the initialization phase, processes move to locations a_i and *tok*, until some process in *tok* writes 1 in the register (or until some process reaches q_f, using a transition from a_i to q_f while reading #). Write γ_0 for the configuration reached when entering the simulation phase (i.e., when 1 is written in the register for the first time). We assume that $st(\gamma_0)(a_i) > 0$ for some i, as otherwise all the processes are in *tok*, and they all will eventually reach q_f. Now, we notice that if $st(\gamma_0)(a_i) = 0$ for some i, then location d_n cannot be reached, so that no process can reach the counting phase. In that case, some process (and actually all of them) will eventually reach q_f. We now consider the case where $st(\gamma_0)(a_i) \geq 1$ for all i. One can prove (inductively) that d_i is reachable when $st(\gamma_0)(tok) \geq 2^i$. Hence d_n, and thus also s_0, can be reached when $st(\gamma_0)(tok) \geq 2^n$. Assuming q_f is not reached, the counting phase must never contain more than n processes, hence we actually have that $st(\gamma_0)(a_i) = 1$. With this new condition, s_0 is reached if, and only if, $st(\gamma_0)(tok) \geq 2^n$. When the latter condition is not true, q_f will be reached almost-surely, which proves the second part of our claim: the final location is reached almost-surely in systems with strictly less than $n + 2^n$ copies of the protocol.

We now consider the case of systems with at least $n + 2^n$ processes. We exhibit a finite execution of those systems from which no continuation can reach q_f, thus proving that q_f is reached with probability strictly less than 1 in those systems. The execution is as follows: during initialization, for each i, one process enters a_i; all other processes move to *tok*, and one of them write 1 in the register. The n processes in the simulation phase then simulate the consecutive incrementations of the counter, consuming one token at each step, until reaching d_n. At that time, all the processes in *tok* move to *sent*, and the process in d_n writes *halt* in the register and enters s_0. The processes in the simulation phase can then enter s_0, and those in *sent* can move to *sink*. We now have n processes in s_0, and the other ones in *sink*. According to Lemma 4, location q_f cannot be reached from this configuration, which concludes our proof.

▶ **Theorem 18.** *There exists a family of register protocols which, equipped with an initial register value and a target location, admit negative tight cut-offs whose size are exponential in the size of the protocol.*

▶ Remark. The question whether there exists protocols with exponential *positive* cut-offs remains open. The family of *filter* protocols described at Section 3.1 is an example of protocols with a linear positive cut-off.

5.2 Upper bounds on tight cut-offs

The results (and proofs) of Section 4 can be used to derive upper bounds on tight cut-offs. We make this explicit in the following theorem.

▶ **Theorem 19.** *For a protocol $\mathcal{P} = \langle Q, D, q_0, T\rangle$ equipped with an initial register value $d_0 \in D$ and a target location $q_f \in Q$, the tight cut-off is at most doubly-exponential in $|\mathcal{P}|$.*

6 Conclusions and future works

We have shown that in networks of identical finite-state automata communicating (non-atomically) through a single register and equipped with a fair stochastic scheduler, there always exists a cut-off on the number of processes which either witnesses almost-sure reachability of a specific control-state (positive cut-off) or its negation (negative cut-off). This cut-off determinacy essentially relies on the monotonicity induced by our model, which

allows to use well-quasi order techniques. By analyzing a well-chosen symbolic graph, one can decide in EXPSPACE whether that cut-off is positive, or negative, and we proved this decision problem to be PSPACE-hard. This approach allows us to deduce some doubly-exponential bounds on the value of the cut-offs. Finally, we gave an example of a network in which there is a negative cut-off, which is exponential in the size of the underlying protocol. Note however that no such lower-bound is known yet for positive cut-offs.

We have several further directions of research. First, it would be nice to fill the gap between the PSPACE lower bound and the EXPSPACE upper bound for deciding the nature of the cut-off. We would like also to investigate further atomic read/write operations, which generate non-monotonic transition systems, but for which we would like to decide whether there is a cut-off or not. Finally, we believe that our techniques could be extended to more general classes of properties, for instance, universal reachability (all processes should enter a distinguished state), or liveness properties.

References

1 C. Aiswarya, Benedikt Bollig, and Paul Gastin. An automata-theoretic approach to the verification of distributed algorithms. In CONCUR'15, LIPIcs 42, pp. 340–353. LZI, 2015. DOI: 10.4230/LIPIcs.CONCUR.2015.340

2 Benjamin Aminof, Swen Jacobs, Ayrat Khalimov, and Sasha Rubin. Parametrized model checking of token-passing systems. In VMCAI'14, LNCS 8318, pp. 262–281. Springer, 2014. DOI: 10.1007/978-3-642-54013-4_15

3 Benjamin Aminof, Sasha Rubin, and Florian Zuleger. On the expressive power of communication primitives in parameterised systems. In LPAR'15, LNCS 9450, p. 313–328. Springer, 2015. DOI: 10.1007/978-3-662-48899-7_22

4 Simon Außerlechner, Swen Jacobs, and Ayrat Khalimov. Tight cutoffs for guarded protocols with fairness. In VMCAI'16, LNCS 9583, pp. 476–494. Springer, 2016. DOI: 10.1007/978-3-662-49122-5_23

5 Christel Baier and Joost-Pieter Katoen. *Principles of Model-Checking*. MIT Press, 2008.

6 Benedikt Bollig, Paul Gastin, and Jana Schubert. Parameterized verification of communicating automata under context bounds. In RP'14, LNCS 8762, pp. 45–57. Springer, 2014. DOI: 10.1007/978-3-319-11439-2_4

7 Patricia Bouyer, Nicolas Markey, Mickael Randour, Arnaud Sangnier, and Daniel Stan. Reachability in networks of register protocols under stochastic schedulers. Technical Report abs/1602.05928, arXiv CoRR, 2016. URL: http://arxiv.org/abs/1602.05928

8 Edmund M. Clarke, Muralidhar Talupur, Tayssir Touili, and Helmut Veith. Verification by network decomposition. In CONCUR'04, LNCS 3170, pp. 276–291. Springer, 2004. DOI: 10.1007/978-3-540-28644-8_18

9 Giorgio Delzanno, Arnaud Sangnier, Riccardo Traverso, and Gianluigi Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In FSTTCS'12, LIPIcs 18, pp. 289–300. LZI, 2012. DOI: LIPIcs.FSTTCS.2012.289

10 Giorgio Delzanno, Arnaud Sangnier, and Gianluigi Zavattaro. Parameterized verification of ad hoc networks. In CONCUR'10, LNCS 6269, pp. 313–327. Springer, 2010. DOI: 10.1007/978-3-642-15375-4_22

11 Antoine Durand-Gasselin, Javier Esparza, Pierre Ganty, and Rupak Majumdar. Model checking parameterized asynchronous shared-memory systems. In CAV'15, LNCS 9206, pp. 67–84. Springer, 2015. DOI: 10.1007/978-3-319-21690-4_5

12 E. Allen Emerson and Vineet Kahlon. Reducing model checking of the many to the few. In CADE'00, LNAI 1831, pp. 236–254. Springer, 2000. DOI: 10.1007/10721959_19

13 E. Allen Emerson and Kedar Namjoshi. On reasoning about rings. *Int. J. Found. Comp. Sci.*, 14(4):527–550, 2003. DOI: 10.1142/S0129054103001881

14 Javier Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In STACS'14, LIPIcs 25, pp. 1–10. LZI, 2014. DOI: 10.4230/LIPIcs.STACS.2014.1

15 Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In LICS'99, pp. 352–359. IEEE Comp. Soc. Press, 1999. DOI: 10.1109/LICS.1999.782630

16 Javier Esparza, Pierre Ganty, and Rupak Majumdar. Parameterized verification of asynchronous shared-memory systems. In CAV'13, LNCS 8044, pp. 124–140. Springer, 2013. DOI: 10.1007/978-3-642-39799-8_8

17 Steven M. German and A. Prasad Sistla. Reasoning about systems with many processes. *J. of the ACM*, 39(3):675–735, 1992.

18 Matthew Hague. Parameterised pushdown systems with non-atomic writes. In FSTTCS'11, LIPIcs 13, pp. 457–468. LZI, 2011. DOI: 10.4230/LIPIcs.FSTTCS.2011.457

19 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theor. Comp. Sci.*, 6:223–231, 1978. DOI: 10.1016/0304-3975(78)90036-1

20 Michael Sipser. *Introduction to the theory of computation.* PWS Publishing Co., 1997.

A Program Logic for Union Bounds*

Gilles Barthe[1], Marco Gaboardi[2], Benjamin Grégoire[3], Justin Hsu[4], and Pierre-Yves Strub[5]

1 **IMDEA Software Institute, Madrid, Spain**
2 **University at Buffalo, SUNY, New, York, USA**
3 **Inria Sophia Antipolis - Méditerranée, Valbonne, France**
4 **University of Pennsylvania, Philadelphia, USA**
5 **IMDEA Software Institute, Madrid, Spain**

Abstract

We propose a probabilistic Hoare logic aHL based on the union bound, a tool from basic probability theory. While the union bound is simple, it is an extremely common tool for analyzing randomized algorithms. In formal verification terms, the union bound allows flexible and compositional reasoning over possible ways an algorithm may go wrong. It also enables a clean separation between reasoning about probabilities and reasoning about events, which are expressed as standard first-order formulas in our logic. Notably, assertions in our logic are non-probabilistic, even though we can conclude probabilistic facts from the judgments.

Our logic can also prove accuracy properties for interactive programs, where the program must produce intermediate outputs as soon as pieces of the input arrive, rather than accessing the entire input at once. This setting also enables adaptivity, where later inputs may depend on earlier intermediate outputs. We show how to prove accuracy for several examples from the differential privacy literature, both interactive and non-interactive.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases Probabilistic Algorithms, Accuracy, Formal Verification, Hoare Logic, Union Bound

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.107

1 Introduction

Probabilistic computations arise naturally in many areas of computer science. For instance, they are widely used in cryptography, privacy, and security for achieving goals that lie beyond the reach of deterministic programs. However, the correctness of probabilistic programs can be quite subtle, often relying on complex reasoning about probabilistic events.

Accordingly, probabilistic computations present an attractive target for formal verification. A long line of research, spanning more than four decades, has focused on expressive formalisms for reasoning about general probabilistic properties both for purely probabilistic programs and for programs that combine probabilistic and non-deterministic choice (see, e.g., [35, 29, 34]).

More recent research investigates specialized formalisms that work with more restricted assertions and proof techniques, aiming to simplify formal verification. As perhaps the purest examples of this approach, some program logics prove probabilistic properties by working purely with non-probabilistic assertions; we call such systems *lightweight* logics.

* This work was partially supported by NSF grants TWC-1513694, CNS-1065060 and CNS-1237235, by EPSRC grant EP/M022358/1 and by a grant from the Simons Foundation (#360368 to Justin Hsu).

Examples include *probabilistic relational Hoare logic* [7] for proving the reductionist security of cryptographic constructions, and the related *approximate probabilistic relational Hoare logic* [8] for reasoning about differential privacy. These logics rely on the powerful abstraction of *probabilistic couplings* to derive probabilistic facts from non-probabilistic assertions [4].

Lightweight logics are appealing because they can leverage ideas for verifying deterministic programs, a rich and well-studied area of formal verification. However, existing lightweight logics apply only to relational verification: properties about the relation between two programs. In this paper, we propose a non-relational, lightweight logic based on the *union bound*, a simple tool from probability theory. For arbitrary properties $E_1, \ldots, E_n$, the union bound states that

$$\Pr\left[\cup_{i=1}^{n} E_i\right] \leq \sum_{i=1}^{n} \Pr[E_i]\,.$$

Typically, we think of the events E_i as *bad events*, describing different ways that the program may fail to satisfy some target property. Bad events can be viewed as propositions on single program states, so they can be represented as non-probabilistic assertions. For example, the formula $x > 10$ defines a bad event for x a program variable. If x stores the result from a random sample, this bad event models when the sample is bigger than 10. The union bound states that no bad events happen, except with probability at most the sum of the probabilities of each bad event.

The union bound is a ubiquitous tool in pen-and-paper proofs due to its flexible and compositional nature: to bound the probability of a collection of failures, consider each failure in isolation. This compositional style is also a natural fit for formal verification. To demonstrate this, we formalize a Hoare logic aHL based on the union bound for a probabilistic imperative language. The assertions in our logic are non-probabilistic, but judgments carry a numeric index for tracking the failure probability. Concretely, the aHL judgment

$$\vdash_{\beta} c : \Phi \Longrightarrow \Psi$$

states that every execution of a program c starting from an initial state satisfying Φ yields a distribution in which Ψ holds except with probability at most β. We define a proof system for the logic and show its soundness. We also define a sound embedding of aHL into standard Hoare logic, by instrumenting the program with ghost code that tracks the index β in a special program variable. This is a useful reduction that also applies to other lightweight logics [5].

Moreover, our logic applies both to standard algorithms and to *interactive* algorithms, a richer class of algorithms that is commonly studied in contexts such as *online learning* (algorithms which make predictions about the future input) and *streaming* (algorithms which operate on datasets that are too large to fit into memory by processing the input in linear passes). Informally, interactive algorithms receive their input in a sequence of chunks, and must produce intermediate outputs as soon as each chunk arrives. In some cases the input can be *adaptive*: later inputs may depend on earlier outputs. Besides enabling new classes of algorithms, interactivity allows more modularity. We can decompose programs into interacting parts, analyze each part in isolation, and reuse the components.

We demonstrate aHL on several algorithms satisfying *differential privacy* [14], a statistical notion of privacy which trades off between the privacy of inputs and the accuracy of outputs. Prior work on verifying private algorithms focuses on the privacy property for non-interactive algorithms (see, e.g. [37, 18, 8]). We provide the first verification of *accuracy* for both non-interactive and interactive algorithms. We note however that aHL, like the union bound, can be applied to a wide range of probabilistic programs beyond differential privacy.

2 A union bound logic

Before introducing the program logic, we will begin by reviewing a largely standard, probabilistic imperative language. We state the soundness of the logic and describe the embedding into Hoare logic. The semantics of the language and the proof of soundness are deferred to the appendix.

2.1 Language

We will work with a core imperative language with a command for random sampling from distributions, and procedure calls. The set of commands is defined as follows:

$\mathcal{C} ::=$	skip	noop
	$\mid \mathcal{X} \leftarrow \mathcal{E}$	deterministic assignment
	$\mid \mathcal{X} \xleftarrow{\$} \mathcal{D}(\mathcal{E})$	probabilistic assignment
	$\mid \mathcal{C}; \mathcal{C}$	sequencing
	$\mid \mathsf{if}\ \mathcal{E}\ \mathsf{then}\ \mathcal{C}\ \mathsf{else}\ \mathcal{C}$	conditional
	$\mid \mathsf{while}\ \mathcal{E}\ \mathsf{do}\ \mathcal{C}$	while loop
	$\mid \mathcal{X} \leftarrow \mathcal{F}(\mathcal{E})$	procedure call
	$\mid \mathcal{X} \leftarrow \mathcal{A}(\mathcal{E})$	external call

Here, $\mathcal{X}$ is a set of *variables*, $\mathcal{E}$ is a set of *expressions*, and $\mathcal{D}$ is a set of *distribution constructors*, which can be parameterized by standard expressions. Variables and expressions are typed, ranging over booleans, integers, lists, etc. The expression grammar is entirely standard, and we omit it.

We distinguish two kinds of procedure calls: $\mathcal{A}$ is a set of external procedure names, and $\mathcal{F}$ is a set of internal procedure names. We assume we have access to the code of internal procedures, but not the code of external procedures. We think of external procedures as controlled by some external *adversary*, who can select the next input in an interactive algorithm. Accordingly, external procedures run in an *external memory* separate from the main program memory, which is shared by all internal procedures.

For simplicity, procedures take a single argument, do not have local variables, and are not mutually recursive. A program consists of a sequence of procedures definitions, each of the following form:

$$\mathbf{proc}\ f(\mathbf{arg}_f)\{c; \mathbf{return}\ r;\}\,.$$

Here, f is a procedure name, $\mathbf{arg}_f \in \mathbf{Vars}$ is the formal argument of f, c is the function body and r is its return value. We assume that distinct procedure definitions do not bind the same procedure name and that the program variable $\mathbf{arg}_f$ can only appear in the body of f.

Before we define the program semantics, we first need to introduce a few definitions from probability theory.

▶ **Definition 1.** A discrete sub-distribution over a set A is defined by a *mass function* $\mu : A \to [0, 1]$ such that:

- the support $\mathrm{supp}(\mu)$ of μ – defined as $\{x \in A \mid \mu(x) \neq 0\}$ – is countable; and
- the weight $\mathrm{wt}(\mu)$ of μ – defined as $\sum_{x \in A} \mu(x)$ – satisfies $\mathrm{wt}(\mu) \leq 1$.

A *distribution* is a sub-distribution with weight 1. The probability of an event P w.r.t. μ, written $\Pr_\mu[P]$ (or $\Pr[P]$ when μ is clear from the context), is defined as $\sum_{x \in A | P(x)} \mu(x)$. When Φ is an assertion (assuming that $A \equiv \mathsf{State}$), we write $\Pr_\mu[\Phi]$ for $\Pr_\mu[\lambda m.\, m \models \Phi]$. Likewise, when $v \in A$, we write $\Pr_\mu[v]$ for $\Pr_\mu[\lambda x.\, x = v]$.

Commands are interpreted as a function from memories to sub-distributions over memories, where memories are finite maps from program and external variables to values. More formally, if State is the set of memories then the interpretation of c, written $[\![c]\!]$, is a function from State to $\mathbf{Distr}(\mathsf{State})$, where $\mathbf{Distr}(\mathsf{T})$ denotes the set of discrete sub-distributions over T. The definition of $[\![c]\!]$ enforces the separation between the internal and external states – only commands performing external procedure calls can act on the external memory. The interpretation of external procedure calls is parameterized by functions – one for each external procedure – of type $\mathsf{State}_{|\mathcal{A}} \to \mathbf{Distr}(\mathsf{State}_{|\mathcal{A}})$, where $\mathsf{State}_{|\mathcal{A}}$ is the set of memories *restricted* to the external variables. Thus, external procedures can only access the external memory.

2.2 Logic

Now that we have seen the programs, let us turn to the program logic. Our judgments are similar to standard Hoare logic with an additional numeric index representing the probability of failure. Concretely, the judgments are of the following form:

$$\vdash_\beta c : \Phi \Longrightarrow \Psi$$

where Φ and Ψ are first-order formulas over the program variables representing the pre- and post-condition, respectively. We stress that Φ and Ψ are *non-probabilistic* assertions: they do not mention the probabilities of specific events, and will be interpreted as properties of individual memories rather than distributions over memories. This is reflected by the validity relation for assertions: $m \models \Phi$ states that Φ is valid in the *single* memory m, rather than in a distribution over memories. Similarly, $\models \Phi$ states that Φ is valid in all (single) memories. By separating the assertions from the probabilistic features of our language, the assertions are simpler and easier to manipulate. The index β is a non-negative real number (typically, from the unit interval $[0, 1]$).

Now, we can define semantic validity for our judgments. In short, the index β will be an upper bound on the probability that the postcondition Ψ does not hold on the output distribution, assuming the precondition Φ holds on the initial memory.

▶ **Definition 2** (Validity). A judgment $\vdash_\beta c : \Phi \Longrightarrow \Psi$ is *valid* if for every memory m such that $m \models \Phi$, we have:

$$\Pr_{[\![c]\!](m)}[\neg\Psi] \leq \beta\,.$$

We present the main proof rules of our logic in Figure 1. The rule for random sampling [Rand] allows us to assume a proposition Ψ about the random sample provided that Ψ fails with probability at most β. This is a semantic condition which we introduce as an axiom for each primitive distribution.

The remaining rules are similar to the standard Hoare logic rules, with special handling for the index. The sequence rule [Seq] states that the failure probabilities of the two commands add together; this is simply the union bound internalized in our logic. The conditional rule [If] assumes that the indices for the two branch judgments are equal – which can always be achieved via weakening – keeping the same index for the conditional. Roughly, this is because only one branch of the conditional is executed. The loop rule [While] simply accumulates the failure probability β throughout the iterations; the side conditions ensure that the loop terminates in at most k iterations except with probability $k \cdot \beta$. To reason about procedure calls, standard (internal) procedure calls use the rule [Call], which substitutes the argument and return variables in the pre- and post-condition, respectively. External procedure calls

$$\frac{}{\vdash_0 \mathsf{skip} : \Phi \Longrightarrow \Phi}\ [\textsc{Skip}] \qquad \frac{}{\vdash_0 x \leftarrow e : \Phi[e/x] \Longrightarrow \Phi}\ [\textsc{Assn}]$$

$$\frac{\forall m.\, m \models \Phi \implies \Pr_{[\![x \xleftarrow{\$} d(e)]\!](m)}[\neg\Psi] \leq \beta}{\vdash_\beta x \xleftarrow{\$} d(e) : \Phi \Longrightarrow \Psi}\ [\textsc{Rand}]$$

$$\frac{\vdash_\beta c : \Phi \Longrightarrow \Phi' \quad \vdash_{\beta'} c' : \Phi' \Longrightarrow \Phi''}{\vdash_{\beta+\beta'} c; c' : \Phi \Longrightarrow \Phi''}\ [\textsc{Seq}] \qquad \frac{\vdash_\beta c : \Phi \wedge e \Longrightarrow \Psi \quad \vdash_\beta c' : \Phi \wedge \neg e \Longrightarrow \Psi}{\vdash_\beta \mathsf{if}\ e\ \mathsf{then}\ c\ \mathsf{else}\ c' : \Phi \Longrightarrow \Psi}\ [\textsc{If}]$$

$$\frac{e_v : \mathbb{N} \qquad \models \Phi \wedge e_v \leq 0 \to \neg e \qquad \vdash_\beta c : \Phi \Longrightarrow \Phi \qquad \forall \eta > 0.\ \vdash_0 c : \Phi \wedge e \wedge e_v = \eta \Longrightarrow e_v < \eta}{\vdash_{k\cdot\beta} \mathsf{while}\ e\ \mathsf{do}\ c : \Phi \wedge e_v \leq k \Longrightarrow \Phi \wedge \neg e}\ [\textsc{While}]$$

$$\frac{\mathbf{proc}\ f(\mathbf{arg}_f)\{c; \mathbf{return}\ r;\} \quad \vdash_\beta c : \Phi \Longrightarrow \Psi[r/\mathbf{res}_f]}{\vdash_\beta x \leftarrow f(e) : \Phi[e/\mathbf{arg}_f] \Longrightarrow \Psi[x/\mathbf{res}_f]}\ [\textsc{Call}] \qquad \frac{}{\vdash_0 x \leftarrow f(e) : \forall v.\ \Psi[v/x] \Longrightarrow \Psi}\ [\textsc{Ext}]$$

$$\frac{\models \Phi' \to \Phi \qquad \models \Psi \to \Psi' \qquad \beta \leq \beta' \qquad \vdash_\beta c : \Phi \Longrightarrow \Psi}{\vdash_{\beta'} c : \Phi' \Longrightarrow \Psi'}\ [\textsc{Weak}] \qquad \frac{c \text{ does not modify variables in } \Phi}{\vdash_0 c : \Phi \Longrightarrow \Phi}\ [\textsc{Frame}]$$

$$\frac{\vdash_\beta c : \Phi \Longrightarrow \Psi \quad \vdash_{\beta'} c : \Phi \Longrightarrow \Psi'}{\vdash_{\beta+\beta'} c : \Phi \Longrightarrow \Psi \wedge \Psi'}\ [\textsc{And}] \qquad \frac{\vdash_\beta c : \Phi \Longrightarrow \Psi \quad \vdash_\beta c : \Phi' \Longrightarrow \Psi}{\vdash_\beta c : \Phi \vee \Phi' \Longrightarrow \Psi}\ [\textsc{Or}] \qquad \frac{}{\vdash_1 c : \Phi \Longrightarrow \bot}\ [\textsc{False}]$$

Figure 1 Selected proof rules.

use the rule [Ext]. We do not have access to the implementation of the procedure; we know just the type of the return value.

The structural rules are also similar to the typical Hoare logic rules. The weakening rule [Weak] allows strengthening the precondition and weakening the postcondition as usual, but also allows increasing the index – this corresponds to allowing a possibly higher probability of failure. The frame rule [Frame] preserves assertions that do not mention variables modified by the command. The conjunction rule [And] is another instance of the union bound, allowing us to combine two postconditions while adding up the failure probabilities. The case rule [Or] is the dual of [And] and takes the maximum failure probability among two post-conditions when taking their disjunction. Finally, the rule [False] allows us to conclude false with failure probability 1: With probability at most 0, false holds in the final memory.

We can show that our proof system is sound with respect to the semantics; the proof is deferred to the appendix.

▶ **Theorem 3** (Soundness). *All derivable judgments* $\vdash_\beta c : \Phi \Longrightarrow \Psi$ *are valid.*

In addition, we can define a sound embedding into Hoare logic in the style of Barthe et al. [5]. Assuming a fresh program variable x_β of type $\mathbb{R}$, we can transform a command c

such that $\vdash_\beta c : \Phi \Longrightarrow \Psi$ to a new command $\lceil c \rceil$ and a proof of the standard Hoare logic judgment

$$\vdash \lceil c \rceil : \Phi \wedge x_\beta = 0 \implies \Psi \wedge x_\beta \leq \beta \,.$$

The command $\lceil c \rceil$ is obtained from c by replacing all probabilistic sampling $x \xleftarrow{\$} d(e)$ with a call to an abstract, non-probabilistic procedure call $x \leftarrow \text{Sample}^\diamond\,(d(e))$, whose specification models the postcondition of [Rand]:

$$\frac{\forall m.\, m \models \Phi \implies \Pr_{[\![x \xleftarrow{\$} d(e)]\!](m)}[\neg\Psi] \leq \iota}{\vdash x \leftarrow \text{Sample}^\diamond\,(d(e)) : \Phi \wedge x_\beta \leq \nu \implies \Psi \wedge x_\beta \leq \nu + \iota} \,.$$

3 Accuracy for differentially private programs

Now that we have presented our logic aHL, we will follow by verifying several examples. Though our system applies to programs from many domains, we will focus on programs satisfying *differential privacy*, a statistical notion of privacy proposed by Dwork et al. [14]. At a very high level, these programs take private data as input and add random noise to protect privacy. (Interested readers should consult a textbook [15] for a more detailed presentation.) In contrast to existing formal verification work, which verifies the privacy property, we will verify *accuracy*. This is just as important as privacy: the constant function is perfectly private but not very useful.

All of our example programs take samples from the Laplace distribution.

▶ **Definition 4.** The *(discrete) Laplace* distribution $\mathcal{L}_\epsilon(e)$ is parameterized by a scale parameter $\epsilon > 0$ and a mean e. The distribution ranges over the real numbers $\{\nu = k + e\}$ for k an integer, releasing ν with probability proportional to:

$$\Pr_{\mathcal{L}_\epsilon(e)}[\nu] \propto \exp\left(-\epsilon \cdot |\nu - e|\right).$$

This distribution satisfies a basic accuracy property.

▶ **Lemma 5.** *Let $\beta \in (0,1)$, and let ν be a sample from the distribution $\mathcal{L}_\epsilon(e)$. Then,*

$$\Pr_{\mathcal{L}_\epsilon(e)}\left[|\nu - e| > \frac{1}{\epsilon}\log\frac{1}{\beta}\right] < \beta \,.$$

Thus, the following sampling rule is sound for our system for every $\beta \in (0,1)$:

$$\frac{}{\vdash_\beta x \xleftarrow{\$} \mathcal{L}_\epsilon(e) : \top \Longrightarrow |x - e| \leq \frac{1}{\epsilon}\log\frac{1}{\beta}} \text{ [LapAcc]} \,.$$

Before presenting the examples, we will set some common notations and terminology. First, we consider a set db of databases,[1] a set query of queries, and primitive functions

$$\begin{array}{rcl} \mathsf{evalQ} & : & \mathsf{query} \to \mathsf{db} \to \mathbb{R} \\ \mathsf{invQ} & : & \mathsf{query} \to \mathsf{query} \\ \mathsf{negQ} & : & \mathsf{query} \to \mathsf{query} \\ \mathsf{size} & : & \mathsf{db} \to \mathbb{N} \\ \mathsf{error} & : & \mathsf{query} \to \mathsf{db} \to \mathsf{query} \end{array}$$

[1] The general setting of differential privacy is that the database contains private information that must be protected. However, this fact will not be important for proving accuracy.

satisfying

$$\begin{aligned} \mathsf{evalQ}(\mathsf{invQ}(q), d) &= -\mathsf{evalQ}(q, d) \\ \mathsf{evalQ}(\mathsf{negQ}(q), d) &= \mathsf{size}(d) - \mathsf{evalQ}(q, d) \\ \mathsf{evalQ}(\mathsf{error}(q, d_1), d_2) &= \mathsf{evalQ}(q, d_1) - \mathsf{evalQ}(q, d_2) \end{aligned}$$

Concretely, one can identify query with the functions $\mathsf{db} \to \mathbb{R}$ and obtain an easy realization of the above functions and axioms.

In some situations, we may need additional structure on the queries to prove the accuracy guarantees. In particular, a query q is *linear* if

- for every two databases d, d', we have $q(d + d') = q(d) + q(d')$ for a commutative and associative operator $+$ on databases; and
- for the database d_0 that is the identity of $+$, we have $q(d_0) = 0$.

Concretely, we can identify db with the set of multisets, $+$ with multiset union, and d_0 with the empty multiset.

3.1 Report-noisy-max

Our first example is the *Report-noisy-max* algorithm (see, e.g., Dwork and Roth [15]). Report-noisy-max is a variant of the *exponential mechanism* [32], which provides the standard way to achieve differential privacy for computations whose outputs lie in a finite (perhaps non-numeric) set $\mathcal{R}$. Both algorithms perform the same computations, except that the exponential mechanism adds *one-sided* Laplace noise whereas Report-noisy-max adds regular Laplace noise. Thus, accuracy for both algorithms is verified in essentially the same way. We focus on Report-noisy-max to avoid defining one-sided Laplace.

Report-noisy-max finds an element of a finite set $\mathcal{R}$ that approximately maximizes some *quality score* function qscore, which takes as input an element $r \in \mathcal{R}$ and a database d. Operationally, Report-noisy-max computes the quality score for each element of $\mathcal{R}$, adds Laplace noise, and returns the element with the highest (noisy) value. We can implement this algorithm with the following code, using syntactic sugar for arrays:

```
proc RNM(R, d) :
  flag ← 1; best ← 0;
  while R ≠ ∅ do
    r ← pick(R); noisy[r] ←$ L_{ε/2}(qscore(r, d));
    if (noisy[r] > best ∨ flag = 1) then
      flag ← 0; r* ← r; best ← noisy[r];
    R ← R \ {r};
return r*;
```

The scale $\epsilon/2$ of the Laplace distribution ensures an appropriate level of differential privacy under certain assumptions; we will not discuss privacy in the remainder.

▶ **Theorem 6.** *Let $\beta \in (0, 1)$, and let $res \in \mathcal{R}$ be the output of Report-noisy-max on input d and quality score* qscore*. Then, we have the following judgment:*

$$\vdash_\beta \mathrm{RNM} : \top \Longrightarrow \forall r \in \mathcal{R}.\ \mathsf{qscore}(res, d) > \mathsf{qscore}(r, d) - \frac{4}{\epsilon} \log \frac{|\mathcal{R}|}{\beta}\,.$$

where $|\mathcal{R}|$ denotes the size of $\mathcal{R}$. This corresponds to the existing accuracy guarantee for Report-noisy-max (see, e.g., Dwork and Roth [15]).

Roughly, this theorem states that while the result *res* may not be the element with the absolute highest quality score, its quality score is not far below the quality score of any other element. For a brief sanity check, note that the guarantee weakens as we increase the range $\mathcal{R}$, or decrease the failure probability β.

The proof of accuracy is based on an instantiation of the rule [LapAcc] with e set to $\mathsf{qscore}(r, d)$, β set to $\beta/|\mathcal{R}|$, and ϵ set to $\epsilon/2$. First, we can show

$$\vdash_{\beta/|\mathcal{R}|} c : \top \Longrightarrow |\mathit{noisy}[r] - \mathsf{qscore}(r, d)| < \frac{2}{\epsilon} \log \frac{|\mathcal{R}|}{\beta} .$$

where c is the loop body. Since the loop runs for $|\mathcal{R}|$ iterations, we also have

$$\vdash_{\beta} \text{RNM} : \top \Longrightarrow \forall r \in \mathcal{R}.\ |\mathit{noisy}[r] - \mathsf{qscore}(r, d)| < \frac{2}{\epsilon} \log \frac{|\mathcal{R}|}{\beta} .$$

In order to prove this judgment, the loop invariant quantifies over all previously seen $r \in \mathcal{R}$. Combined with a straightforward invariant showing that r^* stores the index of the current maximum (noisy) score, the above judgment suffices to prove the accuracy guarantee for Report-noisy-max (Theorem 6).

3.2 Sparse Vector algorithm

Our second example is the *Sparse Vector algorithm*, which indicates which numeric queries take value (approximately) above some threshold value (see, e.g., Dwork and Roth [15]). Simpler approaches can accomplish this task by releasing the noisy answer to all queries and then comparing with the threshold, but the resulting error then grows linearly with the total number of queries. Sparse Vector does not release the noisy answers, but the resulting error grows only *logarithmically* with the total number of queries – a substantial improvement. The differential privacy property of Sparse Vector was recently formally verified [6]; here, we consider the accuracy property.

In the non-interactive setting, the algorithm takes as input a list of queries $q_1, q_2, \ldots$, a database d, and a numeric threshold $t \in \mathbb{R}$.[2] First, we add Laplace noise to the threshold t to calculate the noisy threshold T. Then, we evaluate each query q_i on d, add Laplace noise, and check if the noisy value exceeds T. If so, we output $\top$; if not, we output $\bot$.

Sparse Vector also works in the *interactive* setting. Here, the algorithm is fed one query at a time, and must process this query (producing $\bot$ or $\top$) before seeing the next query. The input may be adaptive – future queries may depend on the answers to earlier queries.

We focus on the interactive version; the non-interactive version can be handled similar to Report-noisy-max. We break the code into two pieces. The first piece initializes variables and computes the noisy threshold, while the second piece accepts a single new query and returns the answer.

$$\begin{array}{l}
\mathsf{proc}\ \text{SV.}\textsc{Init}(T_{in}, \epsilon_{in}) : \\
\quad \epsilon \leftarrow \epsilon_{in}; \\
\quad T \xleftarrow{\$} \mathcal{L}_{\epsilon/2}(T_{in});
\end{array}
\qquad
\begin{array}{l}
\mathsf{proc}\ \text{SV.}\textsc{Step}(q) : \\
\quad a \xleftarrow{\$} \mathcal{L}_{\epsilon/4}(\mathsf{evalQ}(q, d)); \\
\quad \mathsf{if}\ (a < T)\ \mathsf{then}\ \{z \leftarrow \bot;\}\ \mathsf{else}\ \{z \leftarrow \top;\} \\
\mathsf{return}\ z;
\end{array}$$

[2] In some presentations, the algorithm is also parameterized by the maximum number k of queries to answer. This feature is important for privacy but not accuracy, so we omit it. It is not difficult to extend the accuracy proof for answering at most k queries.

The main procedure performs initialization, and then enters into an interactive loop between the external procedure $\mathcal{A}$ – which supplies the queries – and the Sparse Vector procedure SV.Step:

```
proc SV.main(Q, T, ε) :
  SV.Init(T, ε);
  u ← 0; ans[u] ← ⊥;
  while (u < Q) do
    u ← u + 1;
    q[u] ← A(ans[u − 1]);
    ans[u] ← SV.Step(q[u]);
return ans;
```

Sparse Vector satisfies the following accuracy guarantee.

▶ **Theorem 7.** *Let $\beta \in (0,1)$. We have*

$$\vdash_{\beta} \text{SV.main}(Q,T) : \top \Longrightarrow \forall j \in \{1,\ldots,Q\}.\ \Phi(q[j],d),\ \textit{where}$$

$$\Phi(q,d) \triangleq \left(res = \top \rightarrow \mathsf{evalQ}(q,d) > t - \frac{6}{\epsilon}\log\frac{Q+1}{\beta}\right) \wedge \left(res = \bot \rightarrow \mathsf{evalQ}(q,d) < t + \frac{6}{\epsilon}\log\frac{Q+1}{\beta}\right).$$

This judgment corresponds to the accuracy guarantee for Sparse Vector from (see, e.g., Dwork and Roth [15]). Note that the error term depends logarithmically on the total number of queries Q, a key feature of Sparse Vector.

To prove this theorem, we first specify the procedures SV.Init and SV.Step. For initialization, we have

$$\vdash_{\beta/(Q+1)} \text{SV.Init}(T,\epsilon) : \top \Longrightarrow \Phi_t \quad \text{where} \quad \Phi_t \triangleq |t-T| < \frac{2}{\epsilon}\log\frac{Q+1}{\beta} \wedge \epsilon = \epsilon_{in}\,.$$

For the interactive step, we have

$$\vdash_{\beta/(Q+1)} \text{SV.Step}(q) : \Phi_t \Longrightarrow \Phi_t \wedge \Phi(q,d)\,.$$

Combining these two judgments, we can prove accuracy for SV.main (Theorem 7).

3.3 Online Multiplicative Weights

Our final example demonstrates how we can use the union bound to analyze a complex combination of several interactive algorithms, yielding sophisticated accuracy proofs. We will verify the *Online Multiplicative Weights* (OMW) algorithm first proposed by Hardt and Rothblum [21] and later refined by Gupta et al. [20]. Like Sparse Vector, this interactive algorithm can handle adaptive queries while guaranteeing error logarithmic in the number of queries. Unlike Sparse Vector, OMW produces approximate answers to the queries instead of just a bit representing above or below threshold.

At a high level, OMW iteratively constructs a *synthetic* version of the true database. The user can present various linear queries to the algorithm, which applies the Sparse Vector algorithm to check whether the error of the synthetic database on this query is smaller than some threshold. If so, the algorithm simply returns the approximate answer. Otherwise, it

updates the synthetic database using the *multiplicative weights* update rule to better model the true database, and answers the query by adding Laplace noise to the true answer. An inductive argument shows that after enough updates, the synthetic database must be similar to the true database on *all* queries. At this point, we can answer all subsequent queries using the synthetic database alone.

In code, the following procedure implements the Online Multiplicative Weights algorithm.

```
proc MW-SV.MAIN(d, α, ϵ, Q, X, n) :
  η ← α/2n; T ← 2α; c ← 4n² ln(X)/α²;                    set parameters
  u ← 0; k ← 0; ans[k] ← ⊥;                               initialize variables
  mwdb ← MW.INIT(η, X, n); SV.INIT(T, ϵ/4c);              initialize MW and SV
  while (k < Q) do                                        main loop
    k ← k + 1;                                            increment count of queries
    q[k] ← 𝒜(ans[k − 1], mwdb);                           get next query
    approx ← evalQ(q[k], mwdb);                           calculate approx answer
    exact ← evalQ(q[k], d);                               calculate exact answer
    if (k ≥ c) then ans[k] ← approx;                      enough updates, use approx answer
    else
      err> ← error(q[k], mwdb); at ← SV.STEP(err>);       check if approx answer is high
      err< ← invQ(error(q[k], mwdb)); bt ← SV.STEP(err<); check if approx answer is low
      if (at ≠ ⊥ ∨ bt ≠ ⊥) then                           large error
        u ← u + 1;                                        increment count of updates
        if at ≠ ⊥ then up ← q[k];                         approx answer too high
        else up ← negQ(q[k]);                             approx answer too low
        mwdb ← MW.STEP(mwdb, up);                         update synthetic db
        ans[k] ←$ ℒ_{ϵ/2c}(exact);                        estimate true answer
      else                                                small error, do not update
        ans[k] ← approx;                                  answer using approx answer
return ans;
```

Online multiplicative weights satisfies the following accuracy guarantee.

▶ **Theorem 8.** *Let $\beta \in (0,1)$. Then,*

$$\vdash_\beta \text{MW-SV.MAIN}(d, \alpha, \epsilon, Q, X, n) : \alpha \geq \max(\alpha_{sv}, \alpha_{lap}) \implies \\ \forall j.\ j \in \{1, \ldots, Q\} \to |res[j] - \mathsf{evalQ}(q[j], d)| \leq \alpha,$$

$$\textit{where} \quad \gamma \triangleq 4n^2 \ln(X)/\alpha^2, \quad \alpha_{sv} \triangleq \tfrac{24\gamma}{\epsilon} \log \tfrac{2(Q+1)}{\beta}, \quad \textit{and} \quad \alpha_{lap} \triangleq \tfrac{4\gamma}{\epsilon} \log \tfrac{2\gamma}{\beta}.$$

In words, the answers to all the supplied queries are within α of the true answer if α is sufficiently large. The above judgment reflects the accuracy guarantee first proved by Hardt and Rothblum [21] and later generalized by Gupta et al. [20].

The main routine depends on the *multiplicative weights* subroutine (MW), which maintains and updates the synthetic database. Roughly, MW takes as input the current synthetic database and a query where the synthetic database gives an answer that is far from the true answer. Then, MW improves the synthetic database to better model the true database. Our implementation of MW consists of two subroutines: MW.INIT initializes the synthetic database, and MW.STEP updates the current database with a query that has high error. The code for these subroutines is somewhat technical, and we will not present it here.

Instead, we will present their specifications, which are given in terms of an expression $\Psi(x, d)$ where x is the current synthetic database and d is the true database. We omit the definition of Ψ and focus on its three key properties:

- $\Psi(x, d) \geq 0$;
- $\Psi(x, d)$ is initially bounded for the initial synthetic database; and
- $\Psi(x, d)$ decreases each time we update the synthetic database.

Functions satisfying these properties are often called *potential functions*.

The first property follows from the definition of Ψ, while the second and third properties are reflected by the specifications of the MW procedures. Concretely, we can bound the initial value of Ψ with the following specification for MW.INIT:

$$\vdash_0 \text{MW.INIT}(\eta, X, n) : \top \Longrightarrow \Psi(res, d) \leq \ln X \,.$$

We can also show that Ψ decreases with the following specification for MW.STEP:

$$\vdash_0 \text{MW.STEP}(x, q) : \top \Longrightarrow \Psi(x, d) - \Psi(res, d) \geq \eta(\mathsf{evalQ}(q, x) - \mathsf{evalQ}(q, d))/n - \eta^2 \,.$$

We make two remarks. First, these specifications crucially rely on the fact that q is a linear query. Second, both procedures are deterministic. For such procedures, the fragment of aHL with index $\beta = 0$ corresponds precisely to standard Hoare logic.

Now, let us briefly consider the key points in proving the main specification (Theorem 8). First, the key part of the invariant for the main loop is $\Psi(mwdb, d) \leq \log X - u \cdot \alpha^2/4n^2$. Roughly, Ψ is initially at most $\log X$ by the specification for MW.INIT, and every time we call MW.STEP we decrease Ψ by at least $\alpha^2/4n^2$ if the update query up has error at least α. Since Ψ is always non-negative, we can find at most c queries with high error – after c updates, the synthetic database $mwdb$ must give accurate answers on all queries.

Prior to making c updates, there are two cases for each query. If at least one of the Sparse Vector calls returns above threshold, we set the update query up to be $q[u]$ if the approximate answer is too high, otherwise we set up to be the negated query $\mathsf{neqQ}(q[u])$ if the approximate answer is too low. With this choice of update query, we can show that

$$\mathsf{evalQ}(up, mwdb) - \mathsf{evalQ}(up, d) \geq \alpha$$

so Ψ decreases by at least $\alpha^2/4n^2$. Then, we answer the original query $q[u]$ by adding Laplace noise, so our answer is also within α of the true answer. Otherwise, if both Sparse Vector calls return below threshold, then the query $q[u]$ is answered well by our approximation $mwdb$ and there is no need to update $mwdb$ or access the real database d.

The above reasoning assumes that Sparse Vector and the Laplace mechanisms are sufficiently accurate. To guarantee the former, notice that the Sparse Vector subroutine will process at most $2Q$ queries, so we assume that α is larger than the error α_{sv} guaranteed by Theorem 7 for $2Q$ queries and failure probability $\beta/2$. To guarantee the latter, notice that we sample Laplace noise at most c times – once for each update step – so we assume that α is larger than the error α_{lap} guaranteed by [LapAcc] for failure probability $\beta/2c$; by a union bound, all Laplace noises are accurate except with probability $\beta/2$. Taking $\alpha \geq \max(\alpha_{sv}, \alpha_{lap})$, both accuracy guarantees hold except with probability at most β, and we have the desired proof of accuracy for OMW (Theorem 8).

4 Related work

The semantics of probabilistic programming languages has been studied extensively since the late 70s. Kozen's seminal paper [28] studies two semantics for a core probabilistic imperative language. Other important work investigates using monads to structure the semantics of probabilistic languages; e.g. Jones and Plotkin [24]. More recent works study the semantics of

probabilistic programs for applications like statistical computations [9], probabilistic inference for machine learning [10], probabilistic modeling for software defined networks [17], and more.

Likewise, deductive techniques for verifying probabilistic programs have a long history. Ramshaw [35] proposes a program logic with basic assertions of the form $\Pr[E] = p$. Hart el al. [22], Sharir et al. [39] propose a method using intermediate assertions and invariants for proving general properties of probabilistic programs. Kozen [29] introduces PPDL, a logic that can reason about expected values of general measurable functions. Morgan et al. [34] (see McIver and Morgan [31] for an extended account) propose a verification method based on computing *greatest pre-expectations*, a probabilistic analogue of Dijkstra's weakest pre-conditions. Hurd et al. [23] formalize their approach using the HOL theorem prover. Other approaches based on interactive theorem provers include the work of Audebaud and Paulin-Mohring [1], who axiomatize (discrete) probability theory and verify some examples of randomized algorithms using the Coq proof assistant. Gretz et al. [19] extend the work of Morgan et al. [34] with a formal treatment of conditioning. More recently, Rand and Zdancewic [36] formalize another Hoare logic for probabilistic programs using the Coq proof assistant. Barthe et al. [3] implement a general-purpose logic in the EasyCrypt framework, and verify a representative set of randomized algorithms. Kaminski et al. [25] develop a weakest precondition logic to reason about expected run-time of probabilistic programs.

Most of these works support general probabilistic reasoning and additional features like non-determinism, so they most likely could formalize the examples that we consider. However, our logic aHL aims at a sweet spot in the design space, combining expressivity with simplicity of the assertion language. The design of aHL is inspired by existing *relational* program logics, such as pRHL [7] and apRHL [8]. These logics support rich proofs about probabilistic properties with purely non-probabilistic assertions, using a powerful coupling abstraction from probability theory [4] rather than the union bound.

Finally, there are many algorithmic techniques for verifying probabilistic programs. Probabilistic model-checking is a successful line of research that has delivered mature and practical tools and addressed a broad range of case studies; Baier and Katoen [2], Katoen [26], Kwiatkowska et al. [30] cover some of the most interesting developments in the field. Abstract interpretation of probabilistic programs is another rich source of techniques; see e.g. Cousot and Monerau [13], Monniaux [33]. Katoen et al. [27] infer linear invariants for the pGCL language of Morgan et al. [34]. There are several approaches based on martingales for reasoning about probabilistic loops; Chakarov and Sankaranarayanan [11, 12] use martingales for inferring expectation invariants, while Ferrer Fioriti and Hermanns [16] use martingales for analyzing probabilistic termination. Sampson et al. [38] use a mix of static and dynamic analyses to check probabilistic assertions for probabilistic programs.

5 Conclusion and perspective

We propose aHL, a lightweight probabilistic Hoare logic based on the union bound. Our logic can prove properties about bad events in cryptography and accuracy of differentially private mechanisms. Of course, there are examples that we cannot verify. For instance, reasoning involving independence of random variables, a common tool when analyzing randomized algorithms, is not supported. Accordingly, a natural next step is to explore logical methods for reasoning about independence, or to embed aHL into a more general system like pGCL.

References

1. Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. *Science of Computer Programming*, 74(8):568–589, 2009. URL: https://www.lri.fr/~paulin/ALEA/random-scp.pdf.
2. Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
3. Gilles Barthe, Thomas Espitau, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Formal certification of randomized algorithms. 2015. URL: http://justinh.su/files/docs/BEGGHS15paper.pdf.
4. Gilles Barthe, Thomas Espitau, Benjamin Grégoire, Justin Hsu, Léo Stefanesco, and Pierre-Yves Strub. Relational reasoning via probabilistic coupling. In *International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR), Suva, Fiji*, volume 9450, pages 387–401, 2015. URL: http://arxiv.org/abs/1509.03476.
5. Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, César Kunz, and Pierre-Yves Strub. Proving differential privacy in Hoare logic. In *IEEE Computer Security Foundations Symposium (CSF), Vienna, Austria*, 2014. URL: http://arxiv.org/abs/1407.2988, arXiv:Yes.
6. Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *IEEE Symposium on Logic in Computer Science (LICS), New York, New York*, 2016. To appear. URL: http://arxiv.org/abs/1601.05047, arXiv:Yes.
7. Gilles Barthe, Benjamin Grégoire, and Santiago Zanella-Béguelin. Formal certification of code-based cryptographic proofs. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Savannah, Georgia*, pages 90–101, New York, 2009. URL: http://research.microsoft.com/pubs/185309/Zanella.2009.POPL.pdf.
8. Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella-Béguelin. Probabilistic relational reasoning for differential privacy. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Philadelphia, Pennsylvania*, pages 97–110, 2012. URL: http://certicrypt.gforge.inria.fr/2012.POPL.pdf.
9. Sooraj Bhat, Ashish Agarwal, Richard Vuduc, and Alexander Gray. A type theory for probability density functions. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Philadelphia, Pennsylvania*, pages 545–556, 2012. doi:10.1145/2103656.2103721.
10. Johannes Borgström, Andrew D. Gordon, Michael Greenberg, James Margetson, and Jurgen Van Gael. Measure transformer semantics for Bayesian machine learning. *Logical Methods in Computer Science*, 9(3), 2013. URL: 10.2168/LMCS-9(3:11)2013;http://arxiv.org/abs/1308.0689.
11. Aleksandar Chakarov and Sriram Sankaranarayanan. Probabilistic program analysis with martingales. In *International Conference on Computer Aided Verification (CAV), Saint Petersburg, Russia*, pages 511–526, 2013. URL: https://www.cs.colorado.edu/~srirams/papers/cav2013-martingales.pdf.
12. Aleksandar Chakarov and Sriram Sankaranarayanan. Expectation invariants as fixed points of probabilistic programs. In *International Symposium on Static Analysis (SAS), Munich, Germany*, volume 8723 of *Lecture Notes in Computer Science*, pages 85–100. Springer-Verlag, 2014. URL: https://www.cs.colorado.edu/~srirams/papers/sas14-expectations.pdf.
13. Patrick Cousot and Michael Monerau. Probabilistic abstract interpretation. In Helmut Seidl, editor, *European Symposium on Programming (ESOP), Tallinn, Estonia*, volume 7211 of *Lecture Notes in Computer Science*, pages 169–193. Springer, 2012. URL: http://www.di.ens.fr/~cousot/publications.www/Cousot-Monerau-ESOP2012-extended.pdf.

14 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *IACR Theory of Cryptography Conference (TCC), New York, New York*, pages 265–284, 2006. doi:10.1007/11681878_14.

15 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014. doi:10.1561/0400000042.

16 Luis María Ferrer Fioriti and Holger Hermanns. Probabilistic termination: Soundness, completeness, and compositionality. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Mumbai, India*, pages 489–501. ACM, 2015. URL: http://www.ae-info.org/attach/User/Hermanns_Holger/Publications/FH-POPL15.pdf.

17 Nate Foster, Dexter Kozen, Konstantinos Mamouras, Mark Reitblatt, and Alexandra Silva. Probabilistic NetKAT. In *European Symposium on Programming (ESOP), Eindhoven, The Netherlands*, Lecture Notes in Computer Science, 2016.

18 Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C Pierce. Linear dependent types for differential privacy. In *ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages (POPL), Rome, Italy*, pages 357–370, 2013. URL: http://dl.acm.org/citation.cfm?id=2429113.

19 Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. Prinsys – on a quest for probabilistic loop invariants. In *International Conference on Quantitative Evaluation of Systems (QEST)*, pages 193–208, 2013.

20 Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In *IACR Theory of Cryptography Conference (TCC), Taormina, Italy*, pages 339–356, 2012. URL: http://arxiv.org/abs/1107.3731.

21 Moritz Hardt and Guy N Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *IEEE Symposium on Foundations of Computer Science (FOCS), Las Vegas, Nevada*, pages 61–70, 2010. URL: http://www.mit.edu/~rothblum/papers/pmw.pdf.

22 Sergiu Hart, Micha Sharir, and Amir Pnueli. Termination of probabilistic concurrent programs. In *ACM Symposium on Principles of Programming Languages (POPL), Albuquerque, New Mexico*, pages 1–6, 1982. doi:10.1145/582153.582154.

23 Joe Hurd, Annabelle McIver, and Carroll Morgan. Probabilistic guarded commands mechanized in HOL. *Theoretical Computer Science*, 346(1):96–112, 2005.

24 C. Jones and Gordon D. Plotkin. A probabilistic powerdomain of evaluations. In *IEEE Symposium on Logic in Computer Science (LICS), Asilomar, California*, pages 186–195, 1989. doi:10.1109/LICS.1989.39173.

25 Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Federico Olmedo. Weakest precondition reasoning for expected run-times of probabilistic programs. In *European Symposium on Programming (ESOP), Eindhoven, The Netherlands*, Lecture Notes in Computer Science, 2016.

26 Joost-Pieter Katoen. Perspectives in probabilistic verification. In *IEEE/IFIP International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pages 3–10, 2008.

27 Joost-Pieter Katoen, Annabelle McIver, Larissa Meinicke, and Carroll Morgan. Linear-invariant generation for probabilistic programs. In Radhia Cousot and Matthieu Martel, editors, *International Symposium on Static Analysis (SAS), Perpignan, France*, volume 6337 of *Lecture Notes in Computer Science*, pages 390–406. Springer, 2010.

28 Dexter Kozen. Semantics of probabilistic programs. In *IEEE Symposium on Foundations of Computer Science (FOCS), San Juan, Puerto Rico*, pages 101–114, 1979.

29 Dexter Kozen. A probabilistic PDL. *J. Comput. Syst. Sci.*, 30(2):162–178, 1985.

30 Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In *International Conference on Tools and*

Algorithms for the Construction and Analysis of Systems (TACAS), Tallinn, Estonia, pages 52–66, 2002.

31 A. McIver and C. Morgan. *Abstraction, Refinement, and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer, 2005.

32 Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *IEEE Symposium on Foundations of Computer Science (FOCS), Providence, Rhode Island*, pages 94–103, 2007. doi:10.1109/FOCS.2007.41.

33 David Monniaux. Abstract interpretation of probabilistic semantics. In Jens Palsberg, editor, *International Symposium on Static Analysis (SAS), Santa Barbara, California*, volume 1824 of *Lecture Notes in Computer Science*, pages 322–339. Springer, 2000.

34 Carroll Morgan, Annabelle McIver, and Karen Seidel. Probabilistic predicate transformers. *ACM Transactions on Programming Languages and Systems*, 18(3):325–353, 1996.

35 Lyle Harold Ramshaw. *Formalizing the Analysis of Algorithms*. PhD thesis, Stanford University, 1979.

36 Robert Rand and Steve Zdancewic. VPHL: A verified partial-correctness logic for probabilistic programs. In *Mathematical Foundations of Program Semantics (MFPS)*, 2015.

37 Jason Reed and Benjamin C Pierce. Distance makes the types grow stronger: A calculus for differential privacy. In *ACM SIGPLAN International Conference on Functional Programming (ICFP), Baltimore, Maryland*, 2010. URL: http://dl.acm.org/citation.cfm?id=1863568.

38 Adrian Sampson, Pavel Panchekha, Todd Mytkowicz, Kathryn S McKinley, Dan Grossman, and Luis Ceze. Expressing and verifying probabilistic assertions. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), Edinburgh, Scotland*, page 14, 2014. URL: http://research.microsoft.com/pubs/211410/passert-pldi2014.pdf.

39 Micha Sharir, Amir Pnueli, and Sergiu Hart. Verification of probabilistic programs. *SIAM Journal on Computing*, 13(2):292–314, 1984. doi:10.1137/0213021.

The Decidable Properties of Subrecursive Functions

Mathieu Hoyrup

LORIA, Inria, Villers-lès-Nancy, France
mathieu.hoyrup@inria.fr

Abstract

What can be decided or semidecided about a primitive recursive function, given a definition of that function by primitive recursion? What about subrecursive classes other than primitive recursive functions? We provide a complete and explicit characterization of the decidable and semidecidable properties. This characterization uses a variant of Kolmogorov complexity where only programs in a subrecursive programming language are allowed. More precisely, we prove that all the decidable and semidecidable properties can be obtained as combinations of two classes of basic decidable properties: (i) the function takes some particular values on a finite set of inputs, and (ii) every finite part of the function can be compressed to some extent.

1998 ACM Subject Classification F.1.1 Models of Computation, F.4.3 Formal Languages/Decision Problems

Keywords and phrases Rice theorem, subrecursive class, decidable property, Kolmogorov complexity, compressibility

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.108

1 Introduction

What can be decided about a function $f : \mathbb{N} \to \mathbb{N}$ if f is represented by a program computing it? What can be semidecided?

In the 50's, many computability theory results have been proved in order to answer these questions. The answers depend on the class of functions considered.

Partial computable functions.

For the class of partial computable functions, Rice [11] proved that no non-trivial property is decidable, and Shapiro [12] refined it by characterizing the semidecidable properties. These results show that having a program computing f does not give more information than having an oracle giving the values of f, in the sense that the two presentations induce the same classes of decidable and semidecidable properties. In other words, the only way of exploiting a program computing f is to execute it on any input to obtains the values of f. Hence the code of the program contains no more information than a black-box containing the program.

Total computable functions

For the class of total computable functions, Kreisel, Lacombe, Shœnfield [8] and independently Ceitin [2] characterized the decidable properties. Again they are the same whether the function is presented by a program or by an oracle.

However, Friedberg [4] showed that the semidecidable properties of total computable functions do not admit such a characterization. In that case, having a program computing

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 108; pp. 108:1–108:13


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

f gives more information than having the values of f. In [6] we proved that the extra information is simply the size of the program. A remaining open problem is to obtain a nice characterization of the semidecidable properties of total computable functions, when they are presented by programs.

Subrecursive classes

In this paper we investigate the case of a subrecursive class, i.e. a class of total computable functions that can be effectively enumerated. Examples of such classes are the primitive recursive functions, complexity classes such as FPTIME, or the class of provably total functions in Peano Arithmetic. Here the function in the class is presented by a program in a restricted programming language. As for the whole class of total computable functions, the semidecidable properties are not the same if the function is presented by an oracle. However we obtain a characterization of the semidecidable properties, when the function is presented by a subrecursive program. This characterization uses a version of Kolmogorov complexity restricted to a subrecursive programming language. This is the main result of the paper.

We also prove that the semidecidable properties of total computable functions do not admit an analogous characterization.

We also discuss the difference between having an oracle giving the values of a function, and a black-box containing a program computing that function. The difference is that no assumption is made on the time it takes for the oracle to answer, while a program in a black-box has particular halting times. We show that in general it does not make any difference, but we also show a situation where the halting times of the black-box can be exploited.

1.1 Notations

Let $\mathbb{N}^*$ be the set of finite sequences of natural numbers and $\mathbb{N}^{\mathbb{N}}$ the Baire space of functions from $\mathbb{N}$ to $\mathbb{N}$. Given $f \in \mathbb{N}^{\mathbb{N}}$ and $n \in \mathbb{N}$, $f\restriction_n$ denotes the finite sequence $(f(0), \dots, f(n-1)) \in \mathbb{N}^*$. We say that $f \in \mathbb{N}^{\mathbb{N}}$ extends $v = (v_0, \dots, v_{n-1}) \in \mathbb{N}^*$ if $f(0) = v_0, \dots, f(n-1) = v_{n-1}$, i.e. if $f\restriction_n = v$. We denote by $[v] \subseteq \mathbb{N}^{\mathbb{N}}$ the set of all extensions of v and call it a cylinder. The Baire space is endowed with the topology whose open sets are unions of cylinders. An effective open set is the union of a computable sequence of cylinders.

2 Decidable and semidecidable properties

In this paper, a *subrecursive class of functions* is simply a class $\mathscr{C}$ of total computable functions that can be computably enumerated: there is a numbering $\mathscr{C} = \{f_i : i \in \mathbb{N}\}$ such that f_i is computable uniformly in i, i.e. the mapping $(i, n) \mapsto f_i(n)$ is computable. If $f \in \mathscr{C}$ then a $\mathscr{C}$-index of f is any i such that $f = f_i$ (a function may have several indices). There are usually many ways of indexing such a class, and they may induce different decidable properties. A thorough investigation about indexings of subrecursive classes can be found in [7].

Examples of subrecursive classes are: the primitive recursive functions, the class FPTIME of polynomial-time computable functions, the class of provably total computable functions in Peano arithmetic. A numbering of a class can be obtained from a numbering of the programs in a subrecursive programming language, or a restricted model of computation. Hence having an index of a function is usually equivalent to having a program for that function, in the restricted language. Famous examples of restricted programming language are: definitions

by primitive recursion, LOOP programming language [10], polynomially clocked Turing machines, proofs of totality in Peano Arithmetic.

Observe that two programming languages admitting computable translation procedures in both directions induce the same decidable and semidecidable properties. This is for instance the case of definitions by primitive recursion and LOOP programs.

Let $A \subseteq \mathscr{C}$. First observe that the property $f \in A$ is semidecidable given f by an oracle exactly when A is the intersection of an effective open subset of the Baire space with $\mathscr{C}$. Indeed, when the machine semideciding $f \in A$ accepts f in finite time so it has only read a finite segment of f hence it will accept all functions in some cylinder $[f\restriction_n]$. The property $f \in A$ is decidable given f by an oracle exactly when both A and $\mathscr{C} \setminus A$ are the intersections of effective open sets with $\mathscr{C}$.

The goal of this paper is to obtain a similar understanding of the decidable and semidecidable sets $A \subseteq \mathscr{C}$, when $f \in \mathscr{C}$ is presented by a $\mathscr{C}$-index rather than an oracle.

In order to investigate this problem we introduce a notion of Kolmogorov complexity adapted to the class $\mathscr{C}$.

▶ **Definition 1.** The $\mathscr{C}$-complexity of $f : \mathbb{N} \to \mathbb{N}$ is

$$K_{\mathscr{C}}(f) = \begin{cases} \min\{i : f_i = f\} & \text{if } f \in \mathscr{C}, \\ +\infty & \text{otherwise.} \end{cases}$$

If $v = (v_0, \dots, v_n)$ is a finite sequence of natural numbers then its $\mathscr{C}$-complexity is

$$K_{\mathscr{C}}(v) = \min\{i : f_i \text{ extends } v\} = \min\{K_{\mathscr{C}}(f) : f \in [v]\}.$$

If no f_i extends v then $K_{\mathscr{C}}(v) = +\infty$. Observe that for $f : \mathbb{N} \to \mathbb{N}$, $K_{\mathscr{C}}(f\restriction_n)$ is nondecreasing and converges to $K_{\mathscr{C}}(f)$ (which may be infinite). By the assumptions on $\mathscr{C}$, the quantity $K_{\mathscr{C}}(v)$ is computable from v (when it is finite – in, general, the predicate $K_{\mathscr{C}}(v) = i$ is decidable), contrary to usual Kolmogorov complexity which is upper semicomputable only. However $K_{\mathscr{C}}$ usually does not belong to the class $\mathscr{C}$ (modulo encoding of $\mathbb{N}^*$ in $\mathbb{N}$).

It may seem more consistent with usual notions of Kolmogorov complexity (see e.g. [9]) to take for instance $\log(i)$ instead of i in the definition, or to use a machine that is universal for the class $\mathscr{C}$ and define $K_{\mathscr{C}}$ in terms of the size of its inputs. All these choices are equally acceptable and lead exactly to the same result. The important point is that for each such notion of complexity K', an upper bound on $K_{\mathscr{C}}(f)$ can be uniformly computed from any upper bound on $K'(f)$ and vice-versa. Here we take the simplest definition of complexity following directly from the enumeration of $\mathscr{C}$, to avoid technicality.

2.1 An index gives more information than an oracle

Kreisel-Lacombe-Shœnfield/Ceitin's theorem implies that the properties of total computable functions that are decidable from indices are also decidable from oracles. In essence, Rice's theorem states the same for the class of partial computable functions. In general the situation is different when restricting to some subrecursive class: there exist properties that are decidable from indices but not from oracles.

Let us first give a concrete class of such properties. A *computable order* is a non-decreasing unbounded computable function $h : \mathbb{N} \to \mathbb{N}$, such as $\lfloor \log(n) \rfloor$, n^2 or 2^n for instance.

▶ **Definition 2.** Let $\mathscr{C}$ be a subrecursive class of functions and h a computable order. We define the set $A_{\mathscr{C},h}$ of $(\mathscr{C}, h)$-*anticomplex* functions as

$$A_{\mathscr{C},h} = \{f : \mathbb{N} \to \mathbb{N} : \forall n, K_{\mathscr{C}}(f\restriction_n) \leq h(n)\}.$$

We borrow the terminology from [3], where the notion of anti-complex set is defined in terms of usual Kolmogorov complexity, and is studied from a computability-theoretic perspective.

▶ **Proposition 3.** For $f \in \mathscr{C}$, the property $f \in A_{\mathscr{C},h}$ is decidable given any $\mathscr{C}$-index of f.

Proof. Given an index i for f, one has $K_{\mathscr{C}}(f\restriction_n) \leq K_{\mathscr{C}}(f) \leq i$ for all n, so f belongs to $A_{\mathscr{C},h}$ if and only if $K_{\mathscr{C}}(f\restriction_n) \leq h(n)$ for all n such that $h(n) < i$. This property is decidable as $K_{\mathscr{C}}(f\restriction_n)$ is computable from i and n and only a finite number of values of n has to be checked. ◀

Note that it is important that h be unbounded. One can easily show that if for each $g \in \mathscr{C}$ the set $\{j \in \mathbb{N} : f_j = g\}$ is not decidable (usual subrecursive classes satisfy this condition), then when h is bounded the property $f \in A_{\mathscr{C},h}$ is not decidable given any $\mathscr{C}$-index of f.

In general $A_{\mathscr{C},h}$ is no more decidable if instead of an index of f one is only given f as oracle.

▶ **Proposition 4.** If $\mathscr{C}$ is dense in $\mathbb{N}^{\mathbb{N}}$ then $A_{\mathscr{C},h}$ has empty interior in $\mathscr{C}$ (i.e. does not contain the intersection of a cylinder with $\mathscr{C}$), therefore $A_{\mathscr{C},h}$ is not semidecidable when the input function is given as oracle.

Proof. For each $u = (u_0, \dots, u_{n-1}) \in \mathbb{N}^*$, there exist only finitely many $i \in \mathbb{N}$ such that $K_{\mathscr{C}}(u_0, \dots, u_{n-1}, i) \leq h(n+1)$. Take any i outside this finite set: the cylinder $[u_0, \dots, u_{n-1}, i]$ is disjoint from $A_{\mathscr{C},h}$ but intersects $\mathscr{C}$, so $[u] \cap \mathscr{C}$ is not contained in $A_{\mathscr{C},h}$. ◀

All the usual subrecursive classes are dense in $\mathbb{N}^{\mathbb{N}}$. Observe that in computational complexity theory, one is more often interested in classes of *problems* rather than *functions*. Hence $\mathscr{C}$ could be the class of characteristic functions of subsets of $\mathbb{N}$ in some complexity class, such as P. In that case, $\mathscr{C}$ is not dense in $\mathbb{N}^{\mathbb{N}}$, however it is dense in $\{0,1\}^{\mathbb{N}}$ and a similar result holds.

▶ **Proposition 5.** The same result holds if $\mathscr{C}$ is dense in $\{0,1\}^{\mathbb{N}}$ and h is sufficiently small.

Proof. Let g be a computable order such that the number of finite sequences v such that $K_{\mathscr{C}}(v) \leq k$ is bounded by $g(k)$. If $K_{\mathscr{C}}$ is the notion of complexity from Definition 1 then one can take $g(k) = k + 1$. For other notions of complexity based on length of binary programs, one could take $g(k) = 2^{k+1}$ instead.

Let h be a computable order such that $g \circ h(n) = o(2^n)$ (take for instance $h(n) = n$ in our case). Given $u \in \{0,1\}^*$, there exists $n \geq |u|$ such that $g \circ h(n) < 2^{n-|u|}$. By definition of g there are at most $g(h(n)) < 2^{n-|u|}$ finite sequences v such that $K_{\mathscr{C}}(v) \leq h(n)$. As there exist $2^{n-|u|}$ binary extensions of u of length n, at least one of them satisfies $K_{\mathscr{C}}(v) > h(n)$. Hence $[v] \cap \mathscr{C}$ is disjoint from $A_{\mathscr{C},h}$ and non-empty, so $A_{\mathscr{C},h}$ does not contain $[u] \cap \mathscr{C}$. ◀

The result does not hold for any class $\mathscr{C}$: if $\mathscr{C}$ is the class of constant functions, numbered in the obvious way, then having an oracle for $f \in \mathscr{C}$ is equivalent to having a $\mathscr{C}$-index of f. This class has the particular property that all the functions in $\mathscr{C}$ are isolated from each other and as we now show this is the only obstruction to generalizing Propositions 4 and 5. We recall that a function f is not isolated in $\mathscr{C}$ if for each $p \in \mathbb{N}$ there exists some $g \neq f$ in $\mathscr{C}$ such that $g\restriction_p = f\restriction_p$.

▶ **Proposition 6.** If $\mathscr{C}$ contains a function f that is not isolated in $\mathscr{C}$ then there is a computable order h such that f belongs to $A_{\mathscr{C},h}$ but not to its interior, therefore $A_{\mathscr{C},h}$ is not semidecidable when the input function is given as oracle.

Hence having an index for $f \in \mathscr{C}$ usually gives more information than having an access to f via an oracle, as it enables to decide more properties of f. What is the additional information? Having an index for f obviously bounds $K_{\mathscr{C}}(f)$, and we now show that in a sense this is the only additional information.

First observe that the proof of Proposition 3 actually shows that $A_{\mathscr{C},h}$ remains decidable if one is given f via an oracle *together with an upper bound on* $K_{\mathscr{C}}(f)$. This is the case of every decidable, and even semidecidable property.

▶ **Proposition 7.** Let $A \subseteq \mathscr{C}$ be such that the problem $f \in A$ is semidecidable given a $\mathscr{C}$-index of f. Then the problem $f \in A$ is semidecidable given an access to f as oracle together with any upper bound on $K_{\mathscr{C}}(f)$.

Proof. Given f and $k \geq K_{\mathscr{C}}(f)$, one can progressively reject all numbers $i \leq k$ such that $f_i \neq f$. In parallel one can progressively accept all numbers $i \leq k$ that are accepted by the semidecision procedure for A. Wait for a stage when every number $i \leq k$ is accepted or rejected. If this happens then accept f. ◀

If one defines $\mathscr{C}_k = \{f : K_{\mathscr{C}}(f) \leq k\} = \{f_i : i \leq k\}$ then Proposition 7 implies the existence of uniformly effective open sets $U_k \subseteq \mathbb{N}^{\mathbb{N}}$ such that $A \cap \mathscr{C}_k = U_k \cap \mathscr{C}_k$ for all $k \in \mathbb{N}$. Indeed, U_k is defined as the union of finite prefixes of oracles f read and accepted by the machine semideciding A, given k as upper bound on $K_{\mathscr{C}}(f)$.

It was proved in [6] that such a result also holds for the class of total computable functions and much more general classes of computable objects. The proof given here in the case of a subrecursive class $\mathscr{C}$ is much easier because we only deal with total programs (every f_i is total, so one can always recognize whether $f_i \neq f$).

2.2 The main result

We can now state our main result: the cylinders and the sets of anticomplex functions are the basic decidable properties, from which all decidable and semidecidable properties can be obtained.

▶ **Theorem 8.** *Let $\mathscr{C}$ be a subrecursive class and $A \subseteq \mathscr{C}$. The following conditions are equivalent:*

1. *The property $f \in A$ is semidecidable given a $\mathscr{C}$-index of f,*
2. *A is an effective union of sets of the form $[v] \cap A_{\mathscr{C},h}$, i.e.*

$$A = \mathscr{C} \cap \bigcup_n ([v_i] \cap A_{\mathscr{C},h_i})$$

for some computable sequences of finite words $v_i \in \mathbb{N}^*$ and orders $h_i : \mathbb{N} \to \mathbb{N}$.

Proof. We prove that 1. implies 2., the other direction being a direct consequence of Proposition 3. We slightly reformulate the property $A_{\mathscr{C},h}$, using the following sets. Recall the sets $\mathscr{C}_k = \{f_i : i \leq k\}$ defined after Proposition 7. For $k, n \in \mathbb{N}$ let

$$\mathscr{C}_k^n = \bigcup_{u \in \mathbb{N}^n : K_{\mathscr{C}}(u) \leq k} [u] = \bigcup_{f \in \mathscr{C}_k} [f\restriction_n] = \bigcup_{i \leq k} [f_i\restriction_n].$$

Observe that $f \in A_{\mathscr{C},h}$ if and only if $f \in \mathscr{C}_{h(n)}^n$ for all n.

Assume that condition 1. in the statement of the theorem holds. By Proposition 7 there exist uniformly effective open sets $U_k \subseteq \mathbb{N}^{\mathbb{N}}$ such that $A \cap \mathscr{C}_k = U_k \cap \mathscr{C}_k$. One can take $U_{k+1} \subseteq U_k$, replacing U_k with $U_k \cup U_{k+1} \cup \ldots$ if necessary. It follows that $A = \mathscr{C} \cap \bigcap_k U_k$.

Given $k \in \mathbb{N}$ and $v \in \mathbb{N}^*$ such that $[v] \subseteq U_{k+1}$, we now build a computable order h such that $[v] \cap \mathscr{C}_{k+1} \subseteq [v] \cap A_{\mathscr{C},h} \cap \mathscr{C} \subseteq A$. In order to obtain the announced families v_i and h_i to cover the whole set A, we will simply start from all possible $k \in \mathbb{N}$ and all v in the effective enumeration of U_{k+1}.

We now define h, by first constructing a kind of inverse of h. More precisely we define a computable increasing sequence n_i such that for all i,

$$[v] \cap \mathscr{C}_{k+1}^{n_1} \cap \ldots \cap \mathscr{C}_{k+i}^{n_i} \subseteq U_{k+i+1}.$$

The base case $i = 0$ is satisfied as $[v] \subseteq U_{k+1}$. Once $n_1, \ldots, n_i$ have been defined,

$$\begin{aligned}[v] \cap \mathscr{C}_{k+1}^{n_1} \cap \ldots \cap \mathscr{C}_{k+i}^{n_i} \cap \mathscr{C}_{k+i+1} &\subseteq U_{k+i+1} \cap \mathscr{C}_{k+i+1} \\ &\subseteq A \\ &\subseteq U_{k+i+2}.\end{aligned}$$

The left-hand side is a finite set. For each f in that set, there is $n \in \mathbb{N}$ such that $[f\restriction_n] \subseteq U_{k+i+2}$. As the set is finite there is a single n that works for each f in the finite set. As this finite set is computable, such a n can be computed. We then define $n_{i+1} > n_i$ such that

$$[v] \cap \mathscr{C}_{k+1}^{n_1} \cap \ldots \cap \mathscr{C}_{k+i}^{n_i} \cap \mathscr{C}_{k+i+1}^{n_{i+1}} \subseteq U_{k+i+2}.$$

We then have

$$[v] \cap \mathscr{C}_{k+1} \subseteq [v] \cap \bigcap_{i \geq 1} \mathscr{C}_{k+i}^{n_i} \cap \mathscr{C} \subseteq \bigcap_n U_n \cap \mathscr{C} = A. \tag{1}$$

Let h be the computable order defined by $h(n) = k + \min\{i \geq 1 : n \leq n_i\}$. We claim, as announced, that

$$[v] \cap \mathscr{C}_{k+1} \subseteq [v] \cap A_{\mathscr{C},h} \cap \mathscr{C} \subseteq A.$$

The first inequality is straightforward: $h(n) \geq k + 1$ for all n, so $\mathscr{C}_{k+1} \subseteq A_{\mathscr{C},h}$.

To prove the second inequality, observe that if $g \in A_{\mathscr{C},h}$ then for all i, $K_{\mathscr{C}}(g\restriction_{n_i}) \leq h(n_i) = k + i$, i.e. $g \in \bigcap_{i \geq 1} \mathscr{C}_{k+i}^{n_i}$, and then use (1). ◀

A set $A \subseteq \mathscr{C}$ is then decidable from $\mathscr{C}$-indices if and only if both A and $\mathscr{C} \setminus A$ can be expressed as effective unions of sets $[v] \cap A_{\mathscr{C},h}$.

A further question

Each effective numbering of a class $\mathscr{C}$ induces particular classes of decidable and semidecidable properties. What can be said about the properties that are decidable or semidecidable for *every* effective numbering of $\mathscr{C}$? The least we can say is that any property of total functions that is Markov decidable, i.e. decidable given an arbitrary index of the total function, is also decidable for every effective numbering of $\mathscr{C}$ (see next section for more information about Markov computability), and similarly for semidecidable properties. Does the converse also holds? In a sequel to this paper we will show that it does not: in reasonable subrecursive classes, there exists a property that cannot be semidecided from arbitrary indices, but is semidecidable in any effective total numbering of the class.

2.3 The whole class of total computable functions

By the Kreisel-Lacombe-Shœnfield/Ceitin theorem, the properties of total computable functions that are Markov decidable, i.e., decidable from indices coincide with the ones that are decidable from oracles, hence are generated by cylinders. However Friedberg showed that is it not the case for Markov semidecidable properties. Can we obtain a characterization of these properties as in the case of subrecursive classes?

We leave this problem open, but we show that the analog of Theorem 8 does not hold. We first introduce the analog of Definition 2. Here, φ_i is some Gödel numbering of the partial computable functions.

▶ **Definition 9.** If $v = (v_0, \dots, v_n) \in \mathbb{N}^*$ then its *complexity* is

$$K(v) = \min\{i : \varphi_i \text{ extends } v\}.$$

Let h be a computable order. We define the set A_h of *h-anticomplex* functions as

$$A_h = \{f : \mathbb{N} \to \mathbb{N} : \forall n, K(f\restriction_n) \leq h(n)\}.$$

Again, the property $f \in A_h$ is semi-decidable (but this time not decidable) from any index of the total computable function f, but it is not semidecidable if f is given by an oracle. Observe that in the definition of $K(v)$, one considers all *partial* computable functions extending v. A direct analog of Definition 1 would be to consider *total* functions only. However the resulting anticomplexity property would not be semidecidable.

We now prove that the sets of anticomplex functions do not generate all the semidecidable properties.

▶ **Theorem 10.** *There is a semidecidable property of total computable functions that does not contain any non-empty set* $[v] \cap A_h$.

Proof. Let $t(j,i)$ be a partial computable function such that if φ_j is total then $t(j,i)$ is defined for all i. Define the set

$$B = \bigcap_i B_i \text{ where } B_i = \bigcup_{j \leq i} [\varphi_j\restriction_{t(j,i)}]$$

where $[\varphi_j\restriction_{t(j,i)}]$ is empty if $t(j,i)$ is not defined or φ_j is not defined on the first $t(j,i)$ inputs. The property $f \in B$ is semidecidable from indices of f. Indeed, if φ_i is total then $\varphi_i \in B \iff \varphi_i \in B_0 \cap \ldots \cap B_{i-1}$, which is semidecidable.

We now take $t(j,i)$ to be the halting time of $\varphi_j(i)$ plus $i+1$. We prove that the corresponding set B does not contain any non-empty set $[v] \cap A_h$. Let h be some computable order. We want to build a function f in $A_h \setminus B$, i.e. in $A_h \setminus B_b$ for some $b \in \mathbb{N}$.

▶ **Lemma 11.** *Given* $a, b \in \mathbb{N}$ *one can compute* $m = m(a,b)$ *such that if* $[\varphi_a\restriction_m] \setminus B_b$ *is non-empty then it contains some* f *such that* $K(f) \leq h(m)$.

Proof. The idea is simply that for $a, b, m \in \mathbb{N}$, if $[\varphi_a\restriction_m] \setminus B_b$ is non-empty then it contains a function whose complexity can be controlled. Indeed, while such a function cannot be effectively found as the set B_b is only enumerable, it becomes possible if some extra bits of information about B_b are provided.

Consider an algorithm that on inputs a, b, M and $p \leq b+1$ tries to find a set $L \subseteq \{0, \dots, b\}$ of p elements such that for all $j \in L$, $\varphi_j\restriction_{t(j,b)}$ is defined, tests whether $[\varphi_a\restriction_M] \setminus \bigcup_{j \in L} [\varphi_j\restriction_{t(j,b)}]$ is non-empty and if it is so computes some total function f in that set (p is a guess about the number of cylinders in B_b).

The complexity of the output of the algorithm can be bounded in terms of the complexity of its inputs: there is a total computable function $m_0(a,b,e)$ such that if $M := \varphi_e(a,b)$ is defined and $p \leq b+1$ and the algorithm finds a function f, then $K(f) \leq m_0(a,b,e)$.

Now by the Recursion Theorem, there is e such that $\varphi_e(a,b) = \min\{m : h(m) \geq m_0(a,b,e)\}$. Let $m(a,b) = \varphi_e(a,b)$.

Applying the algorithm on inputs $a, b, m(a,b), p$ where p is the number of values of $j \leq b+1$ such that $\varphi_j\restriction_{t(j,b)}$ is defined (p is the "right guess") gives a function $f \in [\varphi_a\restriction_{m(a,b)}] \setminus B_b$ such that $K(f) \leq m_0(a,b,e) \leq h(m(a,b))$. ◀

We can make sure that $m(a,b) > b$ (in the proof above, take instead $\varphi_e(a,b) = \min\{m > b : h(m) \geq m_0(a,b,e)\}$).

▶ **Lemma 12.** *Let $a, b \in \mathbb{N}$ and $m = m(a,b)$. If $\varphi_a \in A_h$ and $[\varphi_a\restriction_m] \setminus B_b$ is non-empty then $[\varphi_a\restriction_m] \cap A_h \setminus B$ is non-empty.*

Proof. By Lemma 11 there exits $f \in [\varphi_a\restriction_m] \setminus B_b$ such that $K(f) \leq h(m)$. Of course, $f \notin B$ and we show that $f \in A_h$ i.e. that $K(f\restriction_i) \leq h(n)$ for all n.

For $n \leq m$, $K(f\restriction_n) = K(\varphi_a\restriction_n) \leq h(n)$ as $\varphi_a \in A_h$.

For $n \geq m$, $K(f\restriction_n) \leq K(f) \leq h(m) \leq h(n)$. ◀

▶ **Lemma 13.** *Let $v \in \mathbb{N}^*$ be such that $[v] \cap A_h \neq \emptyset$. There exist a, b such that $\varphi_a \in A_h$, $[\varphi_a\restriction_{m(a,b)}] \setminus B_b$ is non-empty and $[\varphi_a\restriction_{m(a,b)}] \subseteq [v]$.*

Proof. Define the computable function $b(a) = \min\{b \geq |v| : h(b) \geq a\}$. We now define a and will take $b := b(a)$.

Let $g \in [v] \cap A_h$. By the Recursion theorem, there is a such that

- For $i \neq b(a)$, $\varphi_a(i) = g(i)$,
- For $i = b(a)$, $\varphi_a(i)$ differs from each $\varphi_j(i)$ such that $j \leq b(a)$ and $\varphi_j(i)$ halts in at most $m(a, b(a))$ steps.

Let then $b = b(a)$.

▶ **Claim 14.** $\varphi_a \in A_h$, i.e. $K(\varphi_a\restriction_n) \leq h(n)$ for all n.

Indeed, for $n \leq b$ one has $K(\varphi_a\restriction_n) = K(g\restriction_n) \leq h(n)$ as $g \in A_h$. For $n > b$, $K(\varphi_a\restriction_n) \leq a \leq h(b) \leq h(n)$.

▶ **Claim 15.** Let $m = m(a,b)$. The set $[\varphi_a\restriction_m]$ is not contained in $B_b = \cup_{j \leq b}[\varphi_j\restriction_{t(j,b)}]$.

Indeed for each $j \leq b$:

- If $\varphi_j(b)$ halts in at most m steps then $\varphi_a(b) \neq \varphi_j(b)$ so $[\varphi_a\restriction_m]$ is disjoint from $[\varphi_j\restriction_{t(j,b)}]$ as both m and $t(j,b)$ are larger than b.
- If $\varphi_j(b)$ does not halt in at most m steps then $t(j,b)$ is either undefined or larger than m, so $\varphi_a\restriction_m$ does not contain $[\varphi_j\restriction_{t(j,b)}]$.

Finally, $[\varphi_a\restriction_{m(a,b)}] \subseteq [v]$ as $m(a,b) > b \geq |v|$ and $\varphi_a\restriction_{|v|} = g\restriction_{|v|} = v$. ◀

We can now conclude. If $[v] \cap A_h \neq \emptyset$ then applying Lemma 12 to a, b provided by Lemma 13 directly gives that $[v] \cap A_h \setminus B$ is non-empty, as it contains $[\varphi_a\restriction_{m(a,b)}] \cap A_h \setminus B$ which is non-empty. ◀

In other words, the complement of B is "so big" that its intersection with each A_h is dense in A_h.

We conjecture than there is no way of describing the semidecidable properties of total computable functions, using a parametrization by total computable functions. We say that a set W is extensional if when $\varphi_i = \varphi_j$ is total and $i \in W$, one has $j \in W$. An extensional c.e. set W represents the semidecidable property $\{\varphi_i : i \in W \text{ and } \varphi_i \text{ is total}\}$. Let $\mathrm{Tot} = \{i : \varphi_i \text{ is total}\}$.

▶ **Conjecture.** There is no computable function $h : \mathrm{Tot} \to \mathbb{N}$ such that

- For all $i \in \mathrm{Tot}$, $W_{h(i)}$ is extensional,
- Every semidecidable property is represented by some $W_{h(i)}$ with $i \in \mathrm{Tot}$.

3 Black-box or oracle?

In computer science one often makes the distinction between accessing a program via its *code*, or as a *black-box*. For instance, this distinction appears naturally when validating or evaluating the correctness of a program, either by proving that its code is correct, or testing its outputs on a bunch of inputs, without looking at its code.

As for programs of every day life, looking at the code usually gives much more information than looking at its outputs. What about the general case of arbitrary programs, where information can be obfuscated? What is the difference between reading the code of a program and running it as a black-box? Does one obtain the same information about the function it computes? What additional information does the code of a program contains, compared to a black-box containing the program?

The results presented here (e.g., Proposition 7) and in [6] may be seen as answers to these questions. However, strictly speaking our results involve *oracles* more than *black-boxes*, the difference being that a black-box hides an actual program while no assumption is put on an arbitrary oracle. Does it make a difference? Does a black-box containing a program computing a function f give more information about f than an arbitrary oracle for f? For instance, could the particular halting times of the program (measurable from outside the black-box) be exploited in some way?

In this section we present a few results that are partial answers to these questions.

We first prove a result suggesting that a black-box does not give more information than an arbitrary oracle.

3.1 Observing a Turing machine

Here we prove that if the program is a Turing machine and that we can observe its execution, without knowing the complete transition table, we do not have more information than from an oracle giving the outputs of the machine.

Observing the execution of the machine means that at each step one can see the configuration of the machine, i.e. the contents of the tapes, the positions of the heads and the internal state. However, one may never know the complete transition table and the number of states. Equivalently, the observer progressively obtains the content of the transition table (at least its reachable part), but if the table is incomplete he may never know it entirely.

More formally, let us assume that the set Q of states of a Turing machine is a subset of $\mathbb{N}$, but is not known by the observer. Σ is some known finite alphabet. Instead of having access to the transition table δ as a finite function from $Q \times \Sigma$ to $Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$, the

observer has access to δ as a partial function from $\mathbb{N} \times \Sigma \to \mathbb{N} \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$, defined on $Q \times \Sigma$ only. In particular no upper bound on the elements of Q is known.

▶ **Theorem 16.** *Let A be a set of total computable functions. The following are equivalent:*

1. *The problem $f \in A$ is semidecidable given an enumeration of a transition table of a Turing machine computing f,*
2. *The problem $f \in A$ is semidecidable given an oracle for f.*

Proof. The intuition is as follows. Assume that 1. holds. Given a total computable function f, there is a machine that outputs the same values as f on inputs $0, \ldots, n$ for some n, such that its transition table is accepted by the semidecision procedure and can be extended to the transition table of a machine computing g, for any g that coincides with f on $0, \ldots, n$. As a result, the cylinder $[f\restriction_n]$ is contained in A, which is open (and even effectively open).

Let $E \subseteq \mathbb{N}$ be a noncomputable c.e. set. The following claim is obvious.

▶ **Claim 17.** Given i, one can effectively build a machine M_i such that on input n, $M_i(n)$ halts on the same configuration as the initial one (in particular its input tape contains n), except that its state is q_1 if i is enumerated in E by stage n, q_0 otherwise. q_0 and q_1 are never reached before and there is no transition from these states.

Let N be a Turing machine with initial state q_0, all the other states being fresh (no common state with the machines M_i). Think of N as computing a total function f, but we do not need to assume that N is total.

▶ **Claim 18.** For each i one can effectively build a Turing machine, denoted $N \circ M_i$, such that $N \circ M_i(n)$ reaches q_1 if and only if i is enumerated in E by stage n, and $N \circ M_i$ computes the same function as N if $i \notin E$.

Proof. Given i, taking the union of the transition tables of N and M_i, with the initial state of M_i as initial state, one gets a Turing machine $N \circ M_i$ which first behaves as M_i, and then if $i \notin E$ behaves as N. ◀

We now prove that A is the intersection of an effective open set U with the class of total computable functions, which is equivalent to 2. in the statement of Theorem 16. For each machine N, look for $i \in E$ such that an enumeration of $N \circ M_i$ is accepted by the semidecision procedure. Compute n_0 such that i is enumerated in E at stage n_0. If N is defined on inputs $0, \ldots, n_0 - 1$, with output values $v_0, \ldots, v_{n_0-1}$ respectively then enumerate the cylinder $[v_0, \ldots, v_{n_0-1}]$ in U.

▶ **Claim 19.** A is contained in U.

Proof. Let $f \in A$ and N be a machine computing f. When $i \notin E$, $N \circ M_i$ computes f so any enumeration of its transition table is accepted by the semidecision procedure. As E is not computable, there must exist $i \in E$ such that an enumeration of the table of $N \circ M_i$ is also accepted. Let n_0 be such that i is enumerated in E at stage n_0. $N(n)$ is defined for every $n < n_0$ and outputs $f(n)$, so the cylinder enumerated in U is $[f\restriction_{n_0}]$. ◀

▶ **Claim 20.** Conversely, every computable function in U belongs to A.

Proof. Let $[v_0, \dots, v_{n_0-1}]$ be a cylinder enumerated in U. On inputs $n < n_0$, $N \circ M_i(n)$ outputs v_n, never reaching state q_1. On inputs $n \geq n_0$, $N \circ M_i$ ends in state q_1.

Let $g \in [v_0, \dots, v_{n_0-1}]$ and M_g be a machine computing g with initial state q_1, all the other states being fresh. Taking the union of the transition tables of $N \circ M_i$ and M_g, with the initial state of M_i as initial state, one gets a machine M'_g computing g. The enumeration of the table of $N \circ M_i$, accepted by the semidecision procedure, can be extended to an enumeration of the table of M'_g, which is then also accepted (the semidecision procedure halts before being able to distinguish between $N \circ M_i$ and M'_g). As a result, $g \in A$. ◀

The proof is complete: given f by an oracle, evaluate it successively on all inputs and look for a cylinder of U containing f. ◀

3.2 A difference between a black-box and an oracle

We now exhibit a difference between having a program in a black-box and an oracle.

Instead of deciding or semi-deciding properties, a usual task it to compute a function. In [6] it is proved that

▶ **Theorem 21** ([6]). *Let $F : X_c \to Y$ where X, Y are effective topological spaces and X_c is the set of computable elements of X. The following statements are equivalent:*

- *There is a Turing machine that computes $F(x)$ given any index of x as input,*
- *There is a Turing machine that computes $F(x)$ given any name for x and any $k \geq K(x)$ as input.*

We do not insist on the notion of effective topological space, which is essentially a topological space with a countable basis. The classes of partial computable functions or total computable functions are examples of effective topological spaces. A name for an element x is an infinite binary string encoding x in some canonical way, which we do not describe here (the interested reader may consult [13]).

The assumption about effective topological spaces is essential as there is a non-effective topological space Y for which the result fails, which is the class $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$ of open subsets of $\mathbb{N}^{\mathbb{N}}$ (which is not countably-based for the appropriate topology). Here we take for X the class $\mathcal{P}$ of partial computable functions.

▶ **Theorem 22** ([6]). *There is a functional $F : \mathcal{P} \to \mathcal{O}(\mathbb{N}^{\mathbb{N}})$ such that:*

- *There is a Turing machine that computes $F(\varphi)$ given any index of φ as input,*
- *There is <u>no</u> Turing machine that computes $F(\varphi)$ given any name for φ and any $k \geq K(\varphi)$ as input.*

Computing an element of $\mathcal{O}(\mathbb{N}^{\mathbb{N}})$, i.e. and open set $U \subseteq \mathbb{N}^{\mathbb{N}}$, consists in enumerating a list of finite words $v_i \in \mathbb{N}^*$ such that U is the corresponding union of cylinders $\bigcup_i [v_i]$. A name for a partial function φ is an infinite binary sequence such that $\varphi(m) = n$ if and only if the block $01^{\langle n,m \rangle}0$ appears in the sequence ($\langle ., . \rangle$ is a computable bijection between $\mathbb{N}^2$ and $\mathbb{N}$).

Contrasting with Theorem 22 we now show that

▶ **Theorem 23.** *Let $F : \mathcal{P} \to \mathcal{O}(\mathbb{N}^{\mathbb{N}})$. The following statements are equivalent:*

- *There is a Turing machine that computes $F(\varphi)$ given any index of φ as input,*
- *There is a Turing machine that computes $F(\varphi)$ given an access to a black-box containing a program computing φ, and any upper bound onthe size of the program.*

The difference with the previous theorem is that:

- Contrary to an oracle producing a name, a black-box contains an actual program, with its particular halting times,

- For a particular program p computing a function φ, an upper bound on the size of p is always an upper bound on $K(\varphi)$ (the size of the shortest program computing φ), but not the converse. In particular the theorem fails if an upper bound on $K(\varphi)$ rather than the size of $|p|$ is provided.

Proof idea. The argument is essentially the same as in the proof of Proposition 7. The idea is that the observation of the black-box can be seen as a total computable function that, given an input and a time, tells whether the program on that input halts in that time.

Assume that F is computable from indices. Let φ be given by a black-box and k an upper bound on the size of the program in the black-box.

At each stage we will have a finite list L of programs, such that the program in the black-box belongs to this list. At the beginning, L is the set of programs whose size is bounded by k. We enumerate the intersection of the open sets $F(\varphi_i)$ for all $i \in L$. From time to time we may remove an element of L that we know is not the program in the black-box. Each time we change L, we restart the enumeration of the intersection of the open sets $F(\varphi_i)$ for all i in the new list L (a larger open set). Eventually the list L will contain only programs computing the actual function φ, so we will enumerate the right open set.

We now explain how we progressively remove programs from L. For each program (in the list or in the black-box), each input n and each number t, we can decide whether the program halts in t steps on input n. If for some n and t a program is inconsistent with the black-box (one halts in t steps on input n but not the other), then the program can be rejected. If for some n the program and the black-box both halt on n giving different outputs, we can also reject the program.

Observe that we do not really need to have a precise measure of the halting time of the black-box: if we know that the actual halting time t of a program and the measured halting time $\tilde{t}$ are related by $|t - \tilde{t}| \leq 10$, or $t/2 \leq \tilde{t} \leq 2t$ for instance, then we only rejects programs that do not respect this gap w.r.t. the black-box. ◀

4 Two remarks on The Intensional Content of Rice's Theorem

In this paper we have investigated the properties of functions that are semidecidable, when the function is presented as a program computing it. Such a property can be alternatively seen as a c.e. set of programs that is *extensional*, in the sense that two equivalent programs – two programs computing the same function – are both in the set or both outside the set.

Asperti [1] investigates the case when extensionality is understood in a weaker sense, i.e. for a stronger notion of equivalence: two programs are equivalent if they compute the same function *and* have similar complexities (running time, or space, more generally any measure of complexity in Blum's sense). Such classes of programs are called Complexity Cliques.

It is proved in [1] that under certain assumptions on the measure of complexity (which should "behave well" w.r.t. the s-m-n function, composition and parallel computation),

▶ **Theorem 24** (Asperti [1]). *No Complexity Clique of total functions and containing programs with non constant complexity can be c.e.*

It is asked in [1] whether the assumption about non-constant complexity is needed.

We make the simple observation that it is indeed necessary, because the set of Turing machines with constant time complexity is a c.e. Complexity Clique. Indeed, given $c \in \mathbb{N}$, it is decidable whether a given Turing machine always halts in c steps, because one only has to evaluate it on inputs of size at most $c + 1$, during c steps. Hence it is semidecidable whether a Turing machine has constant time complexity, by trying every possible c.

We also observe that the assumptions about the measure of complexity cannot be dropped either, as the class of one-tape Turing machines that run in linear time is a c.e. Complexity Clique. Indeed, it was recently proved by Gajser [5] that for each $c \in \mathbb{N}$, whether a one-tape Turing machine halts in time cn on inputs of size n is decidable (his result is more general as it applies to a larger class of time bounds in $o(n \log n)$). It gives an indirect proof that one-tape Turing machines and their running time do not satisfy the assumptions of [1].

Acknowledgements. We thank the anonymous referees for their useful comments that helped improving this article, and for suggesting the question at the end of Section 2.2.

References

1. Andrea Asperti. The intensional content of rice's theorem. In George C. Necula and Philip Wadler, editors, *Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2008, San Francisco, California, USA, January 7-12, 2008*, pages 113–119. ACM, 2008. doi:10.1145/1328438.1328455.
2. G. S. Ceitin. Algorithmic operators in constructive metric spaces. *Trudy Matematiki Instituta Steklov*, 67:295–361, 1962. English translation: *American Mathematical Society Translations*, series 2, 64:1-80, 1967.
3. Johanna N. Y. Franklin, Noam Greenberg, Frank Stephan, and Guohua Wu. Anti-complex sets and reducibilities with tiny use. *J. Symb. Log.*, 78(4):1307–1327, 2013. doi:10.2178/jsl.7804170.
4. Richard M. Friedberg. Un contre-exemple relatif aux fonctionnelles récursives. *Comptes Rendus de l'Académie des Sciences*, 247:852–854, 1958.
5. David Gajser. Verifying time complexity of turing machines. *Theoretical Computer Science*, 600:86–97, 2015. doi:10.1016/j.tcs.2015.07.028.
6. Mathieu Hoyrup and Cristóbal Rojas. On the information carried by programs about the objects they compute. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 447–459. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.STACS.2015.447.
7. Dexter Kozen. Indexings of subrecursive classes. *Theoretical Computer Science*, 11(3):277–301, 1980. doi:http://dx.doi.org/10.1016/0304-3975(80)90017-1.
8. Georg Kreisel, Daniel Lacombe, and Joseph R. Schœnfield. Fonctionnelles récursivement définissables et fonctionnelles récursives. *Comptes Rendus de l'Académie des Sciences*, 245:399–402, 1957.
9. Ming Li and Paul M. B. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer-Verlag, Berlin, 1993.
10. Albert R. Meyer and Dennis M. Ritchie. The complexity of loop programs. In *Proceedings of the 1967 22Nd National Conference*, ACM'67, pages 465–469, New York, NY, USA, 1967. ACM. doi:10.1145/800196.806014.
11. Henry G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):pp. 358–366, 1953. URL: http://www.jstor.org/stable/1990888.
12. Norman Shapiro. Degrees of computability. *Transactions of the American Mathematical Society*, 82:281–299, 1956.
13. Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.

Polynomial Time Corresponds to Solutions of Polynomial Ordinary Differential Equations of Polynomial Length: The General Purpose Analog Computer and Computable Analysis Are Two Efficiently Equivalent Models of Computations

Olivier Bournez[1], Daniel S. Graça*[2], and Amaury Pouly[3]

1 **Ecole Polytechnique, LIX, Palaiseau Cedex, France**
2 **CEDMES/FCT, Universidade do Algarve, Faro, Portugal; and SQIG/Instituto de Telecomunicações, Lisbon, Portugal**
3 **Ecole Polytechnique, LIX, Palaiseau Cedex, France; and CEDMES/FCT, Universidade do Algarve, Faro, Portugal**

Abstract

The outcomes of this paper are twofold.

Implicit complexity. We provide an implicit characterization of polynomial time computation in terms of ordinary differential equations: we characterize the class P of languages computable in polynomial time in terms of differential equations with polynomial right-hand side.

This result gives a purely continuous (time and space) elegant and simple characterization of P. We believe it is the first time such classes are characterized using only ordinary differential equations. Our characterization extends to functions computable in polynomial time over the reals in the sense of computable analysis.

Our results may provide a new perspective on classical complexity, by giving a way to define complexity classes, like P, in a very simple way, without any reference to a notion of (discrete) machine. This may also provide ways to state classical questions about computational complexity via ordinary differential equations.

Continuous-Time Models of Computation. Our results can also be interpreted in terms of analog computers or analog model of computation: As a side effect, we get that the 1941 General Purpose Analog Computer (GPAC) of Claude Shannon is provably equivalent to Turing machines both at the computability and complexity level, a fact that has never been established before. This result provides arguments in favour of a generalised form of the Church-Turing Hypothesis, which states that any physically realistic (macroscopic) computer is equivalent to Turing machines both at a computability and at a computational complexity level.

1998 ACM Subject Classification F.1.1 Models of Computation. F.1.3 Complexity Measures and Classes. G.1.7 Ordinary Differential Equations

Keywords and phrases Analog Models of Computation, Continuous-Time Models of Computation, Computable Analysis, Implicit Complexity, Computational Complexity, Ordinary Differential Equations

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.109

* Daniel Graça was partially supported by *Fundação para a Ciência e a Tecnologia* and EU FEDER POCTI/POCI via SQIG – Instituto de Telecomunicações through the FCT project UID/EEA/50008/2013.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 109; pp. 109:1–109:15

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

1 Introduction

The outcomes of this paper are twofold, and are concerning a priori not closely related topics.

Implicit Complexity: Since the introduction of the P and NP complexity classes, much work has been done to build a well-developed complexity theory based on Turing Machines. In particular, classical computational complexity theory is based on limiting resources used by Turing machines, like time and space. Another approach is implicit computational complexity. The term "implicit" in "implicit computational complexity" can sometimes be understood in various ways, but a common point of these characterizations is that they provide (Turing or equivalent) machine-independent alternative definitions of classical complexity.

Implicit characterization theory has gained enormous interest in the last decade. This has led to many alternative characterizations of complexity classes using recursive functions, function algebras, rewriting systems, neural networks, lambda calculus and so on.

However, most of – if not all – these models or characterizations are essentially discrete: in particular they are based on underlying discrete time models working on objects which are essentially discrete such as words, terms, etc. that can be considered as being defined in a discrete space.

Models of computation working on a continuous space have also been considered: they include Blum Shub Smale machines [4], and in some sense Computable Analysis [40], or quantum computers [17] which usually feature discrete-time and continuous-space. Machine-independent characterizations of the corresponding complexity classes have also been devised: see e.g. [10, 24]. However, the resulting characterizations are still essentially discrete, since time is still considered to be discrete.

In this paper, we provide a purely analog machine-independent characterization of the P class. Our characterization relies only on a simple and natural class of ordinary differential equations: P is characterized using ordinary differential equations (ODEs) with polynomial right-hand side. This shows first that (classical) complexity theory can be presented in terms of ordinary differential equations problems. This opens the way to state classical questions, such as P vs NP, as questions about ordinary differential equations.

Analog Computers: Our results can also be interpreted in the context of analog models of computation and actually originate as a side effect from an attempt to understand continuous-time analog models of computation, and if they could solve some problem more efficiently than classical models. Refer to [39] for a very instructive historical account of the history of Analog computers. See also [29, 9] for other discussions.

Indeed, in 1941, Claude Shannon introduced in [38] the General Purpose Analog Computer (GPAC) model as a model for the Differential Analyzer [11], a mechanical programmable machine, on which he worked as an operator. The GPAC model was later refined in [35], [23]. Originally it was presented as a model based on circuits (see Figure 1), where several units performing basic operations (e.g. sums, integration) are interconnected (see Figure 2).

However, Shannon himself realized that functions computed by a GPAC are nothing more than solutions of a special class of polynomial differential equations. In particular it can be shown that a function is computed by Shannon's model if and only if it is a (component of the) solution of an ordinary differential equations (ODEs) with polynomial right-hand side [38], [23]. In this paper, we consider the refined version presented in [23].

We note that the original model of the GPAC presented in [38], [23] is not equivalent to Turing machine based models. However, the original GPAC model performs computations

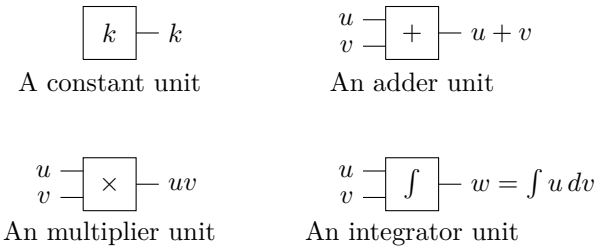


Figure 1 Circuit presentation of the GPAC: a circuit built from basic units.

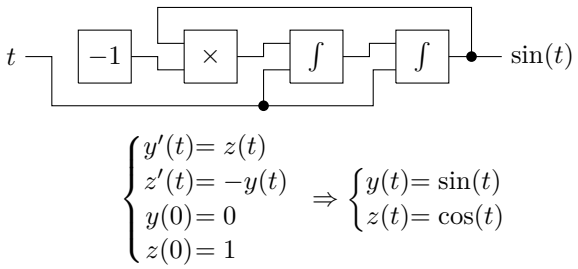


Figure 2 Example of GPAC circuit: computing sine and cosine with two variables.

in real-time: at time t the output is $f(t)$, which different from the notion used by Turing machines. In [19] a new notion of computation for the GPAC, which uses "converging computations" as done by Turing machines was introduced and it was shown in [5],[6] that using this new notion of computation, the GPAC and computable analysis are two equivalent models of computation at a computability level.

In that sense, our paper extends this latter result and proves that the GPAC and computable analysis are two equivalent models of computation, both at the computability and at the complexity level. We also provide as a side effect a robust way to measure time in the GPAC, or more generally in computations performed by ordinary differential equations: basically, by considering the length of the curve.

This paper is organized as follows. Section 2 gives our main definitions and results. Section 3 discusses the related work and consequences of our results. Section 4 gives a very high-level overview of the proof. It also contains more definitions and results so that the reader can understand the big steps of the proof.

2 Our Results

We consider the following class of differential equations:

$$y(0) = y_0 \qquad y'(t) = p(y(t)) \tag{1}$$

where $y : I \to \mathbb{R}^d$ for some interval $I \subset \mathbb{R}$ and where p is a vector of polynomials. Such systems are sometimes called PIVP, for polynomial initial value problems [21]. Observe that there is always a unique solution to the PIVP, which is analytic, defined on a maximum interval of life I containing y_0, which we refer to as "the solution".

Our crucial and key idea is that, when using PIVPs to compute a function f, the complexity should be measured as the length of the solution curve of the PIVP computing the function f. We recall that the length of a curve $y \in C^1(I, \mathbb{R}^n)$ defined over some interval $I = [a, b]$ is given by $\text{len}_y(a, b) = \int_I \|y'(t)\|\, dt$, where $\|y\|$ refers to the infinite norm of y.

We assume the reader familiar with the notion of polynomial time computable function $f : [a, b] \to \mathbb{R}$ (see [40] for an introduction to computable analysis). We take $\mathbb{R}_+ = [0, +\infty[$ and denote by $\mathbb{R}_P$ the set of polynomial time computable reals. For any vector y, $y_{i\dots j}$ refers to the vector $(y_i, y_{i+1}, \dots, y_j)$. For any sets X and Z, $f :\subseteq X \to Z$ refers to any function $f : Y \to Z$ where $Y \subseteq X$ and $\operatorname{dom} f$ refers to the domain of definition of f.

▶ Remark (The space $\mathbb{K}$ of the coefficients). In this paper, the coefficients of all considered polynomials will belong to $\mathbb{K}$. Formally, $\mathbb{K}$ needs to a be *generable field*, as introduced in [33]. However, without a significant loss of generality, the reader can consider that $\mathbb{K} = \mathbb{R}_P$ which is the set of polynomial time computable real numbers. All the reader needs to know about $\mathbb{K}$ is that it is a field and it is stable by generable functions (introduced in Section 4.2), meaning that if $\alpha \in \mathbb{K}$ and f is generable then $f(\alpha) \in \mathbb{K}$. It is shown in [33] that there exists a small generable field $\mathbb{R}_G$ lying somewhere between $\mathbb{Q}$ and $\mathbb{R}_P$, with expected strict inequality on both sides.

Our main results (the class AP is defined in Definition 3, and the notion of language recognized by a continuous system is given in Definition 4) are the following. Let us recall that $\mathrm{P}(\mathbb{R})$ is the class of polynomial time computable real functions, as defined in [27].

▶ **Theorem 1** (An implicit characterization of $\mathrm{P}(\mathbb{R})$). *Let $a, b \in \mathbb{R}_P$. A function $f : [a, b] \to \mathbb{R}$ is computable in polynomial time iff its belongs to the class* AP*.*

▶ **Theorem 2** (An implicit characterization of P). *A decision problem (language) $\mathcal{L}$ belongs to class* P *if and only if it is analog-recognizable.*

▶ **Definition 3** (Complexity Class AP). We say that $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$ is in AP if and only if there exists a vector p of polynomials with $d \geqslant m$ variables and a vector q of polynomials with n variables, both with coefficients in $\mathbb{K}$, and a bivariate polynomial Ω such that for any $x \in \operatorname{dom} f$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = q(x)$ and $y'(t) = p(y(t))$ ▶ y satisfies a PIVP
- for all $\mu \in \mathbb{R}_+$, if $\mathrm{len}_y(0, t) \geqslant \Omega(\|x\|, \mu)$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ ▶ $y_{1..m}$ converges to $f(x)$
- $\mathrm{len}_y(0, t) \geqslant t$ ▶ technical condition: the length grows at least linearly with time[1]

Intuitively, a function f belongs to AP if there is a PIVP that approximates f with a polynomial length to reach a given level of approximation.

In definition 3, the PIVP was given its input x as part of the initial condition: this is very natural because x was a real number. In the following, we will characterize languages with differential equations. Since a language is made up of words, we need to discuss how to represent (encode) a word with a real number. We fix a finite alphabet $\Gamma = \{0, .., k-2\}$ and define the encoding[2] $\psi(w) = \left(\sum_{i=1}^{|w|} w_i k^{-i}, |w|\right)$ for a word $w = w_1 w_2 \dots w_{|w|}$.

▶ **Definition 4** (Analog recognizability). A language $\mathcal{L} \subseteq \Gamma^*$ is called *analog-recognizable* if there exists a vector q of bivariate polynomials and a vector p of polynomials with d variables,

[1] This is a technical condition required for the proof. This can be weakened, for example to $\left\|y'(t)\right\| = \|p(y(t))\| \geqslant \frac{1}{\mathrm{poly}(t)}$. The technical issue is that if the speed of the system becomes extremely small, it might take an exponential time to reach a polynomial length, and we want to avoid such "unatural" cases. This is satisfied by all examples of computations we know [39].

[2] Other encodings may be used, however, two crucial properties are necessary: (i) $\psi(w)$ must provide a way to recover the length of the word, (ii) $\|\psi(w)\| \approx \mathrm{poly}(|w|)$ in other words, the norm of the encoding is roughly the size of the word.

both with coefficients in $\mathbb{K}$, and a polynomial $\Omega : \mathbb{R}_+ \to \mathbb{R}_+$, such that for all $w \in \Gamma^*$ there is a (unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ such that for all $t \in \mathbb{R}_+$:

- $y(0) = q(\psi(w))$ and $y'(t) = p(y(t))$ ▶ y satisfies a differential equation
- if $|y_1(t)| \geqslant 1$ then $|y_1(u)| \geqslant 1$ for all $u \geqslant t$ ▶ the decision is stable
- if $w \in \mathcal{L}$ (resp. $\notin \mathcal{L}$) and $\text{len}_y(0,t) \geqslant \Omega(|w|)$ then $y_1(t) \geqslant 1$ (resp. $\leqslant -1$) ▶ decision
- $\text{len}_y(0,t) \geqslant t$ ▶ technical condition

Intuitively this definition says that a language is analog-recognizable if there is a PIVP such that, if the initial condition is set to be (the encoding of) some word $w \in \Gamma^*$, then by using a portion of *polynomial length* of the curve, we are able to tell if this word should be accepted or rejected, by watching to which region of the space the trajectory will go: the value of y_1 determines if the word has been accepted or not, or if the computation is still in progress.

3 Discussion

Extensions. Our characterizations of the polynomial time can easily be extended to characterizations of deterministic complexity classes above polynomial time. For example, EXPTIME can be shown to correspond to the case where polynomial Ω is replaced by some exponential function.

▶ **Theorem 5.** *Let a and b in $\mathbb{R}_P$. A function $f : [a,b] \to \mathbb{R}$ is computable in exponential time iff its belongs to the class $f \in$ AEXP.*

▶ **Definition 6** (Definition of the complexity class AEXP for continuous systems)**.** We say that $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$ is in AEXP if and only if there exists a vector p of polynomial functions with d variables, a vector q of polynomial with n variables, both with coefficients in $\mathbb{K}$, an exponential function $\Omega : \mathbb{R}_+^2 \to \mathbb{R}_+$ such that for any $x \in \text{dom}\, f$, there exists (a unique) $y : \mathbb{R}_+ \to \mathbb{R}^d$ satisfying for all $t \in \mathbb{R}_+$:

- $y(0) = q(x)$ and $y'(t) = p(y(t))$ for all $t \geqslant 0$ ▶ y satisfies a PIVP
- for any $\mu \in \mathbb{R}_+$, if $\text{len}_y(0,t) \geqslant \Omega(\|x\|, \mu)$ then $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ ▶ $y_{1..m}$ converges
- $\|y'(t)\| \geqslant 1$ ▶ technical condition: The length grows at least linearly with time[3]

Applications to computational complexity. We believe these characterizations to really open a new perspective on classical complexity, as we indeed provide a natural definition (through previous definitions) of P for decision problems and of polynomial time for functions over the reals using analysis only i.e. ordinary differential equations and polynomials, no need to talk about any (discrete) machinery like Turing machines. This may open ways to characterize other complexity classes like NP or PSPACE. In the current settings of course NP can be viewed as an existential quantification over our definition, but we are obviously talking about "natural" characterizations, not involving unnatural quantifiers (for e.g. a concept of analysis like ordinary differential inclusions).

As a side effect, we also establish that solving ordinary differential equations with polynomial right-hand side leads to P- (or EXPTIME-)complete problems, when the length of the solution curve is taken into account. In an less formal way, this is stating that ordinary

[3] This is a technical condition required for the proof. This can be weakened, for example to $\|p(y(t))\| \geqslant \frac{1}{\text{poly}(t)}$. The technical issue is that the speed of the system becomes extremely small, it might take an exponential time to reach a polynomial length, and we want to avoid such "unatural" cases.

differential equations can be solved by following the solution curve (as most numerical analysis method do), but that for general (and even right-hand side polynomial) ODEs, no better method can work, unless some famous complexity questions do not hold. Note that our results only deal with ODEs with a polynomial right-hand side and that we do not know what happens for ODEs with analytic right-hand sides over unbounded domains. There are some results (see e.g. [31]) which show that ODEs with analytic right-hand sides can be computed locally in polynomial time. However these results do not apply to our setting since we need to compute the solution of ODEs over arbitrary large domains, and not only locally.

Applications to continuous-time analog models. PIVPs are known to correspond to functions that can be generated by the GPAC of Claude Shannon [38].

Defining a robust (time) complexity notion for continuous time systems is a well known open problem [9] with no generic solution provided to this day. In short, the difficulty is that the naive idea of using the time variable of the ODE as measure of "time complexity" is problematic, since time can be arbitrarily contracted in a continuous system due to the "Zeno phenomena" (e.g. by using functions like arctan which contract the whole real line into a bounded set). It follows that all computable languages can then be computed by a continuous system in time $O(1)$ (see e.g. [36], [37], [30], [7], [8], [1], [12], [15], [13], [14]).

With that respect, we solve this open problem by stating that the "time complexity" should be measured by the length of the solution curve of the ODE. Doing so, we get a robust notion of time complexity for PIVP systems. Indeed, the length is a geometric property of the curve and is thus "invariant" by rescaling. Notice that this is not sufficient to get robustness: the fact that we restrict to PIVP systems is crucial because more general ODEs are usually hard to simulate (e.g. see [26]). This explains why all previous attempts of a general complexity for general sytems failed in some sense [9]. Super-Turing "Zeno phenomena" can still happen with general ODEs, but not with PIVPs.

Applications to algorithms. We also believe that transferring the notion of time complexity to a simple consideration about length of curves allows for very elegant and nice proofs of polynomiality of many methods for solving continuous but also discrete problems. For example, the zero of a function f can easily be computed by considering the solution of $y' = -f(y)$ under reasonable hypotheses on f. More interestingly, this may also covers many interior-point methods or barrier methods where the problem can be transformed into the optimization of some continuous function (see e.g. [25, 16, 3, 28]).

Related work. We believe no purely continuous-time definition of P has ever been stated before. One direction of our characterization is based on a polynomial time algorithm (in the length of the curve) to solve PIVPs over unbounded time domains, such a result strengthens all existings results on the complexity of solving ODEs over unbounded time domains. In the converse direction, our proof requires a way to simulate a Turing machine using PIVP systems with a polynomial length, a task whose difficulty is discussed below, and still something that has never been done up to date.

Attempts to derive a complexity theory for continous-time systems include [18]. However, the theory developped there is not intended to cover generic dynamical systems but only specific systems that are related to Lyapunov theory for dynamical systems. The global minimizers of particular energy functions are supposed to give solutions of the problem. The structure of such energy functions leads to the introduction of problem classes U and NU, with the existence of complete problems for theses classes.

Another attempt is [2], also focussed on a very specific type of systems: dissipative flow models. The proposed theory is nice but non-generic. This theory has been used in several papers from the same authors to study a particular class of flow dynamics [3] for solving linear programming problems.

Both approaches are not at all intended to cover generic ODEs, and none of them is able to relate the obtained classes to classical classes from computational complexity.

Up to our knowledge, the most up to date survey about continuous time computation are [9, 29].

Relating computational complexity problems (like the P vs NP question) to problems of analysis has already been the motivation of series of works. In particular, Félix Costa and Jerzy Mycka have a series of work (see e.g. [32]) relating the P vs NP question to questions in the context of real and complex analysis. Their approach is very different: they do so at the price of a whole hierarchy of functions and operators over functions. In particular, they can use multiple times an operator which solves ordinary differential equations before defining an element of $DAnalog$ e $NAnalog$ (the counterparts of P and NP introduced in their paper), while in our case we do not need the multiple application of this kind of operator: we only need to use *one* application of such operator (i.e. we only need to solve one ordinary differential equations with polynomial right-hand side).

We also mention that Friedman and Ko (see [27]) proved that polynomial time computable functions are closed under maximization and integration if and only if some open problems of computational complexity (like P = NP for the maximization case) hold. The complexity of solving Lipschitz continuous ordinary differential equation has been proved to be polynomial-space complete by Kawamura [26].

All the results of this paper are fully developped in the PhD thesis of Amaury Pouly [33].

4 Overview of the proof

To show our main results (Theorem 1 and Theorem 2), we need to show two implications: (i) if a function $f : [a, b] \to \mathbb{R}$ (resp. a language $\mathcal{L}$) is polynomial time computable, then it belongs to AP (resp. it is analog-recognizable) and (ii) if a function $f : [a, b] \to \mathbb{R}$ belongs to AP (resp. a language $\mathcal{L}$ is analog-recognizable) then it is polynomial time computable (resp. belongs to P).

The second implication (ii) is proved by computing the solution of a PIVP system using some numerical algorithm. If a function $f : [a, b] \to \mathbb{R}$ in AP can be computed (up to some given accuracy) by following the solution curve of its associated ODE up to a reasonable (polynomial) amount of the length of the curve, the numerical simulation of its associated ODE will use a reasonable (polynomial) amount of resources to simulate this bounded portion of the solution curve. Hence the function f will be computed (up to some given accuracy, as usual in Computable Analysis) by a Turing machine in polynomial time. A similar idea can be used for showing the implication (ii) for P and analog-recognizable languages.

The idea sketched above gives the intuition of the proof but the usual ODE solving algorithms cannot be used here since (1) they are only guaranteed to compute the solution of an ODE with a given accuracy over a bounded time domain, but here we need to compute this solution over an unbounded time domain[4] which introduce further complications and (2)

[4] Note that while f has domain of definition $[a, b]$, from Definition 3 f is approximated by a PIVP whose solution is defined over the unbounded time domain $\mathbb{R}$

we need polynomial complexity in the length of the curve, which is not a classical measure of complexity.

The first implication (i) is proved by simulating Turing machines with PIVPs and by showing that these simulations can be performed by using a reasonable (polynomial) amount of resources (length of the solution curve) if the Turing machine runs in polynomial time.

Some simulation of Turing machines with PIVPs was already performed e.g. in [6], [22]. Basically one has to simulate the behavior of a Turing machine with a continuous system. This is problematic since Turing machines behave discretely (e.g. "if x happens then do A, otherwise do B") and one only has access to continuous (analytic) functions. This can be solved by *approximating* discontinuous functions with continuous functions to obtain an approximation of the transition function of the Turing machine. Then, by using special techniques, one can iterate the new (now continuous) transition function to simulate the step-by-step evolution of the Turing machine. Here we have one new difficult problem to tackle (not covered in previous papers like [6] and [22]) because we must ensure that everything can be done using only a reasonable (polynomial) amount of the length of the solution curve of the PIVP. In particular, this constraint rules out particularly simple techniques like integer encodings of the tape and error correction, as used in the previously mentioned papers.

At a high level, our proof relies on considerations about (polynomial length) *ODE programming*: we prove that it is possible to "program" with polynomial length ODE systems that keep some variable fixed, do assignement, iterate some functions, compute limits, etc. We use those basic operations and basic functions with PIVPs (e.g. $\min, \max$, continuous approximation of rounding, etc.) to create more complex functions and operations that simulate the transition function of a given Turing machine and its iterations. To be sure that the more complex functions still satisfy all the properties we want (e.g. that they belong to AP), we prove several closure properties: in particular, we prove very strong and elegant equivalent definitions of class AP.

For reasons of lack of space, we do not detail all these operators and functions, but we sketch the proof of a few properties and some key ideas of our techniques. We use the following notation: when p is a polynomial, Σp is the sum of the absolute values of its coefficients and $\deg(p)$ its degree. If p is a vector of polynomials, we extend those notions by taking the maximum for each component.

4.1 Polytime analog computability implies polytime computability

We start by sketching the proof of the "only if" direction of Theorem 2, and then of Theorem 1. Recall that a real function is polynomial time computable if given arbitrary approximations of the input, we can produce arbitrary approximations of the output in polynomial time. As it is customary, we proceed in two steps. We first show that the function has a polynomial modulus of continuity. This allows us to restrict the problem to rational inputs of controlled size.

▶ **Theorem 7** (Modulus of continuity). *If $f \in \mathrm{AP}$, then f admits a polynomial modulus of continuity: there exists a polynomial $\mho : \mathbb{R}_+^2 \to \mathbb{R}_+$ such that for all $x, y \in \operatorname{dom} f$ and $\mu \in \mathbb{R}_+$:*

$$\|x - y\| \leqslant e^{-\mho(\|x\|,\mu)} \quad \Rightarrow \quad \|f(x) - f(y)\| \leqslant e^{-\mu}.$$

We then show that the solution of a such a PIVP can be approximated in polynomial time. For this, will need the following theorem to get the complexity of numerically solving this PIVP. The idea of the proof is detailled below.

▶ **Theorem 8** (Complexity of Solving PIVP[34]). *If* [5] $y : \mathbb{R} \to \mathbb{R}^d$ *satisfies for all* $t \geqslant 0$.

$$y(0) = y_0 \qquad y'(t) = p(y(t)). \tag{2}$$

Then $y(t)$ *can be computed with precision* $2^{-\mu}$ *in time bounded by*

$$\mathrm{poly}(\deg(p), \mathrm{len}_y(0,t), \log \|y_0\|, \log \Sigma p, \mu)^d. \tag{3}$$

More precisely, there exists a Turing machine $\mathcal{M}$ *such that for any oracle* $\mathcal{O}$ *representing*[6] (y_0, p, t) *and any* $\mu \in \mathbb{N}$, $\left\|\mathcal{M}^{\mathcal{O}}(\mu) - \mathrm{PIVP}(y_0, p, t)\right\| \leqslant 2^{-\mu}$ *if* $y(t)$ *exists, and the number of steps of the machine is bounded by* (3) *for all such oracles.*

General Idea. Assume that $\mathcal{L}$ is analog-recognizable in the sense of Definition 4, using corresponding notations d, q, p, Ω. Let $w \in \Gamma^*$ and consider the following system: $y(0) = q(\psi(w))$, $y'(t) = p(y(t))$. We show that we can decide in time polynomial in $|w|$ whether $w \in \mathcal{L}$ or not. Theorem 8 can be used to conclude that we can compute $y(t) \pm e^{-\mu}$ in time polynomial in $\log \|q(\psi(w))\|, \mu$ and $\mathrm{len}_y(0,t)$. Recall that $\|\psi(w)\| = |w|$ and that the system is guaranteed to give an answer as soon as $\mathrm{len}_y(0,t) \geqslant \Omega(|w|)$. This means that it is enough to compute $y(t^*)$, where t^* satisfies $\mathrm{len}_y(0,t^*) \geqslant \Omega(|w|)$, with precision $1/2$ to distinguish between $y_1(t) \geqslant 1$ and $y_1(t) \leqslant -1$. Since $\mathrm{len}_y(0,t) \geqslant t$, thanks to the technical condition of the definition, we know that we can find a $t^* \leqslant \Omega(|w|)$. Note that $\mathrm{len}_y(0, \Omega(|w|))$ might not be polynomial in $|w|$ so we cannot simply compute $y(\Omega(|w|))$.

Fortunately, the proof of Theorem 8 provides us with an algorithm that solves the PIVP by making small time steps, and at each step the length cannot increase by more than a constant. This means that we can run algorithm to compute $y(\Omega(|w|))$ and stop it as soon as the length is greater than $\Omega(|w|)$. Let t^* be the time at which the algorithm stops. Then the running time of the algorithm will be polynomial in t^*, μ and $\mathrm{len}_y(0,t^*) \leqslant \Omega(|w|) + \mathcal{O}(1)$. Finally, thanks to the technical condition, $t^* \leqslant \mathrm{len}_y(0,t^*)$, this algorithm has running time polynomial in $|w|$.

The proof of Theorem 1 is established using the same principle based on Theorem 8, observing in addition that functions in AP can easily be approximated by considering only their value on rationals, since they have a polynomial modulus of continuity, as shown by the following theorem.

It thus appears that the true remaining difficulty lies in proving Theorem 8. An important point is that none of the classical methods for solving ordinary differential equations are polynomial time over unbounded time domains. Indeed, no method of fixed order r is polynomial in variable t over the whole domain $\mathbb{R}$.[7] For more information, we refer the reader to [34].

▶ Remark. Observe that the solution of the following PIVP $y_1' = y_1, y_2' = y_1 y_2, y_3' = y_2 y_3, \ldots, y_n' = y_{n-1} y_n$ is a tower of n exponentials. Its solution can be computed in polynomial time over any fixed compact $[a,b]$ [31]. However, the solution cannot be computed in polynomial time over $\mathbb{R}$, as just writing this value in binary cannot ever been done in polynomial time. Hence, the solution of a PIVP cannot be computed in polynomial time, over $\mathbb{R}$, in the general case. A key feature of our method is that we are searching methods polynomial in the length of the curve, which is not a classical framework.

[5] The existence of a solution y up to a given time is undecidable [20] so we have to assume existence.
[6] See [27] for more details. In short, the machine can ask arbitrary approximation of y_0, p and t to the oracle. The polynomial is represented by the finite list of coefficients.
[7] This is why most studies restricts to a compact domain.

4.2 Polytime computability implies polytime analog computability

The idea of the proof of the "if" directions is to simulate a Turing machine using a PIVP. But this is far from trivial since we need to do it with a polynomial length.

About generable functions. The following concept can be attributed to [38]: a function $f : \mathbb{R} \to \mathbb{R}$ is said to be a PIVP function if there exists a system of the form (1) with $f(t) = y_1(t)$ for all t, where y_1 denotes first component of the vector y defined in $\mathbb{R}^d$. We need in our proof to extend the concept to talk about (i) multivariable functions and (ii) the growth of these functions. The following class and closure properties can be seen as extensions of results from [21].

▶ **Definition 9** (Polynomially bounded generable function)**.** Let $d, e \in \mathbb{N}$, I be an open and connected subset of $\mathbb{R}^d$ and $f : I \to \mathbb{R}^e$. We say that $f \in$ GPVAL if and only if there exists a polynomial $\mathtt{sp} : \mathbb{R} \to \mathbb{R}_+$, $n \geqslant e$, a $n \times d$ matrix p consisting of polynomials with coefficients in $\mathbb{K}$, $x_0 \in \mathbb{K}^d$, $y_0 \in \mathbb{K}^n$ and $y : I \to \mathbb{R}^n$ satisfying for all $x \in I$:

- $y(x_0) = y_0$ and $J_y(x) = p(y(x))$ ▶ y satisfies a differential equation[8]
- $f(x) = y_{1..e}(x)$ ▶ f is a component of y
- $\|y(x)\| \leqslant \mathtt{sp}(\|x\|)$ ▶ y is polynomially bounded

▶ **Lemma 10** (Closure properties of GPVAL)**.** *Let $f :\subseteq \mathbb{R}^d \to \mathbb{R}^n \in$ GPVAL and $g :\subseteq \mathbb{R}^e \to \mathbb{R}^m \in$ GPVAL. Then $f + g$, $f - g$, fg and $f \circ g$ are in* GPVAL*.*

▶ **Lemma 11** (Generable functions are closed under ODE)**.** *Let $d \in \mathbb{N}$, $J \subseteq \mathbb{R}$ an interval, $f :\subseteq \mathbb{R}^d \to \mathbb{R}^d$ in* GPVAL*, $t_0 \in \mathbb{K} \cap J$ and $y_0 \in \mathbb{K}^d \cap \operatorname{dom} f$. Assume there exists $y : J \to \operatorname{dom} f$, and a polynomial $\overline{\mathtt{sp}} : \mathbb{R}_+ \to \mathbb{R}_+$ satisfying for all $t \in J$:*

$$y(t_0) = y_0 \qquad y'(t) = f(y(t)) \qquad \|y(t)\| \leqslant \overline{\mathtt{sp}}(t)$$

Then $y \in$ GPVAL *and it is unique.*

It follows that many polynomially bounded usual analytic[9] functions are in the class GPVAL. The inclusion GPVAL $\subset$ AP holds for functions whose domain is simple enough. However, the inclusion GPVAL $\subset$ AP is strict[10], since functions like the inverse of the Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ or Riemann's Zeta function $\zeta(x) = \sum_{k=0}^\infty \frac{1}{k^x}$ are not differentially algebraic [38] but belong to AP.

Robustness of AP. A very strong key argument of our proof is that the notion of computability given by Definition 3 is actually very robust and can be stated in many equivalent ways. A key point is that the definition can be weakened and strengthened. The following theorem shows that we weaken the definition without changing the class. Since it might not be obvious to the reader, we emphasize that this notion is a priori weaker (thus AP is a priori larger than AWP). Indeed, (i) the system accepts errors in the input (ii) the system does not even converge, but merely approximates the output, doing the best it can given the input error.

[8] J_y denotes the Jacobian matrix of y.
[9] Functions from GPVAL are necessarily analytic, as solutions of an analytic ODE are analytic.
[10] Even with functions with star domains with a vantage point.

▶ **Theorem 12** (Weak Computability). AP = AWP *where* AWP *corresponds to the class of functions* $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$ *such that there are some polynomials* $\Omega : \mathbb{R}_+^2 \to \mathbb{R}_+$ *and* $\Upsilon : \mathbb{R}_+^3 \to \mathbb{R}_+$, $d \in \mathbb{N}$, $p, q \in \mathrm{GPVAL}$, *such that for any* $x \in \mathrm{dom}\, f$ *and* $\mu \in \mathbb{R}_+$, *there exists (a unique)* $y : \mathbb{R}_+ \to \mathbb{R}^d$ *satisfying for all* $t \in \mathbb{R}_+$*:*

- $y(0) = q(x, \mu)$ *and* $y'(t) = p(y(t))$ ▶ *y satisfies a PIVP*
- *if* $t \geqslant \Omega(\|x\|, \mu)$ *then* $\|y_{1..m}(t) - f(x)\| \leqslant e^{-\mu}$ ▶ *$y_{1..m}$ approximates $f(x)$ within $e^{-\mu}$*
- $\|y(t)\| \leqslant \Upsilon(\|x\|, \mu, t)$ ▶ *$y(t)$ is polynomially bounded*

The proof of Theorem 12, however, is quite involved: first p and q can be equivalently assumed to be polynomials instead of functions in GPVAL above, from Lemma 11. Then AP $\subset$ AWP, follows from the fact that this is possible to rescale the system using the length of the curve as a new variable to make sure it does not grow faster than a polynomial time, we get what is needed. The other direction (AWP $\subset$ AP) is really harder: the first step is to transform a computation into a computation that tolerates small perturbations of the dynamics (AWP $\subset$ ARP). The second problem is to avoid that the system explodes for inputs not in the domain of the function, or for too big perturbation of the dynamics perturbations on inputs (ARP $\subset$ ASP). As a third step, we allow the system to have its inputs (input and precision) changed during the computation and the system has a maximum delay to react to these changes (ASP $\subset$ AXP). Finally, as a fourth step, we add a mechanism that feeds the system with the input and some precision. By continuously increasing the precision with time, we ensure that the system will converge when the input is stable. The result of these 4 steps is the following lemma, yielding a nice notion of online-computation (AXP $\subset$ AOP). Equality AP = AWP = AOP follows because time and length are related for polynomially bounded systems. The notion of online computability is an example of a priori strengthening of our notion of computation; yet it still corresponds to the same class of function. Intuitively, a function is online computable if, on any (long enough) time interval where the input is almost constant, the system converges (after some delay) the output of the function. Of course, the output will have some error that is related to the input error (due to the input not being exactly constant).

▶ **Lemma 13** (Online computability). AWP $\subset$ AOP*, where* AOP *corresponds to the class of functions* $f :\subseteq \mathbb{R}^n \to \mathbb{R}^m$ *such that for polynomials* $\Upsilon, \Omega, \Lambda : \mathbb{R}_+^2 \to \mathbb{R}_+$*, there exists* $\delta \geqslant 0$, $d \in \mathbb{N}$ *and* $p \in \mathbb{K}^d[\mathbb{R}^d \times \mathbb{R}^n]$ *and* $y_0 \in \mathbb{K}^d$ *such that for any* $x \in C^0(\mathbb{R}_+, \mathbb{R}^n)$*, there exists (a unique)* $y : \mathbb{R}_+ \to \mathbb{R}^d$ *satisfying for all* $t \in \mathbb{R}_+$*:*

- $y(0) = y_0$ *and* $y'(t) = p(y(t), x(t))$
- $\|y(t)\| \leqslant \Upsilon\big(\sup_{u \in [t-\delta, t]} \|x(u)\|, t\big)$
- *For any* $I = [a, b]$*, if there exists* $\bar{x} \in \mathrm{dom}\, f$ *and* $\bar{\mu} \geqslant 0$ *such that for all* $t \in I$, $\|x(t) - \bar{x}\| \leqslant e^{-\Lambda(\|\bar{x}\|, \bar{\mu})}$ *then* $\|y_{1..m}(u) - f(\bar{x})\| \leqslant e^{-\bar{\mu}}$ *whenever* $a + \Omega(\|\bar{x}\|, \bar{\mu}) \leqslant u \leqslant b$.

ODE Programming. With the closure properties of AP, programming with (polynomial length) ODE becomes a rather pleasant exercise, once the logic is understood. For example, simulating the assignement $y := g_\infty$ corresponds to dynamics $y(0) = y_0$, $y'(t) = \mathrm{reach}(\phi(t), y(t), g(t)) + E(t)$, for a fixed function reach $\in$ GPVAL, tolerating bounded error $E(t)$ on dynamics, and g fluctuating around g_∞. Other example: from a AP system computing f, just adding the corresponding AOP-equations for g, yields a PIVP computing $g \circ f$, by feeding output of the system computing f to the (online) input of g.

Turing machines. Consider a Turing machine $\mathcal{M} = (Q, \Sigma, b, \delta, q_0, q_\infty)$. A (instantaneous) configuration of M can be seen as a tuple $c = (x, \sigma, y, q)$ where $x \in \Sigma^*$ is the part of the

tape at left of the head, $y \in \Sigma^*$ is the part at the right, $\sigma \in \Sigma$ is the symbol under the head and $q \in Q$ the current state. Let $\mathcal{C}_{\mathcal{M}}$ be the set of configurations of $\mathcal{M}$, and $\mathcal{M}$ denotes the function mapping a configuration to its next configuration. In order to simulate a machine, we encode configurations with real numbers as follows. Recall that $\Gamma = \{0, 1, \ldots, k-2\}$ and let $\langle c \rangle = (0.x, \sigma, 0.y, q) \in \mathbb{Q} \times \Sigma \times \mathbb{Q} \times Q$ where $0.x = x_1 k^{-1} + x_2 k^{-2} + \cdots + x_{|x|} k^{-|x|} \in \mathbb{Q}$ with $x = x_1 x_2 \ldots x_{|x|}$.

▶ **Theorem 14** (Robust Real Step). *For any machine $\mathcal{M}$, there is some function $\langle \mathcal{M} \rangle \in$ AP such that for all $c \in \mathcal{C}_{\mathcal{M}}$, $\mu \in \mathbb{R}_+$ and $\bar{c} \in \mathbb{R}^4$, if $\|\langle c \rangle - \bar{c}\| \leqslant \frac{1}{2k^2} - e^{-\mu}$ then $\|\langle \mathcal{M} \rangle (\bar{c}, \mu) - \langle \mathcal{M}(c) \rangle\| \leqslant k \|\langle c \rangle - \bar{c}\|$.*

The difficulty of the proof is that one step of Turing machine with our encoding naturally involves computing the integer and fractional parts of a number. These operations are discontinuous and thus cannot be done in AP in full generality. This is solved by proving that a continuous and good enough "fractional part" like-function is in AP (and avoids constructions from [21]).

Iterating Functions. A key point for proving the main result is to show that it is possible to iterate a function using a PIVP under some specific hypotheses. The proof consists in building by ODE programming an ordinary differential equation using three variables y, z and w updating in a cycle to be repeated n times. At all time, y is an online component of the system computing $f(w)$. During the first stage of the cycle, w stays still and y converges to $f(w)$. During the second stage of the cycle, z copies y while w stays still. During the last stage, w copies z thus effectively computing one iterate. This computes all the iterates $f(x), f^{[2]}(x), \ldots$. The crucial point of this process is the error estimation, to guarantee that the system does not diverge, while keeping polynomial length. One of the key assumption to ensure this is for f to admit a specific kind of modulus of continuity. The other key assumption is an effective "openness" of the iteration domain.

▶ **Theorem 15** (Closure by iteration). *Let $I \subseteq \mathbb{R}^m$, $(f : I \to \mathbb{R}^m) \in$ AP, $\eta \in [0, 1/2[$ and assume that there exists a family of subsets $I_n \subseteq I$, for all $n \in \mathbb{N}$ and polynomials $\mho : \mathbb{R}_+ \to \mathbb{R}_+$ and $\Pi : \mathbb{R}_+^2 \to \mathbb{R}_+$ such that:*

- *for all $n \in \mathbb{N}$, $I_{n+1} \subseteq I_n$ and $f(I_{n+1}) \subseteq I_n$*
- *for all $x \in I_n$, $\left\|f^{[n]}(x)\right\| \leqslant \Pi(\|x\|, n)$*
- *for all $x \in I_n$, $y \in \mathbb{R}^m, \mu \in \mathbb{R}_+$, if $\|x - y\| \leqslant e^{-\mho(\|x\|)-\mu}$ then $y \in I$ and $\|f(x) - f(y)\| \leqslant e^{-\mu}$.*

Define $f_\eta^*(x, u) = f^{[n]}(x)$ for $x \in I_n$, $u \in [n - \eta, n + \eta]$ and $n \in \mathbb{N}$. Then $f_\eta^* \in$ AP.

The iteration of the (transition) functions given by Theorem 14 leads to a way to emulate any function computable in polynomial time.

At a high level, the "if" direction of Theorem 2 then follows. Indeed, decidability can be seen as the computability of some particular function with boolean output.

For the "if" direction of Theorem 1, there are further nontrivial obstacles to overcome. Given $x \in [a, b]$ and $\mu \in \mathbb{N}$, we want to compute an approximation of $f(x) \pm 2^{-\mu}$ and take the limit when $\mu \to \infty$. To compute f, we will use a polynomial time computable function g that computes f over rationals, and m a modulus of continuity. All we have to do is simulate g with input $\tilde{x}$ and μ, where $\tilde{x} = x \pm 2^{-m(\mu)}$ because we can only feed the machine with a finite input of course. The remaining nontrivial part of the proof is how to obtain the encoding of $\tilde{x}$ from x and μ. Indeed, the encoding is a discrete quantity whereas x is real number, so by a simple continuity argument, one can see that no such function can exist. The trick is

the following: from x and μ, we can compute two encodings ψ_1 and ψ_2 such that at least one of them is valid, and we know which one it is. So we are going to simulate g on both inputs and then select the result. Again, the select operation cannot be done continuously unless we agree to "mix" both results, i.e. we will compute $\alpha g(\psi_1) + (1-\alpha)g(\psi_2)$. The trick is to ensure that $\alpha = 1$ or 0 when only one encoding is valid, $\alpha \in]0,1[$ when both are valid (by "when" we mean with respect to x). This way, a mixing of both will ensure continuity but in fact when both encodings are valid, the outputs are nearly the same so we are still computing f. Obtaining such encodings ψ_1 and ψ_2 is also nontrivial and requires more uses of the closure by iteration property.

References

1. Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.
2. A. Ben-Hur, H. T. Siegelmann, and S. Fishman. A theory of complexity for continuous time systems. *J. Complexity*, 18(1):51–86, 2002.
3. Asa Ben-Hur, Joshua Feinberg, Shmuel Fishman, and Hava T. Siegelmann. Probabilistic analysis of a differential equation for linear programming. *Journal of Complexity*, 19(4):474–510, 2003. doi:S0885-064X(03)00032-3.
4. L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer, 1998.
5. O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. The General Purpose Analog Computer and Computable Analysis are two equivalent paradigms of analog computation. In J.-Y. Cai, S. B. Cooper, and A. Li, editors, *Theory and Applications of Models of Computation TAMC'06*, LNCS 3959, pages 631–643. Springer-Verlag, 2006.
6. O. Bournez, M. L. Campagnolo, D. S. Graça, and E. Hainry. Polynomial differential equations compute all real computable functions on computable compact intervals. *J. Complexity*, 23(3):317–335, 2007.
7. Olivier Bournez. Some bounds on the computational power of piecewise constant derivative systems (extended abstract). In *ICALP*, pages 143–153, 1997. doi:10.1007/3-540-63165-8_172.
8. Olivier Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoret. Comput. Sci.*, 210(1):21–71, 1999.
9. Olivier Bournez and Manuel L. Campagnolo. A survey on continuous time computations. In S.B. Cooper, B. Löwe, and A. Sorbi, editors, *New Computational Paradigms. Changing Conceptions of What is Computable*, pages 383–423. Springer-Verlag, New York, 2008.
10. Olivier Bournez, Felipe Cucker, Paulin Jacobé de Naurois, and Jean-Yves Marion. Implicit complexity over an arbitrary structure: Sequential and parallel polynomial time. *Journal of Logic and Computation*, 15(1):41–58, 2005.
11. V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.
12. C. S. Calude and B. Pavlov. Coins, quantum measurements, and Turing's barrier. *Quantum Information Processing*, 1(1-2):107–127, April 2002.
13. B. Jack Copeland. Even Turing machines can compute uncomputable functions. In C.S. Calude, J. Casti, and M.J. Dinneen, editors, *Unconventional Models of Computations*. Springer-Verlag, 1998.
14. B. Jack Copeland. Accelerating Turing machines. *Minds and Machines*, 12:281–301, 2002.

15 E. B. Davies. Building infinite machines. *The British Journal for the Philosophy of Science*, 52:671–682, 2001.

16 Leonid Faybusovich. Dynamical systems which solve optimization problems with linear constraints. *IMA Journal of Mathematical Control and Information*, 8:135–149, 1991.

17 R. P. Feynman. Simulating physics with computers. *Internat. J. Theoret. Phys.*, 21(6/7):467–488, 1982.

18 Marco Gori and Klaus Meer. A step towards a complexity theory for analog systems. *Mathematical Logic Quarterly*, 48(Suppl. 1):45–58, 2002.

19 D. S. Graça. Some recent developments on Shannon's General Purpose Analog Computer. *Math. Log. Quart.*, 50(4-5):473–485, 2004.

20 D. S. Graça, J. Buescu, and M. L. Campagnolo. Boundedness of the domain of definition is undecidable for polynomial ODEs. In R. Dillhage, T. Grubba, A. Sorbi, K. Weihrauch, and N. Zhong, editors, *4th International Conference on Computability and Complexity in Analysis (CCA 2007)*, volume 202 of *Electron. Notes Theor. Comput. Sci.*, pages 49–57. Elsevier, 2007.

21 D. S. Graça, J. Buescu, and M. L. Campagnolo. Computational bounds on polynomial differential equations. *Appl. Math. Comput.*, 215(4):1375–1385, 2009.

22 D. S. Graça, M. L. Campagnolo, and J. Buescu. Computability with polynomial differential equations. *Adv. Appl. Math.*, 40(3):330–349, 2008.

23 Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5):644–664, 2003.

24 Erich Grädel and Klaus Meer. Descriptive complexity theory over the real numbers. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on the Theory of Computing*, pages 315–324, Las Vegas, Nevada, 29May–1June 1995. ACM Press.

25 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

26 A. Kawamura. Lipschitz continuous ordinary differential equations are polynomial-space complete. *Computational Complexity*, 19(2):305–332, 2010.

27 Ker-I Ko. *Complexity Theory of Real Functions*. Progress in Theoretical Computer Science. Birkhaüser, Boston, 1991.

28 Masakazu Kojima, Nimrod Megiddo, Toshihito Noma, and Akiko Yoshise. *A unified approach to interior point algorithms for linear complementarity problems*, volume 538. Springer Science & Business Media, 1991.

29 Bruce J MacLennan. Analog computation. In *Encyclopedia of complexity and systems science*, pages 271–294. Springer, 2009.

30 Cristopher Moore. Recursion theory on the reals and continuous-time computation. *Theoretical Computer Science*, 162(1):23–44, 5 August 1996.

31 N. Müller and B. Moiske. Solving initial value problems in polynomial time. In *Proc. 22 JAIIO – PANEL 1993, Part 2*, pages 283–293, 1993.

32 J. Mycka and J. F. Costa. The $p \neq np$ conjecture in the context of real and complex analysis. *J. Complexity*, 22(2):287–303, 2006.

33 Amaury Pouly. *Continuous models of computation: from computability to complexity*. PhD thesis, Ecole Polytechnique and Unidersidade Do Algarve, Defended on July 6, 2015. 2015. https://pastel.archives-ouvertes.fr/tel-01223284.

34 Amaury Pouly and Daniel S. Graça. Computational complexity of solving polynomial differential equations over unbounded domains. *Theor. Comput. Sci.*, 626:67–82, 2016. doi:10.1016/j.tcs.2016.02.002.

35 M. B. Pour-El. Abstract computability and its relations to the general purpose analog computer. *Trans. Amer. Math. Soc.*, 199:1–28, 1974.

36 Keijo Ruohonen. Undecidability of event detection for ODEs. *Journal of Information Processing and Cybernetics*, 29:101–113, 1993.

37 Keijo Ruohonen. Event detection for ODEs and nonrecursive hierarchies. In *Proceedings of the Colloquium in Honor of Arto Salomaa. Results and Trends in Theoretical Computer Science (Graz, Austria, June 10-11, 1994)*, volume 812 of *Lecture Notes in Computer Science*, pages 358–371. Springer-Verlag, Berlin, 1994.

38 C. E. Shannon. Mathematical theory of the differential analyser. *Journal of Mathematics and Physics MIT*, 20:337–354, 1941.

39 Bernd Ulmann. *Analog computing*. Walter de Gruyter, 2013.

40 K. Weihrauch. *Computable Analysis: an Introduction*. Springer, 2000.

Algorithmic Complexity for the Realization of an Effective Subshift By a Sofic*

Mathieu Sablik[1] and Michael Schraudner[2]

1 **Aix Marseille Université, CNRS, Marseille, France**
`mathieu.sablik@univ-amu.fr`
2 **CMI, Universidad de Chile, Santiago Centro, Chile**

Abstract

Realization of d-dimensional effective subshifts as projective sub-actions of $d + d'$-dimensional sofic subshifts for $d' \geq 1$ is now well known [6, 4, 2]. In this paper we are interested in qualitative aspects of this realization. We introduce a new topological conjugacy invariant for effective subshifts, the speed of convergence, in view to exhibit algorithmic properties of these subshifts in contrast to the usual framework that focuses on undecidable properties.

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases Subshift, computability, time complexity, space complexity, tilings

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.110

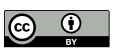
Introduction

A d-dimensional subshift is a set of colorings of $\mathbb{Z}^d$ by a finite set of colors in which a set of forbidden patterns never appear. The simplest class, called subshifts of finite type, corresponds at finite sets of forbidden patterns. In dimension 2, they are equivalent to the usual notion of tilings introduced by Wang [16]. Applying a block map on a subshift of finite type, one obtains a sofic subshift which can be characterized, in dimension 1, by a set of forbidden patterns accepted by a finite automaton [17].

For multidimensional subshifts, we can consider their stability according to another dynamical operation: projective subaction which consists of restricting the configurations of a subshift to a sublattice of $\mathbb{Z}^d$. The smallest class stable under this operation which contains the class of sofic shifts is the set of effective subshifts defined by a set of forbidden patterns enumerated by a Turing machine. A consequence of the main result of [6] states that every d-dimensional effective subshift can be obtained via projective subaction of a $d+2$-dimensional sofic. This result was improved in [4, 2] to hold for $d+1$-dimensional sofics.

These three classes evoked are stable by conjugacy and underline links between dynamical characterization and computability property of their set of forbidden patterns. Other classes are exhibited in [1], using forbidden patterns recursively enumerated by Turing machine with oracle. In this article, we introduce new conjugacy invariant classes which subdivide the class of effective subshift based on the speed of convergence of the realization via projective subaction. In contrast to the usual framework that focuses on undecidable properties and their position relatively to some hierarchies [7, 15, 9, 10], the approach proposed here emphasizes the algorithmic properties of subshifts using time and space complexity.

* This work was partially supported by the ANR project QuasiCool (ANR-12-JS02-011-01).



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 110; pp. 110:1–110:14

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

In [14], the authors characterize one-dimensional sofic subshifts obtained by a projective subaction of subshift of finite type. It appears a difference between certain types of sofic subshifts, according to whether their realization can be stable or unstable that is to say if a bounded strip around the central one is necessary to obtain the desired sofic subshift or whether there is no bounds which guarantee to the central row to be in the subshift. This approach is inspired by the notion of stable and unstable limit-set for cellular automata [12].

In this article, we would like to go beyond the dichotomy stable vs unstable realization and try to quantify this notion. We introduce the notion of speed of convergence of the realization of an effective subshift by projective subaction of a sofic. This is defined as the function which, for a given integer k, returns the width of the strip necessary to obtain the language of the effective subshift up to a word of size k in the central rows.

Given an effective subshift, we study the set of speeds of convergence which realizes it as projective subaction. Modulo an equivalence relation this set is invariant under conjugacy (Sections 1.3). In Section 2 we compare the general constructions of realization of an effective subshift given in [6, 2] and we propose a quicker construction if the effective subshift has a periodic point. Moreover we show that when the dimension of the sofic increase the convergence is quicker. These results give upper bounds for realization by sofic, but is also possible to obtain lower bounds (see Section 3). In Section 4 we present some examples of different classes which exhibit the optimality of the different previous results.

1 Definitions and first properties

1.1 Classes of subshifts

Subshifts. Let $\mathcal{A}$ be a finite alphabet, a *configuration* x is an element of $\mathcal{A}^{\mathbb{Z}^d}$. Let $\mathbb{U}$ be a finite subset of $\mathbb{Z}^d$, denote $x_{\mathbb{U}}$ the *restriction* of x to $\mathbb{U}$. A *d-dimensional pattern* of support $\mathbb{U}$ is an element $p \in \mathcal{A}^{\mathbb{U}}$. Denote by $\mathcal{A}^*$ the set of d-dimensional patterns and $p \in \mathcal{A}^{\mathbb{U}}$ *appears* in a configuration x, denoted by $p \sqsubset x$, if there exists $\mathbf{i} \in \mathbb{Z}^d$ such that $p = x_{\mathbf{i}+\mathbb{U}}$.

For the product topology, $\mathcal{A}^{\mathbb{Z}^d}$ is a compact metric space on which $\mathbb{Z}^d$ acts by translation via the shift map σ defined for all $\mathbf{i} \in \mathbb{Z}^d$ by $\sigma^{\mathbf{i}}(x)_{\mathbf{j}} = x_{\mathbf{i}+\mathbf{j}}$ for all $x \in \mathcal{A}^{\mathbb{Z}^d}$ and $\mathbf{j} \in \mathbb{Z}^d$. The $\mathbb{Z}^d$-dynamical system $(\mathcal{A}^{\mathbb{Z}^d}, \sigma)$ is called the *fullshift* and a *subshift* is a σ-invariant closed subset of $\mathcal{A}^{\mathbb{Z}^d}$. Let $\mathbf{T} \subset \mathcal{A}^{\mathbb{Z}^d}$ be a subshift, define $\mathcal{L}(\mathbf{T}) = \{p \in \mathcal{A}^* : \exists x \in \mathbf{T} \text{ such that } p \sqsubset x\}$ the *language of* $\mathbf{T}$ and $\mathcal{L}_n(\mathbf{T}) = \{p \in \mathcal{A}^{[0,n-1]^d} : p \in \mathcal{L}(\mathbf{T})\}$ the *square language of size* n.

Finite type condition. Let F be a set of patterns, define the *subshift of forbidden patterns* F by $\mathbf{T}_F = \left\{x \in \mathcal{A}^{\mathbb{Z}^d} : \forall p \in F,\, p \not\sqsubset x\right\}$. Every subshift can be defined in this way and this allows to define classes of subshifts according to the complexity of F. Let $\mathbf{T}$ be a subshift,

- $\mathbf{T}$ is a *subshift of finite type* if there exists a finite set of patterns F such that $\mathbf{T} = \mathbf{T}_F$;
- $\mathbf{T}$ is an *effective subshift* if there exists a recursively enumerable set of patterns F (that is to say enumerated by a Turing machine) such that $\mathbf{T} = \mathbf{T}_F$.

Factor. A *block map* is a continuous function $\pi : \mathcal{A}^{\mathbb{Z}^d} \to \mathcal{B}^{\mathbb{Z}^d}$ such that $\pi \circ \sigma^{\mathbf{i}} = \sigma^{\mathbf{i}} \circ \pi$ for all $\mathbf{i} \in \mathbb{Z}^d$. Equivalently, there exists a local function $\overline{\pi} : \mathcal{A}^{\mathbb{U}} \longrightarrow \mathcal{B}$ where $\mathbb{U} \subset \mathbb{Z}^d$ is a finite set called *neighborhood* such that $\pi(x)_{\mathbf{i}} = \overline{\pi}(x_{\mathbf{i}+\mathbb{U}})$ for all $x \in \mathcal{A}^{\mathbb{Z}^d}$ and $\mathbf{i} \in \mathbb{Z}^d$.

Let $\mathbf{T} \subset \mathcal{A}^{\mathbb{Z}^d}$ be a subshift and $\pi : \mathcal{A}^{\mathbb{Z}^d} \to \mathcal{B}^{\mathbb{Z}^d}$ a block map, then $\pi(\mathbf{T}) \subset \mathcal{B}^{\mathbb{Z}^d}$ is a subshift called *factor subshift* of $\mathbf{T}$ by π which is called the *factor map*. A subshift $\mathbf{T}$ is called *sofic* if there exists a subshift of finite type $\mathbf{T}_F$ and a factor map π such that $\mathbf{T} = \pi(\mathbf{T}_F)$. The factor map π can be considered letter to letter.

Two subshifts $\mathbf{T}$ and $\mathbf{T}'$ are *conjugate* if there exists a bijective factor map $\psi : \mathbf{T} \longrightarrow \mathbf{T}'$. The different classes of subshifts defined here (finite type, sofic and effective subshifts) are stable under conjugacy.

Projective subactions. Let $\mathbf{T} \subseteq \mathcal{A}^{\mathbb{Z}^d}$ be a subshift and $d' < d$, the *projective subdynamics* of $\mathbf{T}$ of dimension d' is the subshift $\mathbf{SA}_{d'}(\mathbf{T})$ where $\mathbf{SA}_{d'} : \mathcal{A}^{\mathbb{Z}^d} \longrightarrow \mathcal{A}^{\mathbb{Z}^{d'}}$ is defined by $\mathbf{SA}_{d'}(x) = x_{\mathbb{Z}^{d'} \times \{\mathbf{0}\}}$ for all $x \in \mathcal{A}^{\mathbb{Z}^d}$.

▶ **Theorem 1.** *[6, 2, 4] Let $\Sigma \subset \mathcal{A}^{\mathbb{Z}^d}$ be an effective subshift, then the $d+1$-dimensional subshift $\widetilde{\Sigma} = \{x \in \mathcal{A}^{\mathbb{Z}^{d+1}} : \exists y \in \Sigma \text{ such that } x_{\mathbb{Z}^d \times \{i\}} = y \text{ for all } i \in \mathbb{Z}\}$ is sofic.*

In particular there exists a subshift of finite type $\mathbf{T} \subset \mathcal{B}^{\mathbb{Z}^{d+1}}$ and a factor map $\pi : \mathcal{B}^{\mathbb{Z}^{d+1}} \to \mathcal{A}^{\mathbb{Z}^{d+1}}$, which can be considered letter to letter, such that $\mathbf{SA}_d(\pi(\mathbf{T})) = \Sigma$.

1.2 Speed of convergence

Approximation row. Let F be a finite set of d-dimensional forbidden patterns on $\mathcal{B}$ and $d' < d$. Define $\mathbb{B}_n = \{k_{d'+1}\mathbf{e_{d'+1}}, \dots, k_d\mathbf{e_d} : (k_{d'+1}, \dots, k_d) \in [-n, n]^{d-d'}\}$ where $\mathbf{e_1}, \dots, \mathbf{e_d}$ are the canonical vectors of $\mathbb{Z}^d$ and denote $\mathrm{Proj}_{\mathbf{i}} : \mathcal{B}^{\mathbb{B}_n} \longrightarrow \mathcal{B}$ the projection according to the coordinates $\mathbf{i} \in \mathbb{B}_n$.

One considers the *n-approximation row of* $\mathbf{T}_F \subset (\mathcal{B}^{\mathbb{B}_n})^{\mathbb{Z}^{d'}}$ the d'-dimensional subshift of finite type defined by the finite condition where no patterns of F appears in the row of width n. Formally, it is defined by:

$$\mathbf{T}_F^{n,d\to d'} = \left\{ x \in \left(\mathcal{B}^{\mathbb{B}_n}\right)^{\mathbb{Z}^{d'}} : \forall p \in F, p \not\sqsubset \left(\mathrm{Proj}_{\mathbf{j}}(x_{\mathbf{i}})\right)_{(\mathbf{i},\mathbf{j}) \in \mathbb{Z}^{d'} \times \mathbb{B}_n} \right\}.$$

Let $\pi : \mathcal{B}^{\mathbb{Z}^d} \to \mathcal{A}^{\mathbb{Z}^d}$ be a factor map. One has $\mathbf{SA}_{d'}(\pi(\mathbf{T}_F)) = \bigcap_{n\in\mathbb{N}} \mathbf{SA}_{d'}\left(\pi\left(\mathbf{T}_F^{n,d\to d'}\right)\right)$ where for n sufficiently large $\mathbf{SA}_{d'}\left(\pi\left(\mathbf{T}_F^{n,d\to d'}\right)\right)$ denote the central row of $\pi\left(\mathbf{T}_F^{n,d\to d'}\right)$.

Speed of convergence. By definition of $\mathbf{T}_F^{n,d\to d'}$, if $u \in \mathcal{L}\left(\mathbf{SA}_{d'}(\pi(\mathbf{T}_F))\right)$, then $u \in \mathcal{L}\left(\mathbf{SA}_{d'}\left(\pi\left(\mathbf{T}_F^{n,d\to d'}\right)\right)\right)$. We want to quantify the reciprocal, that is to say given a k, find the smallest n such that $u \notin \mathcal{L}_k(\mathbf{SA}_{d'}(\pi(\mathbf{T}_F))) \implies u \notin \mathcal{L}_k\left(\mathbf{SA}_{d'}\left(\pi(\mathbf{T}_F^{n,d\to d'})\right)\right)$. This allows to quantify when a word is forbidden by the local rules F in the approximation row. The *speed of convergence as sofic* of the cover $\mathbf{T}_F$ with the factor π is the following function:

$$\begin{array}{rccl} \varphi_{F,\pi,d\to d'} : & \mathbb{N} & \longrightarrow & \mathbb{N} \\ & k & \longmapsto & \min\{n \in \mathbb{N} : u \notin \mathcal{L}_k(\mathbf{SA}_{d'}(\pi(\mathbf{T}_F))) \\ & & & \qquad \implies u \notin \mathcal{L}_k\left(\mathbf{SA}_{d'}\left(\pi(\mathbf{T}_F^{n,d\to d'})\right)\right)\}. \end{array}$$

$\varphi_{F,\pi,d\to d'}(k)$ corresponds to the minimum size of the row to detect a forbidden pattern in the effective subshift realized as projective subaction of $\pi(\mathbf{T}_F)$.

▶ **Example 2.** Consider the following set of 2-dimensional forbidden patterns

$$F = \left\{ \begin{array}{ccc} \begin{smallmatrix} \bar{\alpha} \\ \beta\, a \end{smallmatrix}, & \begin{smallmatrix} \beta \\ a\, a \end{smallmatrix}, & \begin{smallmatrix} \\ a\, \$ \end{smallmatrix}, \\ \begin{smallmatrix} \bar{\alpha} \\ b\, \bar{\gamma} \end{smallmatrix}, & \begin{smallmatrix} \bar{\gamma} \\ b\, \bar{b} \end{smallmatrix}, & \begin{smallmatrix} \\ \$\, \bar{b} \end{smallmatrix} \end{array} \right. \quad \text{such that} \quad \left. \begin{array}{l} \bar{\alpha} \in \{ a, \bar{b} \}, \\ \beta \in \{ \$, \bar{b} \}, \\ \bar{\gamma} \in \{ \$, a \}. \end{array} \right\}.$$

$\mathbf{SA}_1(\mathbf{T}_F)$ is the subshift where $\{\$a^nb^m\$, \$a^nb^ma, ba^nb^m\$, ba^nb^ma : m \neq n\}$ are the forbidden patterns. The idea is that in a configuration of $\mathbf{T}_F$, if a line contains $\$a^nb^m\$$ with

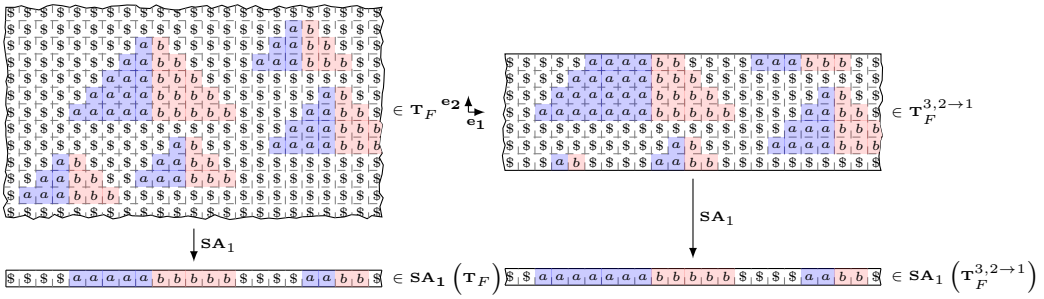


Figure 1 Application of $\mathbf{SA}_1$ under a configuration of $\mathbf{T}_F$ and $\mathbf{T}_F^{3,2\to1}$.

$n \neq m$ then the next line in the direction $\mathbf{e_2}$ contains $\$a^{n-1}b^{m-1}\$$ and recursively. Thus the pattern $a\$$ or $\$b$ appear and the configuration considered is excluded (see Figure 1).

In $\mathbf{T}_F^{n,2\to1}$ there is only n lines to detect a forbidden pattern so $\mathbf{SA}_1\left(\mathbf{T}_F^{n,2\to1}\right)$ is the subshift where the forbidden patterns are $\{\$a^pb^m\$, \$a^pb^ma, ba^pb^m\$, ba^pb^ma \ : \ m \neq p \text{ and } \max(p,m) \leq n\}$. We deduce that $\varphi_{F,\mathrm{Id},2\to1}(n) = \lfloor \frac{n}{2} \rfloor$. In Section 4 we will see that it is possible to obtain $\mathbf{SA}_1\left(\mathbf{T}_F\right)$ thanks to another sofic but with a better speed.

1.3 Subshift (φ, d)-realizable by sofic

A speed of convergence is in $\mathcal{F}$, the set of non-decreasing functions from $\mathbb{N}$ to $\mathbb{N}$. Denote

$$\mathcal{F}_{\Sigma,d\to d'}^{Sofic} = \left\{\varphi \in \mathcal{F} : \exists F \underset{\text{finite}}{\subset} \mathcal{B}^* \text{ and } \pi : \mathcal{B} \to \mathcal{A} \text{ with } \varphi = \varphi_{F,\pi,d\to d'} \text{ and } \mathbf{SA}_{d'}\left(\pi(\mathbf{T}_F)\right) = \Sigma\right\}.$$

By Theorem 1, $\mathcal{F}_{\Sigma,d\to d'}^{Sofic} \neq \emptyset$ if and only if Σ is effective. Using the fact that a sofic subshift can superpose different layers and delete them with the factor, it is easy to verify that $\mathcal{F}_{\Sigma,d\to d'}^{Sofic}$ is stable by min, max, multiplication by an integer, division by an integer, addition and multiplication. Moreover $\mathcal{F}_{\Sigma,d\to d'}^{Sofic} \subset \mathcal{F}_{\Sigma,d+1\to d'}^{Sofic}$.

Invariance of (φ, d)-realizable subshift under conjugacy. We need to introduce a preorder relation on $\mathcal{F}$. We say that $\varphi \prec \varphi'$ if there exists $r, M \in \mathbb{N}$ such that $\varphi(k) \leq M\varphi'(k+r)$ for all $k \in \mathbb{N}$. We say that $\varphi \sim \varphi'$ if $\varphi \prec \varphi'$ and $\varphi' \prec \varphi$. Multiplication by M comes from the fact that a given speed can be improved by division by an integer and addition by r allows stability by conjugacy.

A d'-dimensional subshift Σ is (φ, d)-*realizable by projective subaction of sofic* if there exist a finite set of d-dimensional forbidden patterns F and a factor π such that $\mathbf{SA}_{d'}\left(\pi(\mathbf{T}_F)\right) = \Sigma$ and $\varphi_{F,\pi,d\to d'} \prec \varphi$. The subshift Σ is *sharp* (φ, d)-*realizable* if moreover $\varphi \prec \varphi'$ for all $\varphi' \in \mathcal{F}_{\Sigma,d\to d'}^{Sofic}$.

▶ **Proposition 3.** *Let Σ and Σ' be two conjugated d'-dimensional subshfits. The subshift Σ is (φ, d)-realizable by projective subaction of a sofic if and only if it is the same for Σ'.*

Proof. Let $\psi : \Sigma \longrightarrow \Sigma'$ be the conjugation map of neighborhood $\mathbb{U} = [-r, r]^{d'}$. The local function can be extended in a function $\psi : \mathcal{A}^{\mathbb{Z}^d} \longrightarrow \mathcal{B}^{\mathbb{Z}^d}$ of neighborhood $\mathbb{U} = [-r, r]^{d'} \times \{\mathbf{0}\}$.

Let $\mathbf{T}_F$ be a subshift of finite type and π be a factor such that $\mathbf{SA}_{d'}\left(\pi(\mathbf{T}_F)\right) = \Sigma$, one

has $\mathbf{SA}_{d'}(\psi \circ \pi(\mathbf{T}_F)) = \Sigma'$. Let $u \in \mathcal{B}^{[0,k-1]^{d'}}$ and $\varphi = \varphi_{F,\pi,d\to d'}$, one has

$$\begin{aligned} u \notin \mathcal{L}(\Sigma') &\implies \left\{v \in \mathcal{A}^{[-r,k+r-1]^{d'}} : \psi(v) = u\right\} \not\subset \mathcal{L}(\Sigma) \\ &\implies \left\{v \in \mathcal{A}^{[-r,k+r-1]^{d'}} : \psi(v) = u\right\} \not\subset \mathcal{L}\left(\mathbf{SA}_{d'}\left(\pi\left(\mathbf{T}_F^{\varphi(k+2r),d\to d'}\right)\right)\right) \\ &\implies u \notin \mathcal{L}\left(\mathbf{SA}_{d'}\left(\psi \circ \pi\left(\mathbf{T}_F^{\varphi(k+2r),d\to d'}\right)\right)\right) \end{aligned}$$

Thus $\varphi_{F,\psi\circ\pi,d\to d'}(k) \leq \varphi_{F,\pi,d\to d'}(k+2r)$, the reciprocal is obtained using ψ^{-1}. ◀

2 Speed of convergence in general constructions

2.1 Notion of Turing machines

A *k-tapes Turing machine* $\mathcal{M} = (k, Q, \Gamma, \#, q_0, \delta, Q_F)$ is defined by:

- Γ a finite alphabet, with a blank symbol $\# \in \Gamma$. Initially, k infinite memory tapes represented as an element of $(\Gamma^k)^{\mathbb{Z}}$, are filled with $\#$, except for a finite prefix on the first tape (the input), and a computing head is located on the first letter of the tape;
- Q the finite set of states of the head and $q_0 \in Q$ is the initial state;
- $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{\leftarrow, \cdot, \rightarrow\}^k$ the transition function. Given the state of the head and the letter associated, it reads on the tape, depending on its position, the head can change state, replace the letter and move by one cell at most.
- $Q_F \subset Q$ the set of final states, when a final state is reached, the computation stops and the output is the value currently written on the tape.

Turing machines are a very robust model of computation, there exist several variants in the literature which are equivalent from a decidability point of view. Nevertheless these modifications on the definition are not without effects on the time and space complexities (time unit is one application of the transition function, space unit is one cell of the tape). To detect forbidden patterns in the projective subaction, one of the fundamental construction is the use of laical rules to encode Turing machine computations. In this article we choose to use the basic version of $\mathcal{M}$ but the reader should have in mind that it is possible to improve time and space complexities, using by instance these non-exhaustive acceleration techniques:

- **Compare-Copy:** compare or copy instantaneously a word between two markers between two tapes;
- **Transfer head:** transfer instantaneously the head to another cell of the tape marked by a special symbol;
- **Fill:** fill instantaneously a part of a tape with a periodic pattern.

Let F be a recursively enumerable set of forbidden patterns, then the complementary of $\mathcal{L}(\mathbf{T}_F)$ in $\mathcal{A}^*$, denoted $\mathcal{L}(\mathbf{T}_F)^c$ is also recursively enumerable. Consider $\mathcal{M}_{\mathcal{L}(\mathbf{T}_F)^c}$ be a Turing machine which enumerates $\mathcal{L}(\mathbf{T}_F)^c$, denote

- $\mathrm{Dtime}_{\mathcal{M}_{\mathcal{L}(\mathbf{T}_F)^c}}(k)$ the maximal time needed by the Turing machine $\mathcal{M}_{\mathcal{L}(\mathbf{T}_F)^c}$ to know if a pattern of size k is not in the language of $\mathbf{T}_F$;
- $\mathrm{Dspace}_{\mathcal{M}_{\mathcal{L}(\mathbf{T}_F)^c}}(k)$ the maximal space needed by the Turing machine $\mathcal{M}_{\mathcal{L}(\mathbf{T}_F)^c}$ to know if a pattern of size k is not in the language of $\mathbf{T}_F$ (only the space necessary for the computation is taken in consideration and the input is considered in an auxiliary tape).

▶ Remark. $\mathrm{Dtime}_{\mathcal{M}_{\mathcal{L}(\mathbf{T}_F)^c}}$ and $\mathrm{Dspace}_{\mathcal{M}_{\mathcal{L}(\mathbf{T}_F)^c}}$ are not computable if $\mathcal{L}(\mathbf{T}_F)^c$ is not recursive.

Let F be a set of patterns and $\mathcal{M}_F$ a Turing machine, called *enumerative Turing machine of* F, with the following behavior: it starts on the empty tape and successively writes the patterns of F on its tape. A set of finite patterns F *forbids the pattern* w if $w \notin \mathcal{L}(\mathbf{T}_F)$. Let $\mathcal{M}_F$ be an enumerative Turing machine of F, denote $\mathrm{Dtime}^{\mathrm{enu}}_{\mathcal{M}_F}(k)$ (resp. $\mathrm{Dspace}^{\mathrm{enu}}_{\mathcal{M}_F}(k)$) the smallest time (resp. the smallest space) taken by the Turing machine $\mathcal{M}_F$ such that the subset $F_{\mathrm{Dtime}^{\mathrm{enu}}_{\mathcal{M}_F}(k)}$ of F (resp. $F_{\mathrm{Dspace}^{\mathrm{enu}}_{\mathcal{M}_F}(k)} \subset F$) generated at this time (resp. at this space) forbid all the words of $\mathcal{L}_k(\mathbf{T}_F)^c$.

2.2 Speed of convergence for previous constructions

In this section, we give some elements to determine the speed of convergence given by the construction of [6] and [2]. The idea is to "program" a d-dimensional subshift of finite type, denoted $\mathbf{T}_{\mathtt{Final}}$ whose projective subaction is a given effective subshift $\Sigma \subset \mathcal{A}^{\mathbb{Z}}$ where $d = 3$ in [6] and $d = 2$ in [2]. In the two constructions, $\mathbf{T}_{\mathtt{Final}}$ is constituted by three layers:

- the first one is $\mathcal{A}^{\mathbb{Z}^d}$ and contains different copies of the same configuration $y \in \mathcal{A}^{\mathbb{Z}}$ superposed on additional directions, the additional finite type conditions check if $y \in \Sigma$;
- the second is $\mathbf{T}_{\mathtt{Grid}} \subset \mathcal{A}^{\mathbb{Z}^d}_{\mathtt{Grid}}$ and constructs a grid which allows to implement well initialized Turing machine in all configurations with different sizes for time and space;
- the third is $\mathbf{T}_{\mathcal{M}} \subset \mathcal{A}^{\mathbb{Z}^d}_{\mathcal{M}}$ and checks if no forbidden pattern appears: the purpose is to implement a Turing machine $\mathcal{M}_F$ which enumerates forbidden patterns which define Σ and an additional procedure $\mathcal{M}_{\mathtt{Search}}$ which checks if the patterns produced appear in the configuration of the first layer (if it is the case, the Turing machine enters in a special state which is forbidden by $\mathbf{T}_{\mathtt{Final}}$).

Thus $x \in \mathbf{T}_{\mathtt{Final}} \subset \mathcal{A}^{\mathbb{Z}^d} \times \mathbf{T}_{\mathtt{Grid}} \times \mathbf{T}_{\mathcal{M}}$ if and only if there exists $y \in \Sigma$ such that $y = \pi(x)_{\mathbf{i}+\mathbb{Z}\mathbf{e_1}}$ for all $\mathbf{i} \in \langle \mathbf{e_2}, \dots, \mathbf{e_d} \rangle_{\mathbb{Z}}$ where π is the factor on the first layer which deletes computation states. In particular $\Sigma = \mathbf{SA}_1(\pi(\mathbf{T}_{\mathtt{Final}}))$ but moreover Σ is conjugate to a sub-action of $\pi(\mathbf{T}_{\mathtt{Final}})$. This result is stronger that just realization by projective subaction and allows to construct local rules for exotic tilings [3, 5].

In the two articles, $\mathbf{T}_{\mathrm{Grid}}$ is defined by a substitution. Mozes' result [13] gives local rules which force a cell to be in a super tile of order n well formed without considering the whole configuration. To determine $\varphi_{F_{\mathtt{Final}},\pi,d\to 1}$, it is sufficient to analyze the size in $\mathbf{T}_{\mathtt{Grid}}$ necessary for that $\mathcal{M}_F$ enumerates patterns of size k and all zones are checked by the additional procedure $\mathcal{M}_{\mathtt{Search}}$. This depends of $\mathrm{Dtime}^{\mathrm{enu}}_{\mathcal{M}_F}(k)$ and $\mathrm{Dspace}^{\mathrm{enu}}_{\mathcal{M}_F}(k)$.

Speed of convergence in the construction of [6]. As it is described in Section 4 of [6], $\mathbf{T}_{\mathtt{Grid}}$ gives a rectangular partition of $\mathbb{Z}^3$ generated by $\widehat{W}_3 \times \widehat{W}_5$ where $\widehat{W}_3$ and $\widehat{W}_5$ are obtained by a substitution. Thus for $s, t \in \mathbb{N}$ there exists $\mathbb{M} \subset \mathbb{Z}$ such that for all $i \in \mathbb{M}$, the slice $\{i\} \times \mathbb{Z}^2$ is partitioned into rectangles of size $3^s \times 5^t$ which delimits computation zones. Moreover $\mathbb{M}$ does not have gap bigger than $3^s 5^t$. To copy the initial configuration onto the first layer, we need an approximation row of width $O(3^s 5^t)$ to detect a forbidden word enumerated in space less than 3^s and in time less than 5^t. One deduces that

$$(k \longmapsto \varphi_{F_{\mathtt{Final}},\pi,3\to 1}) \sim \left(k \longmapsto \mathrm{Dspace}^{\mathrm{enu}}_{\mathcal{M}_F}(k)\mathrm{Dtime}^{\mathrm{enu}}_{\mathcal{M}_F}(k)\right).$$

Speed of convergence in the construction of [2]. As it is described in Section 2, Fact 2.4, of [2], $\mathbf{T}_{\mathtt{Grid}}$ defines fractured zone of computation to implement the Turing machine of size $2^n \times 2^{2^n}$, the first coordinate according to $\mathbf{e_1}$ corresponds to the space and the second according to $\mathbf{e_2}$ corresponds to the time. By the substitution rules and the clock rules, this fractured zone of computation is included in a pattern of $\mathbf{T}_{\mathtt{Grid}}$ of size $4^n \times (2^{n+2^n})$

and every row $\mathbf{T}_{\texttt{Final}}^{2^{n+2^n},2\to1}$ contains such computation zone every 4^n cells. Since the time to cheek if a forbidden pattern of size k appears in the responsibility zone ($n^2 2^n$ steep in direction $\mathbf{e_2}$ by Fact 3.4 of [2]) is negligible according to the time given to the Turing machine to compute forbidden patterns (2^{n+2^n} steep in direction $\mathbf{e_2}$), one deduces that $(k \mapsto \varphi_{F_{\texttt{Final}},\pi,2\to1}(k)) \sim \left(k \mapsto 2^{n(k)+2^{n(k)}}\right)$ where $n(k) = \min\{n : \text{Dspace}^{\text{enu}}_{\mathcal{M}_F}(k) < 2^n\}$. So

$$(k \longmapsto \varphi_{F_{\texttt{Final}},\pi,2\to1}) \sim \left(k \longmapsto \text{Dspace}^{\text{enu}}_{\mathcal{M}_F}(k) 2^{\text{Dspace}^{\text{enu}}_{\mathcal{M}_F}(k)}\right).$$

2.3 A more efficient construction

In the particular case where Σ is an effective subshift with a periodic configuration, the construction can be highly simplified and the speed of convergence is improved. In a few words, the same type of construction with different layers is built, however the computation checks if no forbidden patterns appear only in one line, the other lines are mapped into the periodic configuration by the factor map. Thus the computation zones do not need to be fractionated and simplified layer $\mathbf{T}_{\texttt{Grid}}$ allows a computation in real time.

▶ **Theorem 4.** *Let $\Sigma \subset \mathcal{A}^{\mathbb{Z}^d}$ be an effective subshift of dimension d with a periodic point ($^\infty w^\infty \in \Sigma$) defined by a set F of forbidden patterns enumerated by a Turing machine $\mathcal{M}_F$. Then there exists a subshift of finite type $\mathbf{T}_{F_{Final}}$ of dimension $d+1$ and a factor map π such that $\mathbf{SA}_d(\pi(\mathbf{T}_{F_{Final}})) = \Sigma$ and $\varphi_{F_{Final},\pi,d+1\to d} \sim Dtime^{enu}_{\mathcal{M}_F}$.*

Proof. Assume that $\mathcal{M}_F$ enumerates patterns of F on the first tape separated by the symbol \$ and that the tapes of $\mathcal{M}$ are onesided. The different layers of $\mathbf{T}_{F_{\texttt{Final}}}$ are:

- **Layer 1:** The first layer is $\mathbf{T}_{\texttt{Line}} \subset ((\mathcal{A} \times \{$  $\}) \cup \{$ $\})^{\mathbb{Z}^2}$ the subshift of finite type such that for $x \in \mathbf{T}_{\texttt{Line}}$ there is at most one $i \in \mathbb{Z}$ such that $\mathbf{SA}_1\left(\sigma^{i\mathbf{e_2}}(x)\right) =^\infty$ $^\infty$ and for all $j \neq i$ one has $\mathbf{SA}_1\left(\sigma^{j\mathbf{e_2}}(x)\right) \in \{\sigma^k(^\infty w^\infty) : k \in \mathbb{Z}\} \times \{$  $\}^{\mathbb{Z}}$.
- **Layer 2:** The second layer is the subshift $\mathbf{T}_{\texttt{Config}} = \{x \in \mathcal{A}^{\mathbb{Z}^2} : \sigma^{\mathbf{e_1}-\mathbf{e_2}}(x) = x\}$, the configuration is shifted in view to scan two adjacent areas (and their frontier) during the comparison.
- **Layer 3:** The third layer is the subshift of finite type $\mathbf{T}_{\texttt{Grid}} \subset \{$  $\}^{\mathbb{Z}^2}$ such that on each line of $x \in \mathbf{T}_{\texttt{Grid}}$, the two colors alternates and this alternation is repeated above until it crosses a line which contains the symbol $*$. In this case the transitions red/blue become monochromatic and the transitions blue/red force the alternation. Thus the sequences of monochromatic colors become larger. We remark that if a line contains the periodic configuration $^\infty$ ( $)^\infty$, then all lines below contain this periodic configuration and above, if we have crossed n times a line with the symbol $*$, we obtain a line with the periodic configuration $^\infty$($^{2^n}$ $^{2^n})^\infty$ (see Figure 2).
- **Layer 4:** Denote $\mathcal{A}_{\mathcal{M}} = ((Q \times \Gamma) \cup \Gamma)^k$ where k is the number of tapes, the fourth layer is a subshift of finite type $\mathbf{T}_{\mathcal{M}} \subset \mathcal{A}_{\mathcal{M}}^{\mathbb{Z}^2}$ where the local rules are given by the transition rules δ of $\mathcal{M}_F$.
- **Layer 5:** The fifth layer is the full-shift $\mathbf{T}_{\texttt{Compar}} = \{$  $\}^{\mathbb{Z}^2}$.

To obtain the subshift of finite type $\mathbf{T}_{\texttt{Final}} \subset \mathbf{T}_{\texttt{Line}} \times \mathbf{T}_{\texttt{Config}} \times \mathbf{T}_{\texttt{Grid}} \times \mathbf{T}_{\mathcal{M}} \times \mathbf{T}_{\texttt{Compar}}$ we add a finite set of forbidden patterns $F_{\texttt{SynchroLine}} \cup F_{\texttt{Init}} \cup F_{\texttt{Extend}} \cup F_{\texttt{Compar}}$ which codes the interaction between the different layers. These local rules are:

- **Rules $F_{\texttt{SynchroLine}}$:** These rules imply that if a line $^\infty$  $^\infty$ appears in the layer $\mathbf{T}_{\texttt{Line}}$ of a configuration, then it is synchronized with a periodic point $^\infty$  $^\infty$ or $^\infty$  $^\infty$ in the layer $\mathbf{T}_{\texttt{Grid}}$.

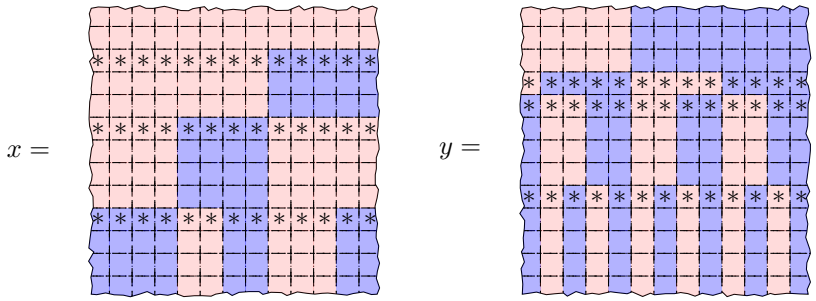


Figure 2 x and y are two examples of configurations of $\mathbf{T}_{\texttt{Grid}}$ and y contains $^{\infty}$ ( )$^{\infty}$.

- **Rules $F_{\texttt{Init}}$:** They imply that the initialization state appears in the layer $\mathbf{T}_{\mathcal{M}}$ on each cell in correspondence to the line $^{\infty}$ $^{\infty}$ in the layer $\mathbf{T}_{\texttt{Line}}$.
- **Rules $F_{\texttt{Extend}}$:** They imply that if a computation needs more space, the symbol $*$ appears in the layer $\mathbf{T}_{\texttt{Grid}}$ (thus the computation zones is doubled) and the tape in the layer $\mathbf{T}_{\mathcal{M}}$ corresponding to the old red zone is erased (to have only one computation by computation zone). Thus the space allowed by a Turing machine is doubled if the head was in a blue zone.
- **Rules $F_{\texttt{Compar}}$:** They imply that if a forbidden pattern appears in the enumeration obtained in $\mathbf{T}_{\mathcal{M}}$ then it is compared with the corresponding pattern which appears in $\mathbf{T}_{\texttt{Config}}$. If the two patterns coincide then the configuration is forbidden in $\mathbf{T}_{\texttt{Final}}$.

Define the factor map $\pi_{\texttt{Final}} : \mathbf{T}_{\texttt{Final}} \to \mathcal{A}^{\mathbb{Z}^2}$ such that for $x \in \mathbf{T}_{\texttt{Final}}$ and $\mathbf{i} \in \mathbb{Z}^2$, $\pi(x)_{\mathbf{i}}$ is the cell of the layer $\mathbf{T}_{\texttt{Config}}$ if we are in the line $^{\infty}$ $^{\infty}$ in $\mathbf{T}_{\texttt{Config}}$ and the cell corresponding to the periodic orbit of $\mathbf{T}_{\texttt{Line}}$ if not.

For $x \in \Sigma$ it is easy to construct $y \in \mathbf{T}_{\texttt{Final}}$ such that $\mathbf{SA}_1(\pi_{\texttt{Final}}(y)) = x$. Reciprocally, consider $y \in \mathbf{T}_{\texttt{Final}}$. If $\pi_{\texttt{Line}}(y)_{(0,0)} \neq$ then $\mathbf{SA}_1(\pi_{\texttt{Final}}(y)) =^{\infty} w^{\infty} \in \Sigma$. If $\pi_{\texttt{Line}}(y)_{(0,0)} =$, we consider u a sub-pattern of $x = \mathbf{SA}_1(\pi_{\texttt{Final}}(y))$ of size n. Assume that $u \notin \mathcal{L}(\Sigma)$, so there exists a word $w \sqsubset x$ enumerated by $\mathcal{M}$ in time $t_F(n) = \text{Dtime}^{\text{enu}}_{\mathcal{M}}(n)$ and space $s_F(n) = \text{Dspace}^{\text{enu}}_{\mathcal{M}}(n)$ such that $w \sqsubset u$. By construction of $\mathbf{T}_{\texttt{Final}}$, one has $\mathbf{SA}_1\left(\pi_{\texttt{Grid}}(\sigma^{t_F(n)}(y))\right) =^{\infty}$ ($^{2^k}$ $^{2^k}$)$^{\infty}$ where $k = \min\{k' : s_F(n) < 2^{k'}\}$. Since the configuration is shifted on $\mathbf{T}_{\texttt{Config}}$ and compared instantaneously thanks to $\mathbf{T}_{\texttt{Compar}}$, we conclude there exists k' such that $t_F(n) \leq k' \leq t_F(n) + 2^{1+min\{k:s_F(n)<2^k\}}$ where the word w is detected in the line $y_{\mathbb{Z},k'}$. By the condition $F_{\texttt{Compar}}$ this is impossible.

Thus $\mathbf{SA}_{\mathbf{e}_1\mathbb{Z}}\left(\pi_{\texttt{Final}}(\mathbf{T}_{\texttt{Final}})\right) = \Sigma$ and $\varphi_{F,\pi,2\to1}(k) = t_F(k) + 2^{1+\min\{n:s_F(k)<2^n\}}$ for all $k \in \mathbb{N}$. In particular $\varphi_{F,\pi,2\to1} \sim \max(\text{Dtime}^{\text{enu}}_{\mathcal{M}_F}, \text{Dspace}^{\text{enu}}_{\mathcal{M}_F}) = \text{Dtime}^{\text{enu}}_{\mathcal{M}_F}$. ◀

2.4 Increase the dimension to increase the speed

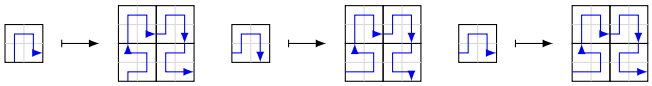
Generally, properties studied on subshifts of finite type exhibit a gap between dimension one and dimension two. The most famous is the undecidability of the domino problem in dimension $d \geq 2$. In this section we exhibit a gap which appears in an algorithmic point of view.

▶ **Theorem 5.** *Let $\Sigma \subset \mathcal{A}^{\mathbb{Z}^d}$ be a subshift which is $(\varphi, d+d')$-realizable by projective subaction of a sofic then it is $(\varphi^{\frac{d'}{d''}}, d+d'')$-realizable by projective subaction of a sofic for $d'' \geq d'$.*

Proof. Let $\Sigma \subset \mathcal{A}^{\mathbb{Z}}$ be an effective subshift. Consider $\mathbf{T}_F \subset \mathcal{B}^{\mathbb{Z}^2}$ a subshift of finite type such that $\mathbf{SA}_1(\pi(\mathbf{T}_F)) = \Sigma$. Denote $\varphi = \varphi_{F,\pi,2\to1}$. One constructs $\mathbf{T}_{F'} \subset \mathcal{B}'^{\mathbb{Z}^3}$ a subshift of finite

type and $\pi' : \mathcal{B}' \to \mathcal{A}$ a factor map such that $\mathbf{SA}_1(\pi'(\mathbf{T}_{F'})) = \Sigma$ and $\varphi_{F',\pi',3\to1} \in \Theta(\sqrt{\varphi})$. This prove the Theorem for $d = 1$, $d' = 1$ and $d'' = 2$.

Construction of a tangled grid. Consider the alphabet $\mathcal{C}$ formed by  ,  ,  , their rotations and their symmetrized about to the axis, thus $\mathrm{card}(\mathcal{C}) = 3 \times 4 \times 2 = 24$ and define the following substitution on $\mathcal{C}$ (modulo rotations and symmetries):

s: 

By iterating substitution s on a letter $a \in \mathcal{C}$, we construct for every $n \in \mathbb{N}$ the pattern $s^n(a)$ called the *super-tile of order n and type a*. The substitutive subshift defined by

$$\mathbf{T}_s = \left\{ x \in \mathcal{C}^{\mathbb{Z}^2} : u \sqsubset x \text{ if there exists } n \in \mathbb{N} \text{ and } a \in \mathcal{C} \text{ which verifies } u \sqsubset s^n(a) \right\},$$

is sofic according to Mozes' result [13]. Thus there exists a finite set of forbidden patterns F_s and a factor map $\pi_s : \mathcal{C}_s \to \mathcal{C}$ such that $\pi_s(\mathbf{T}_{F_s}) = \mathbf{T}_s$. In the Mozes' construction the local rules F_s force every super tile of order n to be assembled in a super tile of order $n+1$. Thus if $p \in \mathcal{C}_s^{[-k,k]^2}$ does not contain patterns of F_s, then the center letter $p_{\mathbf{0}}$ is in a super-tile of order n such that $2^n \leq k < 2^{n+1}$. In this super tile, the arrows form a connected tangled segment of size 2^{n^2}.

Construction of a three-dimensional sofic subshift which realizes Σ. Consider the subshift of finite type $\mathbf{T}_{F'} \subset \mathcal{B}'^{\mathbb{Z}^3}$ where $\mathcal{B}' = \mathcal{B} \times \mathcal{C}_s$ such that

- for all $i \in \mathbb{Z}$ the $\mathbb{Z}^2$-configuration $\pi_{\mathcal{C}_s}(x)_{i\mathbf{e_1}+\mathbb{Z}^2}$ is an element of $\mathbf{T}_s$;
- the 2-dimensional forbidden patterns of F are transfered in 3-dimensional forbidden patterns where the second coordinate is wrapped following the tangled grid (see Figure 3).

Let π' be the application of π following the tangled grid, we obtain $\mathbf{SA}_1(\pi'(\mathbf{T}_{F'})) = \Sigma$.

$\pi'(\mathbf{T}_{F'})$ has the expected speed of convergence. By definition of the speed of convergence, for any $u \in \mathcal{A}^k$, if $u \notin \Sigma$ then $u \notin \mathcal{L}\left(\pi\left(\mathbf{SA}_1\left(\mathbf{T}_F^{\varphi(k),2\to1}\right)\right)\right)$.

Let $z \in \mathbf{T}_{F'}^{2\left\lceil\sqrt{\varphi(k)}\right\rceil,3\to1}$. The condition F_s verified on $z_{\{0\}\times\left[-2\left\lceil\sqrt{\varphi(k)}\right\rceil,2\left\lceil\sqrt{\varphi(k)}\right\rceil\right]^2}$ implies that $z_{\mathbf{0}}$ is included in a super-tile of order $n = \left\lfloor \log_2\left(\left\lceil\sqrt{\varphi(k)}\right\rceil\right)\right\rfloor$. One deduces that $z_{\mathbf{0}}$ is in the center of a segment constituted following the arrows of $\mathcal{C}$ of amplitude $\left(2^{\left\lfloor\log_2\left(\left\lceil\sqrt{\varphi(k)}\right\rceil\right)\right\rfloor}\right)^2$. Like the local transitions F are transfered, there exists $y \in \mathbf{SA}_1\left(\mathbf{T}_F^{\varphi(k),2\to1}\right)$ which correspond to z in the wrapped zone. Thus $u \notin \mathcal{L}\left(\mathbf{T}_{F'}^{2\left\lceil\sqrt{\varphi(k)}\right\rceil,3\to1}\right)$ that is to say $\varphi_{F',\pi',3\to1} \prec \sqrt{\varphi}$. In the same way the reverse holds and so $\varphi_{F',\pi',3\to1} \succ \sqrt{\varphi}$. ◀

▶ Remark. Exemples of Section 4 show that this theorem is optimal.

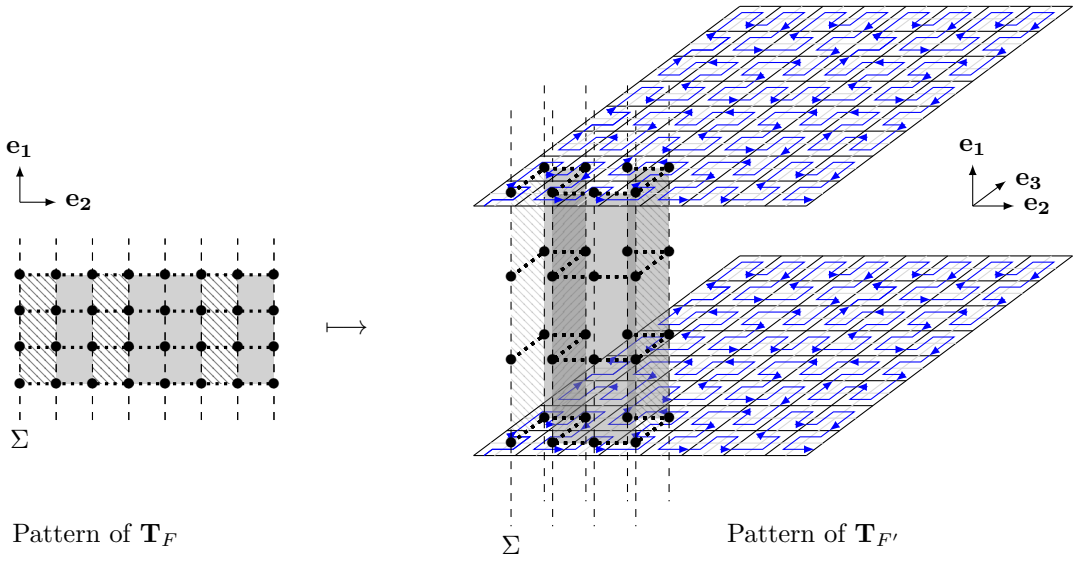


Figure 3 Pattern of $\mathbf{T}_F$ wrapped following the tangled grid in a pattern of $\mathbf{T}_{F'}$. The subshift Σ is obtained taking factor π or π' and projective subaction following $\mathbf{e_1}$.

3 Lower bounds for the speed of convergence of a subshift

3.1 Combinatorial lower bounds

Let Σ be a one dimensional subshift and let $u \in \mathcal{A}^*$, the *follower set of word of size* k of u is $\mathrm{Fol}_\Sigma^k(u) = \{v \in \mathcal{L}_k(\Sigma) : uv \in \mathcal{L}(\Sigma)\}$. If $u \notin \mathcal{L}(\Sigma)$ then $\mathrm{Fol}_\Sigma^k(u) = \emptyset$. Moreover one has $\mathrm{card}(\{\mathrm{Fol}_\Sigma^{k_2}(u) : u \in \mathcal{A}^{k_1}\}) \leq \mathrm{card}(\mathcal{A})^{k_1}$.

▶ **Theorem 6.** *Let $\Sigma \subset \mathcal{A}^{\mathbb{Z}}$ be an one dimensional effective subshift and $\varphi \in \mathcal{F}_{\Sigma,d}^{Sofic}$ with $d \geq 2$. Then there exists a constant M such that for all $k_1, k_2 \in \mathbb{N}$ one has:*

$$M\varphi_{F,\pi,d\to 1}(k_1+k_2) \geq \left(\log(card(\{Fol_\Sigma^{k_2}(u) : u \in \mathcal{A}^{k_1}\}))\right)^{\frac{1}{d-1}}.$$

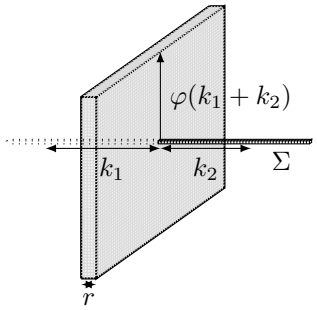
Proof. Assume that $\Sigma = \mathbf{SA}_1(\pi(\mathbf{T}_F))$ and $\varphi = \varphi_{F,\pi,d\to 1}$. For $u \in \mathcal{L}_{k_1}(\Sigma)$, one has

$$\mathrm{Fol}_\Sigma^{k_2}(u) = \Big\{\mathbf{SA}_1(\pi(x))_{[0,k_2-1]} \in \mathcal{A}^{k_2} : x \in \mathbf{T}_F^{\varphi(k_1+k_2),d\to 1} \text{ such that } \mathbf{SA}_1(\pi(x))_{[-k_1,-1]} = u\Big\}.$$

Let r such that the support of every pattern of F is included in $[0, r-1]^d$. For $x \in \mathbf{T}_F^{\varphi(k_1+k_2),d\to 1} \subset \mathcal{B}^{\mathbb{Z}^d}$ such that $\mathbf{SA}_1(\pi(x))_{[-k_1,-1]} = u \in \mathcal{A}^{k_1}$, the knowledge of $x_{[-r,-1]\times[-\varphi(k_1+k_2),\varphi(k_1+k_2)]^{d-1}}$ is sufficient to determine which set of $\{\mathrm{Fol}_\Sigma^{k_2}(u') : u' \in \mathcal{A}^{k_1}\}$ is allowed to complete $u \in \mathcal{A}^{k_1}$ by a word $v \in \mathcal{A}^{k_2}$ such that

$$uv \in \mathcal{L}_{k_1+k_2}(\mathbf{SA}_1\left(\pi(\mathbf{T}_F^{\varphi(k_1+k_2),d\to 1}\right)) = \mathcal{L}_{k_1+k_2}(\Sigma).$$

Thus $\mathrm{card}(\{\mathrm{Fol}_\Sigma^{k_2}(u) : u \in \mathcal{A}^{k_1}\}) \leq \mathcal{B}^{r(2\varphi(k_1+k_2)+1)^{d-1}}$.



◀

3.2 Computational lower bounds

▶ **Theorem 7.** *Let $\Sigma \subset \mathcal{A}^{\mathbb{Z}}$ be an one dimensional effective subshift and $\varphi \in \mathcal{F}_{\Sigma,d}^{Sofic}$. There exists a Turing machine $\mathcal{M}$ whose the domaine is $\mathcal{L}(\Sigma)^c$ such that*

- $\max(\log, (\varphi_{F,\pi,d\to 1})^{d-1}) \succ \log \circ Dtime_{\mathcal{M}}$;
- $(\varphi_{F,\pi,d\to 1})^{d-1} \succ Dspace_{\mathcal{M}}$.

Since $\mathcal{L}(\Sigma)^c$ is not necessarily recursive, $\mathrm{Dtime}_{\mathcal{M}}$ and $\mathrm{Dspace}_{\mathcal{M}}$ are not necessarily computable.

Proof. Let F be a finite set of forbidden patterns of maximal size r such that $\Sigma = \mathbf{SA}_1(\pi(\mathbf{T}_F))$ and $\varphi = \varphi_{F,\pi,d\to 1}$. Denote $\mathbb{B}_n = \{k_2\mathbf{e_2} + \cdots + k_d\mathbf{e_d} : (k_2, \ldots, k_d) \in [-n, n]^{d-1}\}$ and $\mathbf{T}^m = \mathbf{T}_F^{m,d\to 1}$. One has $\mathcal{L}_k\left(\mathbf{SA}_1\left(\pi(\mathbf{T}^{\varphi(k)})\right)\right) = \mathcal{L}_k(\Sigma)$ and $\mathbf{T}^{\varphi(k)} \subset \left(\mathcal{B}^{\mathbb{B}_{\varphi(k)}}\right)^{\mathbb{Z}}$ is a one-dimensional subshift of finite type of order r. This subshift can be represented by a graph where the vertices are $\left(\mathcal{B}^{\mathbb{B}_{\varphi(k)}}\right)^r \cap \mathcal{L}(\mathbf{T}^{\varphi(k)})$ and there is an edge from u to v if the two words coincide except for the extremal letters (see [11]). Thus this graph has at most $\mathrm{card}(\mathcal{B})^{r(2\varphi(k))^{d-1}}$ vertices and can be viewed as an automaton which accepts words of $\mathcal{L}(\pi(\mathbf{T}^{\varphi(k)}))$, this takes a linear time in the size of the graph.

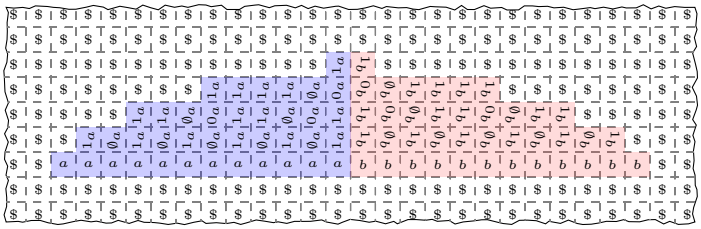
To determine if $u \notin \mathcal{L}(\Sigma)$, it is sufficient that $u \notin \mathcal{L}(\mathbf{SA}_1(\pi(\mathbf{T}^m)))$ for some $m \in \mathbb{N}$. We implement an algorithm which explores the graph generated by $\mathbf{T}^m$ for each $m \in \mathbb{N}$ and search if u is accepted with the corresponding automaton. One knows if $u \in \mathcal{L}(\mathbf{SA}_1(\pi(\mathbf{T}^m)))$ in time $O(k\,\mathrm{card}(\mathcal{A})^{r(2m)^{d-1}})$. This algorithm halts on $u \notin \mathcal{L}(\Sigma)$ in time

$$\mathrm{Dtime}_{\mathcal{M}}(k) \leq M\,k \sum_{m=1}^{\varphi(k)} \mathrm{card}(\mathcal{A})^{r(2m)^{d-1}} \leq M\,k\,\varphi(k)\mathrm{card}(\mathcal{A})^{r(2\varphi(k))^{d-1}}$$

Since $\varphi(k) \leq \varphi^{d-1}(k)$, it follows that $\max(\log, \varphi^{d-1}) \succ \log \circ \mathrm{Dtime}_{\mathcal{M}}$.We deduce the first point of the theorem.

To prove the second point, the naive procedure to find a configuration of $\mathbf{SA}_1(\pi(\mathbf{T}^m))$ which contains u in the center is to start from an element of $\left(\mathcal{B}^{\mathbb{B}_n}\right)^r \cap \mathcal{L}(\mathbf{T}^m)$ and complete it respecting the condition F until it finds again a one-sided periodic orbit. To be sure to explore all the orbits it is possible to order them lexicographically. Thus, the algorithm just needs to know the last orbit checked, this needs $r(2m)^{d-1}$ space to know if $u \in \mathcal{L}(\mathbf{SA}_1(\pi(\mathbf{T}^m)))$. If $u \notin \mathcal{L}(\Sigma)$, the algorithm halts when it explores $\left(\mathcal{B}^{\mathbb{B}_n}\right)^r \cap \mathcal{L}(\mathbf{T}^{\varphi(k)})$. So there exists $M > 0$ such that $M(\varphi(k))^{d-1} \geq \mathrm{Dspace}_{\mathcal{M}}(k)$. We recall that the word u is written on an annex tape which is only used for the reading and which is not counted in $\mathrm{Dspace}_{\mathcal{M}}$. ◀

▶ Remark. These theorems do not generalize to dimension 2: Theorem 6 uses a characterization of one dimensional sofic subshifts with follower sets and Theorem 7 is blocked by the undecidability of emptiness of two-dimensional subshifts of finite type.



A configuration of $\mathbf{T}_{F_{\log}}$

Figure 4 A configuration of $\mathbf{T}_{F_{\log}}$.

4 Some classes of speed of convergence and perspectives

In this section we give the sharp realization of some one-dimensional subshifts.

▶ **Sofic subshift.** A subshift is constant-realizable by sofic if and only if it is sofic (see [14]).

▶ **Gap under constant-realizable.** If a subshift $\Sigma \subset \mathcal{A}^{\mathbb{Z}}$ is $(\varphi, 2)$-realizable by sofic with $\varphi \in o(\log(\log(n)))$ then this subshift is sofic. Indeed, by Theorem 7, $\mathcal{L}(\Sigma)^c$ can be recognized in space $o(\log(\log(n)))$, thus $\mathcal{L}(\Sigma)^c$ is rational (see [8]), that is to say Σ is sofic.

Let $\mathcal{L} \subset \mathcal{A}^*$ be a language and $\$ \notin \mathcal{A}$. Define the subshift $\mathbf{T}(\mathcal{L}) = \mathbf{T}_{F_{\mathcal{L}}} \subset \mathcal{A}'^{\mathbb{Z}}$ where $\mathcal{A}' = \mathcal{A} \cup \{\$\}$ and $F = \{\$u\$: u \notin \mathcal{L}\}$. If $\mathcal{L}$ is effective then $\mathbf{T}(\mathcal{L})$ is an effective subshift.

▶ **log-realizable.** Consider $\mathcal{L}_= = \{a^n b^n : n \in \mathbb{N}\}$. The subshift $\mathbf{T}(\mathcal{L}_=) \subset \{a, b, \$\}^{\mathbb{Z}}$ is sharp $((\log)^{\frac{1}{d-1}}, d)$-realizable by sofic for $d \geq 2$. Theorem 6 gives the lower bound since

$$\mathrm{card}(\{\mathrm{Fol}^n_\Sigma(u) : u \in \mathcal{A}^n\}) \geq \mathrm{card}(\{\mathrm{Fol}^k_\Sigma(\$a^k) : k \in [0, n-1]\}) = n.$$

For $d = 2$, the upper bound is obtained considering the subshift of finite type $\mathbf{T}_{F_{\log}} \subset \{a, b, \$, 0_a, 1_a, \emptyset_a, 0_b, 1_b, \emptyset_b\}^{\mathbb{Z}^2}$ where $F_{\log}$ are the forbidden patterns of shape $\mathbb{U} =$  which do not appear in the configuration represented in Figure 4. The factor π maps \$ on \$, $\{0_a, 1_a, \emptyset_a\}$ on a and $\{0_b, 1_b, \emptyset_b\}$ on b. The idea is to implement counters which grow when going from \$'s region and compare them at the frontier. The upper bound for $d \geq 3$ is obtained using Theorem 5.

In the same way the subshift $\mathbf{T}(\mathcal{L}_{\mathrm{square}})$ defined with the langage $\mathcal{L}_{\mathrm{square}} = \{a^n b^{n^2} : n \in \mathbb{N}\}$ is sharp $((\log)^{\frac{1}{d-1}}, d)$-realizable by sofic.

▶ **Linear-realizable.** For $u \in \{0,1\}^*$, define $\overline{u}$ the miror of u. Consider $\mathcal{L}_{\mathrm{palin}} = \{u\overline{u} : u \in \{0,1\}^*\}$, the subshift $\mathbf{T}(\mathcal{L}_{\mathrm{palin}}) \subset \{0, 1, \$\}^{\mathbb{Z}}$ is sharp $((\mathrm{Id})^{\frac{1}{d-1}}, d)$-realizable by projective subaction of sofic for $d \geq 2$ where $\mathrm{Id} : k \mapsto k$. Theorem 6 gives the lower bound.

For $d = 2$, the upper bound is obtained considering the subshift $\mathbf{T}_{F_{\mathrm{lin}}} \subset \{\$, 0_l, 1_l, 0_r, 1_r\}^{\mathbb{Z}^2}$ where F_{lin} are the patterns of shape $\mathbb{U} =$  or which do not appear in the configuration represented in Figure 5. The factor π maps \$ on \$, $\{0_l, 0_r\}$ on 0 and $\{1_l, 1_r\}$ on 1. The principle is to compare vertically the two words of $\{0,1\}^*$. The upper bound for $d \geq 3$ is obtained using Theorem 5.

▶ **Dspace$_{\mathcal{M}}$ realizable.** Let $\mathcal{L}$ be a computable language in space $\mathrm{Dspace}_{\mathcal{M}}$ and $\# \notin \mathcal{L}$. Consider $\mathcal{L}' = \{u\#^{\mathrm{Dtime}_{\mathcal{M}}(|u|)}\}$, then $\mathbf{T}(\mathcal{L}')$ is sharp $(\mathrm{Dspace}_{\mathcal{M}}, 2)$-realizable (the time of the Turing machine is coded following $\mathbf{e_1}$ in the sofic which realizes $\mathbf{T}(\mathcal{L}')$).

▶ **Substitutive subshift.** Let s be a one-dimensional substitution, $\mathbf{T}_s \cup \{^\infty a^\infty\}$ is sharp $(\log, 2)$-realisable. The lower bound is given by Theorem 6 and the upper bound follows from the sofic subshift where elements of $\mathbf{T}_s$ are in at most in one row. If it appears, this row is de-substituted in the next row in direction $\mathbf{e_2}$.

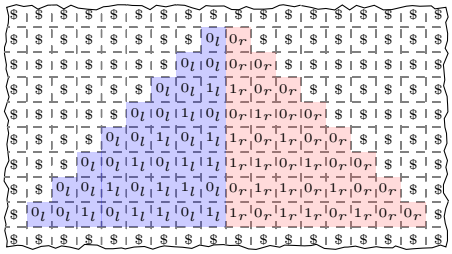


Figure 5 A configuration of $\mathbf{T}_{F_{\text{lin}}}$.

▶ **No-computable realization.** Consider the recursively enumerable set $F = \{01^n0 : n \text{ such that the Turing machine of number } n \text{ halts}\}$. Then $\varphi \in \mathcal{F}_{\Sigma,2}^{Sofic}$ is larger than any recursive function, otherwise it is possible to decide if the Turing machine of number n halts.

▶ **Perspectives.** This article highlights the importance of algorithmic properties and optimality in the realization of effective subshifts by sofic. Particularly, the last section exhibits the existence of different subclasses of effective subshift but does not present systematic study: characterization of classes of subshifts with the same speed of convergence, links between dynamical properties and speed of convergence, sharp realization for effective subshift without periodic point (as the substitutive subshift $\mathbf{T}_s$)...

Acknowledgements. The authors want to thank the anonymous referees for their detailed reviews which helped us to clarify the paper.

References

1. Nathalie Aubrun and Mathieu Sablik. An order on sets of tilings corresponding to an order on languages. In Susanne Albers, Susanne Albers, Susanne Albers, Susanne Albers, and Jean-Yves Marion, editors, *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 3 of *LIPIcs*, pages 99–110. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Germany, 2009. doi:10.4230/LIPIcs.STACS.2009.1833.
2. Nathalie Aubrun and Mathieu Sablik. Simulation of Effective Subshifts by Two-dimensional Subshifts of Finite Type. *Acta Appl. Math.*, 126:35–63, 2013. doi:10.1007/s10440-013-9808-5.
3. Nathalie Aubrun and Mathieu Sablik. Multidimensional effective s-adic systems are sofic. *Uniform Distribution Theory*, 9(2), 2014.
4. Bruno Durand, Andrei Romashchenko, and Alexander Shen. Fixed-point tile sets and their applications. *J. Comput. System Sci.*, 78(3):731–764, 2012. doi:10.1016/j.jcss.2011.11.001.
5. Thomas Fernique and Mathieu Sablik. Local rules for computable planar tilings. In *Proceedings 18th international workshop on Cellular Automata and Discrete Complex Systems and 3rd international symposium Journées Automates Cellulaires*, pages 133–141, 2012. doi:10.4204/EPTCS.90.11.
6. Michael Hochman. On the dynamics and recursive properties of multidimensional symbolic systems. *Invent. Math.*, 176(1):131–167, 2009. doi:10.1007/s00222-008-0161-7.
7. Michael Hochman and Tom Meyerovitch. A characterization of the entropies of multi-dimensional shifts of finite type. *Annals of Mathematics*, 171(3):2011–2038, 2010. doi:10.4007/annals.2010.171.2011.

8 John E. Hopcroft and Jeffrey D. Ullman. Some results on tape-bounded turing machines. *J. ACM*, 16(1):168–177, 1969. doi:10.1145/321495.321508.

9 Emmanuel Jeandel and Pascal Vanier. π_1^0 sets and tilings. In *TAMC'11*, pages 230–239, 2011. doi:10.1007/978-3-642-20877-5_24.

10 Emmanuel Jeandel and Pascal Vanier. Hardness of conjugacy, embedding and factorization of multidimensional subshifts of finite type. In *STACS'13*, pages 490–501, 2013. doi:10.4230/LIPIcs.STACS.2013.490.

11 Douglas A. Lind and Brian Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, New York, NY, USA, 1995.

12 Alejandro Maass. On the sofic limit sets of cellular automata. *Ergodic Theory Dynam. Systems*, 15(4):663–684, 1995. doi:10.1017/S0143385700008609.

13 Shahar Mozes. Tilings, substitutions systems and dynamical systems generated by them. *J. d'Analyse Math.*, 53:139–186, 1989.

14 Ronie Pavlov and Michael Schraudner. Classification of sofic projective subdynamics of multidimensional shifts of finite type. *to appear in J. d'Analyse Math.*, 2010.

15 Stephen G Simpson. Medvedev degrees of 2-dimensional subshifts of finite type. *To appear in Ergodic Theory and Dynamical Systems*, 2010.

16 Hao Wang. Proving theorems by Pattern Recognition II. *Bell Systems technical journal*, 40:1–41, 1961.

17 Benjamin Weiss. Subshifts of finite type and sofic systems. *Monatsh. Math.*, 77:462–474, 1973.

On Word and Frontier Languages of Unsafe Higher-Order Grammars*†

Kazuyuki Asada[1] and Naoki Kobayashi[2]

1 **The University of Tokyo, Tokyo, Japan**
asada@kb.is.s.u-tokyo.ac.jp
2 **The University of Tokyo, Tokyo, Japan**
koba@is.s.u-tokyo.ac.jp

Abstract

Higher-order grammars are an extension of regular and context-free grammars, where non-terminals may take parameters. They have been extensively studied in 1980's, and restudied recently in the context of model checking and program verification. We show that the class of unsafe order-(n+1) word languages coincides with the class of frontier languages of unsafe order-n tree languages. We use intersection types for transforming an order-(n+1) word grammar to a corresponding order-n tree grammar. The result has been proved for safe languages by Damm in 1982, but it has been open for unsafe languages, to our knowledge. Various known results on higher-order grammars can be obtained as almost immediate corollaries of our result.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases intersection types, higher-order grammars

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.111

1 Introduction

Higher-order grammars are an extension of regular and context-free grammars, where non-terminals may take trees or (higher-order) functions on trees as parameters. They were extensively studied in the 1980's [6, 7, 8], and recently reinvestigated in the context of model checking [10, 17] and applied to program verification [11].

The present paper shows that the class of unsafe order-$(n+1)$ word languages coincides with the class of "frontier languages" of unsafe order-n tree languages. Here, the frontier of a tree is the sequence of symbols that occur in the leaves of the tree from left to right, and the frontier language of a tree language consists of the frontiers of elements of the tree language. The special case where $n = 0$ corresponds to the well-known fact that the frontier language of a regular tree language is a context-free language. The result has been proved by Damm [6] for grammars with the safety restriction (see [16] for a nice historical account of the safety restriction), but it has been open for unsafe grammars, to our knowledge.[1]

Damm's proof relied on the safety restriction (in particular, the fact that variable renaming is not required for safe grammars [3]) and does not apply (at least directly) to the case of unsafe grammars. We instead use intersection types to transform an order-$(n+1)$ word grammar $\mathcal{G}$ to an order-n tree grammar $\mathcal{G}'$ such that the frontier language of $\mathcal{G}'$ coincides

* A full version [2] of the paper is available at http://arxiv.org/abs/1604.01595.

† This work was supported by JSPS Kakenhi 23220001 and 15H05706.

[1] Kobayashi et al. [13] mentioned the result, referring to the paper under preparation: "On Unsafe Tree and Leaf Languages," which is actually the present paper.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 111; pp. 111:1–111:13

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

with the language generated by $\mathcal{G}$. Intersection types have been used for recent other studies of higher-order grammars and model checking [11, 13, 12, 15, 19, 18, 14, 20]; our proof in the present paper provides even more evidence that intersection types are a versatile tool for studies of higher-order grammars. Compared with the previous work on intersection types for higher-order grammars, the technical novelties include: (i) our intersection types (used in Section 3) are mixtures of non-linear and linear intersection types and (ii) our type-based transformation involves global restructuring of terms. These points have made the correctness of the transformations non-trivial and delicate.

As stressed by Damm [6] at the beginning of his paper, the result will be useful for analyzing properties of higher-order languages by induction on the order of grammars. Our result allows properties on (unsafe) order-n languages to be reduced to those on order-$(n-1)$ tree languages, and then the latter may be studied by investigating those on the path languages of order-$(n-1)$ tree languages, which are order-$(n-1)$ word languages.

The rest of this paper is structured as follows. Section 2 reviews the definition of higher-order grammars, and states the main result. Sections 3 and 4 prove the result by providing the (two-step) transformations from order-$(n+1)$ word grammars to order-n tree grammars. Section 5 discusses applications of the result. Section 6 discusses related work and Section 7 concludes the paper. For the space restriction, we omit some details and proofs, which are found in the full version [2].

2 Preliminaries

This section defines higher-order grammars and the languages generated by them, and then explains the main result. Most of the following definitions follow those in [13].

A higher-order grammar consists of non-deterministic rewriting rules of the form $A \to t$, where A is a non-terminal and t is a simply-typed λ-term that may contain non-terminals and terminals (tree constructors).

▶ **Definition 1** (types and terms)**.** The set of *simple types*,[2] ranged over by κ, is given by: $\kappa ::= \mathtt{o} \mid \kappa_1 \to \kappa_2$. The order and arity of a simple type κ, written $\mathtt{order}(\kappa)$ and $\mathtt{ar}(\kappa)$, are defined respectively by:

$$\mathtt{order}(\mathtt{o}) = 0 \qquad \mathtt{order}(\kappa_1 \to \kappa_2) = \max(\mathtt{order}(\kappa_1) + 1, \mathtt{order}(\kappa_2))$$
$$\mathtt{ar}(\mathtt{o}) = 0 \qquad \mathtt{ar}(\kappa_1 \to \kappa_2) = 1 + \mathtt{ar}(\kappa_2)$$

The type $\mathtt{o}$ describes trees, and $\kappa_1 \to \kappa_2$ describes functions from κ_1 to κ_2. The set of *λ-terms*, ranged over by t, is defined by: $t ::= x \mid A \mid a \mid t_1\, t_2 \mid \lambda x : \kappa.t$. Here, x ranges over variables, A over symbols called non-terminals, and a over symbols called terminals. We assume that each terminal a has a fixed arity; we write Σ for the map from terminals to their arities. A term t is called an *applicative term* (or simply a *term*) if it does not contain λ-abstractions. A (simple) type environment $\mathcal{K}$ is a sequence of type bindings of the form $x : \kappa$ such that if $\mathcal{K}$ contains $x : \kappa$ and $x' : \kappa'$ in different positions then $x \neq x'$. In type environments, non-terminals are also treated as variables. A λ-term t has type κ under $\mathcal{K}$ if $\mathcal{K} \vdash_{\mathtt{ST}} t : \kappa$ is derivable from the following typing rules.

$$\frac{}{\mathcal{K}, x : \kappa, \mathcal{K}' \vdash_{\mathtt{ST}} x : \kappa} \qquad \frac{}{\mathcal{K} \vdash_{\mathtt{ST}} a : \underbrace{\mathtt{o} \to \cdots \to \mathtt{o}}_{\Sigma(a)} \to \mathtt{o}}$$

[2] We sometimes call simple types *sorts* in this paper, to avoid confusion with intersection types introduced later for grammar transformations.

$$\frac{\mathcal{K} \vdash_{\mathtt{ST}} t_1 : \kappa_2 \to \kappa \qquad \mathcal{K} \vdash_{\mathtt{ST}} t_2 : \kappa_2}{\mathcal{K} \vdash_{\mathtt{ST}} t_1\, t_2 : \kappa} \qquad \frac{\mathcal{K},\, x : \kappa_1 \vdash_{\mathtt{ST}} t : \kappa_2}{\mathcal{K} \vdash_{\mathtt{ST}} \lambda x : \kappa_1.t : \kappa_1 \to \kappa_2}$$

We call t a (finite, Σ-ranked) *tree* if t is an applicative term consisting of only terminals, and $\vdash_{\mathtt{ST}} t : \mathtt{o}$ holds. We write $\mathbf{Tree}_\Sigma$ for the set of Σ-ranked trees, and use the meta-variable π for a tree.

We often omit type annotations and just write $\lambda x.t$ for $\lambda x : \kappa.t$. We consider below only well-typed λ-terms of the form $\lambda x_1.\cdots\lambda x_k.t$, where t is an applicative term. We are now ready to define higher-order grammars.

▶ **Definition 2** (higher-order grammar)**.** A *higher-order grammar* is a quadruple $(\Sigma, \mathcal{N}, \mathcal{R}, S)$, where (i) Σ is a ranked alphabet; (ii) $\mathcal{N}$ is a map from a finite set of non-terminals to their types; (iii) $\mathcal{R}$ is a finite set of *rewriting rules* of the form $A \to \lambda x_1.\cdots\lambda x_\ell.t$, where $\mathcal{N}(A) = \kappa_1 \to \cdots \to \kappa_\ell \to \mathtt{o}$, t is an applicative term, and $\mathcal{N}, x_1 : \kappa_1, \ldots, x_\ell : \kappa_\ell \vdash_{\mathtt{ST}} t : \mathtt{o}$ holds for some $\kappa_1, \ldots, \kappa_\ell$. (iv) S is a non-terminal called *the start symbol*, and $\mathcal{N}(S) = \mathtt{o}$. The *order* of a grammar $\mathcal{G}$, written $\mathtt{order}(\mathcal{G})$, is the largest order of the types of non-terminals. We sometimes write $\Sigma_\mathcal{G}, \mathcal{N}_\mathcal{G}, \mathcal{R}_\mathcal{G}, S_\mathcal{G}$ for the four components of $\mathcal{G}$.

For a grammar $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, the rewriting relation $\longrightarrow_\mathcal{G}$ is defined by:

$$\frac{(A \to \lambda x_1.\cdots\lambda x_k.t) \in \mathcal{R}}{A\, t_1 \cdots t_k \longrightarrow_\mathcal{G} [t_1/x_1, \ldots, t_k/x_k]t} \qquad \frac{t_i \longrightarrow_\mathcal{G} t_i' \qquad i \in \{1, \ldots, k\} \qquad \Sigma(a) = k}{a\, t_1 \cdots t_k \longrightarrow_\mathcal{G} a\, t_1 \cdots t_{i-1}\, t_i'\, t_{i+1} \cdots t_k}$$

Here, $[t_1/x_1, \ldots, t_k/x_k]t$ is the term obtained by substituting t_i for the free occurrences of x_i in t. We write $\longrightarrow_\mathcal{G}^*$ for the reflexive transitive closure of $\longrightarrow_\mathcal{G}$.

The *tree language generated by* $\mathcal{G}$, written $\mathcal{L}(\mathcal{G})$, is the set $\{\pi \in \mathbf{Tree}_{\Sigma_\mathcal{G}} \mid S \longrightarrow_\mathcal{G}^* \pi\}$. We call a grammar $\mathcal{G}$ a *word grammar* if all the terminal symbols have arity 1 except the special terminal $\mathtt{e}$, whose arity is 0. The *word language* generated by a word grammar $\mathcal{G}$, written $\mathcal{L}_\mathtt{w}(\mathcal{G})$, is $\{a_1 \cdots a_n \mid a_1(\cdots(a_n\, \mathtt{e})\cdots) \in \mathcal{L}(\mathcal{G})\}$. The frontier word of a tree π, written $\mathbf{leaves}(\pi)$, is the sequence of symbols in the leaves of π. It is defined inductively by: $\mathbf{leaves}(a) = a$ when $\Sigma(a) = 0$, and $\mathbf{leaves}(a\, \pi_1 \cdots \pi_k) = \mathbf{leaves}(\pi_1) \cdots \mathbf{leaves}(\pi_k)$ when $\Sigma(a) = k > 0$. The *frontier language* generated by $\mathcal{G}$, written $\mathcal{L}_\mathtt{leaf}(\mathcal{G})$, is the set: $\{\mathbf{leaves}(\pi) \mid S \longrightarrow_\mathcal{G}^* \pi \in \mathbf{Tree}_{\Sigma_\mathcal{G}}\}$. In our main theorem, we assume that there is a special nullary symbol $\mathtt{e}$ and consider $\mathtt{e} \in \mathcal{L}_\mathtt{leaf}(\mathcal{G})$ as the empty word ε; i.e., we consider $\mathcal{L}^\varepsilon_\mathtt{leaf}(\mathcal{G})$ defined by:

$$\mathcal{L}^\varepsilon_\mathtt{leaf}(\mathcal{G}) := (\mathcal{L}_\mathtt{leaf}(\mathcal{G}) \setminus \{\mathtt{e}\}) \cup \{\varepsilon \mid \mathtt{e} \in \mathcal{L}_\mathtt{leaf}(\mathcal{G})\}.$$

We note that the classes of order-0 and order-1 word languages coincide with those of regular and context-free languages respectively. We often write $A\, x_1 \cdots x_k \to t$ for the rule $A \to \lambda x_1.\cdots\lambda x_k.t$. When considering the frontier language of a tree grammar, we assume, without loss of generality, that the ranked alphabet Σ has a unique binary symbol $\mathtt{br}$, and that all the other terminals have arity 0.

▶ **Example 3.** Consider the order-2 (word) grammar $\mathcal{G}_1 = (\{\mathtt{a} : 1, \mathtt{b} : 1, \mathtt{e} : 0\}, \{S : \mathtt{o}, F : (\mathtt{o} \to \mathtt{o}) \to \mathtt{o}, A : (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}), B : (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o})\}, \mathcal{R}_1, S)$, where $\mathcal{R}_1$ consists of:

$$S \to F\, \mathtt{a} \qquad S \to F\, \mathtt{b} \qquad A\, f\, x \to \mathtt{a}(f\, x) \qquad B\, f\, x \to \mathtt{b}(f\, x),$$
$$F\, f \to f(f\, \mathtt{e}) \qquad F\, f \to F\, (A\, f) \qquad F\, f \to F\, (B\, f).$$

S is reduced, for example, as follows.

$$S \longrightarrow F\, \mathtt{b} \longrightarrow F\, (A\, \mathtt{b}) \longrightarrow (A\, \mathtt{b})(A\, \mathtt{b}\, \mathtt{e}) \longrightarrow \mathtt{a}\, (\mathtt{b}\, (A\, \mathtt{b}\, \mathtt{e})) \longrightarrow \mathtt{a}(\mathtt{b}\, (\mathtt{a}(\mathtt{b}\, \mathtt{e}))).$$

The word language $\mathcal{L}_{\mathtt{w}}(\mathcal{G}_1)$ is $\{ww \mid w \in \{\mathtt{a},\mathtt{b}\}^+\}$.

Consider the order-1 (tree) grammar $\mathcal{G}_2 = (\{\mathtt{br}\!:\!2, \mathtt{a}\!:\!0, \mathtt{b}\!:\!0, \mathtt{e}\!:\!0\}, \{S\!:\!\mathtt{o}, F\!:\!\mathtt{o}\to\mathtt{o}\}, \mathcal{R}_2, S)$, where $\mathcal{R}_2$ consists of:

$$S \to F\,\mathtt{a} \qquad S \to F\,\mathtt{b} \qquad F\,f \to \mathtt{br}\,f\,f \qquad F\,f \to F(\mathtt{br}\,\mathtt{a}\,f) \quad F\,f \to F(\mathtt{br}\,\mathtt{b}\,f).$$

The frontier language $\mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}_2)$ coincides with $\mathcal{L}_{\mathtt{w}}(\mathcal{G}_1)$ above.

The following is the main theorem we shall prove in this paper.

▶ **Theorem 4.** *For any order-$(n+1)$ word grammar $\mathcal{G}$ ($n \geq 0$), there exists an order-n tree grammar $\mathcal{G}'$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.*

The converse of the above theorem also holds:

▶ **Theorem 5.** *For any order-n tree grammar $\mathcal{G}'$ such that no word in $\mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$ contains $\mathtt{e}$, there exists a word grammar $\mathcal{G}$ of order at most $n+1$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.*

Since the construction of $\mathcal{G}$ is easy, we sketch it here; For $n \geq 1$, the grammar $\mathcal{G}$ is obtained by (i) changing the arity of each nullary terminal a ($\neq \mathtt{e}$) to one, i.e., $\Sigma_{\mathcal{G}}(a) := 1$, (ii) replacing the terminal $\mathtt{e}$ with a new non-terminal E of type $\mathtt{o}\to\mathtt{o}$, defined by $E\,x \to x$, and also the unique binary terminal $\mathtt{br}$ with a new non-terminal Br of type $(\mathtt{o}\to\mathtt{o})\to(\mathtt{o}\to\mathtt{o})\to(\mathtt{o}\to\mathtt{o})$, defined by $Br\,f\,g\,x \to f(g\,x)$, (iii) applying η-expansion to the right hand side of each (original) rule to add an order-0 argument, and (iv) adding new start symbol S' with rule $S' \to S\,\mathtt{e}$. For example, given the grammar $\mathcal{G}_2$ above, the following grammar is obtained:

$$\begin{array}{l} S' \to S\,\mathtt{e} \qquad S\,x \to F\,\mathtt{a}\,x \qquad S\,x \to F\,\mathtt{b}\,x \\ F\,f\,x \to Br\,f\,f\,x \qquad F\,f\,x \to F(Br\,\mathtt{a}\,f)\,x \qquad F\,f\,x \to F(Br\,\mathtt{b}\,f)\,x \\ E\,x \to x \qquad Br\,f\,g\,x \to f(g\,x). \end{array}$$

Theorem 4 is proved by two-step grammar transformations, both of which are based on intersection types. In the first step, we transform an order-$(n+1)$ word grammar $\mathcal{G}$ to an order-n tree grammar $\mathcal{G}''$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'')\!\uparrow_{\mathtt{e}}$, where $\mathcal{L}\!\uparrow_{\mathtt{e}}$ is the word language obtained from $\mathcal{L}$ by removing all the occurrences of the special terminal $\mathtt{e}$; that is, the frontier language of $\mathcal{G}''$ is almost the same as $\mathcal{L}_{\mathtt{w}}(\mathcal{G})$, except that the former may contain multiple occurrences of the special, dummy symbol $\mathtt{e}$. In the second step, we clean up the grammar to eliminate $\mathtt{e}$ (except that a singleton tree $\mathtt{e}$ may be generated when $\epsilon \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$). The first and second steps shall be formalized in Sections 3 and 4 respectively.

For the target of the transformations, we use the following extended terms, in which a *set* of terms may occur in an argument position:

$$\begin{array}{rl} u \text{ (extended terms)} & ::= x \mid A \mid a \mid u_0 U \mid \lambda x.u \\ U & ::= \{u_1, \ldots, u_k\}\ (k \geq 1). \end{array}$$

Here, $u_0\,u_1$ is interpreted as just a shorthand for $u_0\{u_1\}$. Intuitively, $\{u_1,\ldots,u_k\}$ is considered a non-deterministic choice $u_1 + \cdots + u_k$, which (lazily) reduces to u_i non-deterministically. The typing rules are extended accordingly by:

$$\frac{\mathcal{K} \vdash_{\mathtt{ST}} u_0 : \kappa_1 \to \kappa \qquad \mathcal{K} \vdash_{\mathtt{ST}} U : \kappa_1}{\mathcal{K} \vdash_{\mathtt{ST}} u_0\,U : \kappa} \qquad\qquad \frac{\mathcal{K} \vdash_{\mathtt{ST}} u_i : \kappa \text{ for each } i \in \{1,\ldots,k\}}{\mathcal{K} \vdash_{\mathtt{ST}} \{u_1,\ldots,u_k\} : \kappa}$$

An *extended higher-order grammar* is the same as a higher-order grammar, except that each rewriting rule in $\mathcal{R}$ may be of the form $\lambda x_1 \cdots \lambda x_\ell.u$, where u may be an applicative extended term. The reduction rule for non-terminals is replaced by:

$$\frac{(A \to \lambda x_1 \cdots \lambda x_k.u) \in \mathcal{R} \qquad u' \in [U_1/x_1, \ldots, U_k/x_k]u}{A\,U_1 \cdots U_k \longrightarrow_{\mathcal{G}} u'}$$

where the substitution θu is defined by:

$$\theta a = \{a\} \qquad \theta x = \begin{cases} \theta(x) & (\text{if } x \in dom(\theta)) \\ \{x\} & (\text{otherwise}) \end{cases}$$
$$\theta(u_0 U) = \{v(\theta U) \mid v \in \theta u_0\} \qquad \theta\{u_1, \ldots, u_k\} = \theta u_1 \cup \cdots \cup \theta u_k\,.$$

Also, the other reduction rule is replaced by the following two rules:

$$\frac{u \longrightarrow_{\mathcal{G}} u' \qquad i \in \{1, \ldots, k\} \qquad \Sigma(a) = k}{a\,U_1 \cdots U_{i-1}\,\{u\}\,U_{i+1} \cdots U_k \longrightarrow_{\mathcal{G}} a\,U_1 \cdots U_{i-1}\,\{u'\}\,U_{i+1} \cdots U_k}$$

$$\frac{u \in U_i \qquad U_i \text{ is not a singleton} \qquad i \in \{1, \ldots, k\} \qquad \Sigma(a) = k}{a\,U_1 \cdots U_k \longrightarrow_{\mathcal{G}} a\,U_1 \cdots U_{i-1}\,\{u\}\,U_{i+1} \cdots U_k}$$

Note that unlike in the extended grammar introduced in [13], there is no requirement that each u_i in $\{u_1, \ldots, u_k\}$ is used at least once. Thus, the extended syntax does not change the expressive power of grammars. A term set $\{u_1, \ldots, u_k\}$ can be replaced by $A\,x_1 \cdots x_\ell$ with the rewriting rules $A\,x_1 \cdots x_\ell \to u_i$, where $\{x_1, \ldots, x_\ell\}$ is the set of variables occurring in some of $u_1, \ldots, u_k$. In other words, for any order-n extended grammar $\mathcal{G}$, there is an (ordinary) order-n grammar $\mathcal{G}'$ such that $\mathcal{L}(\mathcal{G}) = \mathcal{L}(\mathcal{G}')$.

3 Step 1: from order-$(n+1)$ grammars to order-n tree grammars

In this section, we show that for any order-$(n+1)$ grammar $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ such that $\Sigma(\mathtt{e}) = 0$ and $\Sigma(a) = 1$ for every $a \in dom(\Sigma) \setminus \{\mathtt{e}\}$, there exists an order-$n$ grammar $\mathcal{G}'$ such that $\Sigma_{\mathcal{G}'} = \{\mathtt{br} \mapsto 2, \mathtt{e} \mapsto 0\} \cup \{a \mapsto 0 \mid \Sigma(a) = 1\}$ and $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}')\uparrow_{\mathtt{e}}$.

For technical convenience, we assume below that, for every type κ occurring in $\mathcal{N}_{\mathcal{G}}(A)$ for some A, if κ is of the form $\mathtt{o} \to \kappa'$, then $\mathtt{order}(\kappa') \leq 1$. This does not lose generality, since any function $\lambda x : \mathtt{o}.t$ of type $\mathtt{o} \to \kappa'$ with $\mathtt{order}(\kappa') > 1$ can be replaced by the term $\lambda x' : \mathtt{o} \to \mathtt{o}.[x'\mathtt{e}/x]t$ of type $(\mathtt{o} \to \mathtt{o}) \to \kappa'$ (without changing the order of the term), and any term t of type $\mathtt{o}$ can be replaced by the term $K\,t$ of type $\mathtt{o} \to \mathtt{o}$, where K is a non-terminal of type $\mathtt{o} \to \mathtt{o} \to \mathtt{o}$, with rule $K\,x\,y \to x$. See [2] for the details of this transformation.

The basic idea of the transformation is to remove all the order-0 arguments (i.e., arguments of tree type $\mathtt{o}$). This reduces the order of each term by 1; for example, terms of types $\mathtt{o} \to \mathtt{o}$ and $(\mathtt{o} \to \mathtt{o}) \to \mathtt{o}$ will respectively be transformed to those of types $\mathtt{o}$ and $\mathtt{o} \to \mathtt{o}$. Order-0 arguments can indeed be removed as follows. Suppose we have a term $t_1\,t_2$ where $t_1 : \mathtt{o} \to \mathtt{o}$. If t_1 does not use the order-0 argument t_2, then we can simply replace $t_1\,t_2$ with $t_1^{\#}$ (where $t_1^{\#}$ is the result of recursively applying the transformation to t_1). If t_1 uses the argument t_2, the word generated by $t_1\,t_2$ must be of the form $w_1 w_2$, where w_2 is generated by t_2; in other words, t_1 can only append a word to the word generated by t_2. Thus, $t_1\,t_2$ can be transformed to $\mathtt{br}\;t_1^{\#}\;t_2^{\#}$, which can generate a tree whose frontier coincides with $w_1 w_2$ (if $\mathtt{e}$ is ignored). As a special case, a constant word $\mathtt{a}\,\mathtt{e}$ can be transformed to $\mathtt{br}\;\mathtt{a}\;\mathtt{e}$. As a little more complex example, consider the term $A\,(\mathtt{b}\,\mathtt{e})$, where A is defined by $A\,x \to \mathtt{a}\,x$. Since A uses the argument, the term $A\,(\mathtt{b}\,\mathtt{e})$ is transformed to $\mathtt{br}\;A\;(\mathtt{br}\;\mathtt{b}\;\mathtt{e})$. Since A no longer takes an argument, we substitute $\mathtt{e}$ for x in the body of the rule for A (and apply the transformation recursively to $\mathtt{a}\,\mathtt{e}$). The resulting rule for A is: $A \to \mathtt{br}\;\mathtt{a}\;\mathtt{e}$. Thus, the

term after the transformation generates the tree $\mathtt{br}\,(\mathtt{br}\,\mathtt{a}\,\mathtt{e})\,(\mathtt{br}\,\mathtt{b}\,\mathtt{e})$. Its frontier word is $\mathtt{aebe}$, which is equivalent to the word $\mathtt{ab}$ generated by the original term, up to removals of $\mathtt{e}$; recall that redundant occurrences of $\mathtt{e}$ will be removed by the second transformation. Note that the transformation sketched above depends on whether each order-0 argument is actually used or not. Thus, we introduce intersection types to express such information, and define the transformation as a type-directed one.

Simple types are refined to the following intersection types.

$$\delta ::= \mathtt{o} \mid \sigma \to \delta \qquad \sigma ::= \delta_1 \wedge \cdots \wedge \delta_k \quad (k \geq 0)$$

We write $\top$ for $\delta_1 \wedge \cdots \wedge \delta_k$ when $k = 0$. We assume some total order $<$ on intersection types, and require that $\delta_1 < \cdots < \delta_k$ whenever $\delta_1 \wedge \cdots \wedge \delta_k$ occurs in an intersection type. Intuitively, $(\delta_1 \wedge \cdots \wedge \delta_k) \to \delta$ describes a function that uses an argument according to types $\delta_1, \ldots, \delta_k$, and the returns a value of type δ. As a special case, the type $\top \to \mathtt{o}$ describes a function that ignores an argument, and returns a tree. Thus, according to the idea of the transformation sketched above, if x has type $\top \to \mathtt{o}$, $x\,t$ would be transformed to x; if x has type $\mathtt{o} \to \mathtt{o}$, $x\,t$ would be transformed to $\mathtt{br}\;x\;t^{\#}$. In the last example above, the type $\mathtt{o} \to \mathtt{o}$ should be interpreted as a function that uses the argument *just once*; otherwise the transformation to $\mathtt{br}\;x\;t^{\#}$ would be incorrect. Thus, the type $\mathtt{o}$ should be treated as a linear type, for which weakening and dereliction are disallowed. In contrast, we need not enforce, for example, that a value of the intersection type $\mathtt{o} \to \mathtt{o}$ should be used just once. Therefore, we classify intersection types into two kinds; one called *balanced*, which may be treated as non-linear types, and the other called *unbalanced*, which must be treated as linear types. For that purpose, we introduce two refinement relations $\delta ::_{\mathrm{b}} \kappa$ and $\delta ::_{\mathrm{u}} \kappa$; the former means that δ is a balanced intersection type of sort κ, and the latter means that δ is an unbalanced intersection type of sort κ. The relations are defined as follows, by mutual induction; k may be 0.

$$\frac{\delta_j ::_{\mathrm{u}} \kappa \qquad j \in \{1, \ldots, k\} \qquad \delta_i ::_{\mathrm{b}} \kappa \ (\text{for each } i \in \{1, \ldots, k\} \setminus \{j\})}{\delta_1 \wedge \cdots \wedge \delta_k ::_{\mathrm{u}} \kappa} \qquad \frac{\delta_i ::_{\mathrm{b}} \kappa \ (\text{for each } i \in \{1, \ldots, k\})}{\delta_1 \wedge \cdots \wedge \delta_k ::_{\mathrm{b}} \kappa}$$

$$\frac{}{\mathtt{o} ::_{\mathrm{u}} \mathtt{o}} \qquad \frac{\sigma ::_{\mathrm{b}} \kappa \qquad \delta ::_{\mathrm{u}} \kappa'}{\sigma \to \delta ::_{\mathrm{u}} \kappa \to \kappa'} \qquad \frac{\sigma ::_{\mathrm{u}} \kappa \qquad \delta ::_{\mathrm{u}} \kappa'}{\sigma \to \delta ::_{\mathrm{b}} \kappa \to \kappa'} \qquad \frac{\sigma ::_{\mathrm{b}} \kappa \qquad \delta ::_{\mathrm{b}} \kappa'}{\sigma \to \delta ::_{\mathrm{b}} \kappa \to \kappa'}$$

A type δ is called *balanced* if $\delta ::_{\mathrm{b}} \kappa$ for some κ, and called *unbalanced* if $\delta ::_{\mathrm{u}} \kappa$ for some κ. Intuitively, unbalanced types describe trees or closures that contain the end of a word (i.e., symbol $\mathtt{e}$). Intersection types that are neither balanced nor unbalanced are considered ill-formed, and excluded out. For example, the type $\mathtt{o} \to \mathtt{o} \to \mathtt{o}$ (as an intersection type) is ill-formed; since $\mathtt{o}$ is unbalanced, $\mathtt{o} \to \mathtt{o}$ must also be unbalanced according to the rules for arrow types, but it is actually balanced. Note that, in fact, no term can have the intersection type $\mathtt{o} \to \mathtt{o} \to \mathtt{o}$ in a word grammar. We write $\delta :: \kappa$ if $\delta ::_{\mathrm{b}} \kappa$ or $\delta ::_{\mathrm{u}} \kappa$.

We introduce a type-directed transformation relation $\Gamma \vdash t : \delta \Rightarrow u$ for terms, where Γ is a set of type bindings of the form $x : \delta$, called a *type environment*, t is a source term, and u is the image of the transformation, which may be an extended term. We write $\Gamma_1 \cup \Gamma_2$ for the union of Γ_1 and Γ_2; it is defined only if, whenever $x : \delta \in \Gamma_1 \cap \Gamma_2$, δ is balanced. In other words, unbalanced types are treated as linear types, whereas balanced ones as non-linear (or idempotent) types. We write $\mathbf{bal}(\Gamma)$ if δ is balanced for every $x : \delta \in \Gamma$.

The relation $\Gamma \vdash t : \delta \Rightarrow u$ is defined inductively by the following rules.

$$\frac{\mathbf{bal}(\Gamma)}{\Gamma, x : \delta \vdash x : \delta \Rightarrow x_\delta} \ (\textsc{Tr1-Var}) \qquad \frac{A :: \mathcal{N}(A) \qquad \mathbf{bal}(\Gamma)}{\Gamma \vdash A : \delta \Rightarrow A_\delta} \ (\textsc{Tr1-NT})$$

$$\frac{\mathbf{bal}(\Gamma)}{\Gamma \vdash \mathtt{e} : \mathtt{o} \Rightarrow \mathtt{e}} \quad (\textsc{Tr1-Const0}) \qquad \frac{\Sigma(a) = 1 \qquad \mathbf{bal}(\Gamma)}{\Gamma \vdash a : \mathtt{o} \to \mathtt{o} \Rightarrow a} \quad (\textsc{Tr1-Const1})$$

$$\frac{\begin{array}{c}\Gamma_0 \vdash s : \delta_1 \wedge \cdots \wedge \delta_k \to \delta \Rightarrow v \\ \Gamma_i \vdash t : \delta_i \Rightarrow U_i \text{ and } \delta_i \neq \mathtt{o} \text{ (for each } i \in \{1, \ldots, k\})\end{array}}{\Gamma_0 \cup \Gamma_1 \cup \cdots \cup \Gamma_k \vdash st : \delta \Rightarrow v U_1 \cdots U_k} \quad (\textsc{Tr1-App1})$$

$$\frac{\Gamma_0 \vdash s : \mathtt{o} \to \delta \Rightarrow V \qquad \Gamma_1 \vdash t : \mathtt{o} \Rightarrow U}{\Gamma_0 \cup \Gamma_1 \vdash st : \delta \Rightarrow \mathtt{br}\, V\, U} \quad (\textsc{Tr1-App2})$$

$$\frac{\Gamma \vdash t : \delta \Rightarrow u_i \text{ (for each } i \in \{1, \ldots, k\}) \qquad k \geq 1}{\Gamma \vdash t : \delta \Rightarrow \{u_1, \ldots, u_k\}} \quad (\textsc{Tr1-Set})$$

$$\frac{\begin{array}{c}\Gamma, x : \delta_1, \ldots, x : \delta_k \vdash t : \delta \Rightarrow u \qquad x \notin dom(\Gamma) \\ \delta_i \neq \mathtt{o} \text{ for each } i \in \{1, \ldots, k\}\end{array}}{\Gamma \vdash \lambda x.t : \delta_1 \wedge \cdots \wedge \delta_k \to \delta \Rightarrow \lambda x_{\delta_1} \cdots \lambda x_{\delta_k}.u} \quad (\textsc{Tr1-Abs1})$$

$$\frac{\Gamma, x : \mathtt{o} \vdash t : \delta \Rightarrow u}{\Gamma \vdash \lambda x.t : \mathtt{o} \to \delta \Rightarrow [\mathtt{e}/x_{\mathtt{o}}]u} \quad (\textsc{Tr1-Abs2})$$

In rule (Tr1-Var), a variable is replicated for each type. This is because the image of the transformation of a term substituted for x is different depending on the type of the term; accordingly, in rule (Tr1-Abs1), bound variables are also replicated, and in rule (Tr1-App1), arguments are replicated. In rule (Tr1-NT), a non-terminal is also replicated for each type. In rules (Tr1-Const0) and (Tr1-Const1), constants are mapped to themselves; however, the arities of all the constants become 0. In these rules, Γ may contain only bindings on balanced types.

In rule (Tr1-App1), the first premise indicates that the function s uses the argument t according to types $\delta_1, \ldots, \delta_k$. Since the image of the transformation of t depends on its type, we replicate the argument to $U_1, \ldots, U_k$. For each type δ_i, the result of the transformation is not unique (but finite); thus, we represent the image of the transformation as a *set* U_i of terms. (Recall the remark at the end of Section 2 that a set of terms can be replaced by an ordinary term by introducing auxiliary non-terminals.) For example, consider a term $A(x\,y)$. It can be transformed to $A_{\delta_1 \to \delta}\{x_{\delta_0 \to \delta_1} y_{\delta_0}, x_{\delta_0' \to \delta_1} y_{\delta_0'}\}$ under the type environment $\{x : \delta_0 \to \delta_1, x : \delta_0' \to \delta_1, y : \delta_0, y : \delta_0'\}$. Note that k in rule (Tr1-App1) (and also (Tr1-Abs1)) may be 0, in which case the argument disappears in the image of the transformation.

In rule (Tr1-App2), as explained at the beginning of this section, the argument t of type $\mathtt{o}$ is removed from s and instead attached as a sibling node of the tree generated by (the transformation image of) s. Accordingly, in rule (Tr1-Abs2), the binder for x is removed and x in the body of the abstraction is replaced with the empty tree $\mathtt{e}$. In rule (Tr1-Set), type environments are shared. This is because $\{u_1, \ldots, u_k\}$ represents the choice $u_1 + \cdots + u_k$; unbalanced (i.e. linear) values should be used in the same manner in $u_1, \ldots, u_k$.

The transformation rules for rewriting rules and grammars are given by:

$$\frac{\emptyset \vdash \lambda x_1. \cdots \lambda x_k.t : \delta \Rightarrow \lambda x_1'. \cdots \lambda x_\ell'.u \qquad \delta :: \mathcal{N}(A)}{(A\, x_1\, \cdots\, x_k \to t) \Rightarrow (A_\delta\, x_1'\, \cdots\, x_\ell' \to u)} \quad (\textsc{Tr1-Rule})$$

$$\frac{\Sigma' = \{\mathtt{br} \mapsto 2, \mathtt{e} \mapsto 0\} \cup \{a \mapsto 0 \mid \Sigma(a) = 1\} \quad \mathcal{N}' = \{A_\delta : [\![\delta :: \kappa]\!] \mid \mathcal{N}(A) = \kappa \wedge \delta :: \kappa\} \quad \mathcal{R}' = \{r' \mid \exists r \in \mathcal{R}. r \Rightarrow r'\}}{(\Sigma, \mathcal{N}, \mathcal{R}, S) \Rightarrow (\Sigma', \mathcal{N}', \mathcal{R}', S_{\mathtt{o}})} \quad (\textsc{Tr1-Gram})$$

Here, $[\![\delta :: \kappa]\!]$ is defined by:

$$[\![\delta :: \kappa]\!] = \mathtt{o} \qquad (\text{if } \mathtt{order}(\kappa) \leq 1)$$
$$[\![(\delta_1 \wedge \cdots \wedge \delta_k \to \delta) :: (\kappa_0 \to \kappa)]\!] = [\![\delta_1 :: \kappa_0]\!] \to \ldots \to [\![\delta_k :: \kappa_0]\!] \to [\![\delta :: \kappa]\!] \qquad (\text{if } \mathtt{order}(\kappa_0 \to \kappa) > 1)$$

▶ **Example 6.** Recall the grammar $\mathcal{G}_1$ in Example 3. For the term $\lambda f.\lambda x.\mathtt{a}(f\,x)$ of the rule for A, we have the following derivation:

$$\dfrac{\dfrac{\dfrac{}{\emptyset \vdash \mathtt{a} : \mathtt{o} \to \mathtt{o} \Rightarrow \mathtt{a}}\ \textsc{Const1} \quad \dfrac{\dfrac{}{f : \mathtt{o} \to \mathtt{o} \vdash f : \mathtt{o} \to \mathtt{o} \Rightarrow f_{\mathtt{o}\to\mathtt{o}}}\ \textsc{Var} \quad \dfrac{}{x : \mathtt{o} \vdash x : \mathtt{o} \Rightarrow x_{\mathtt{o}}}\ \textsc{Var}}{f : \mathtt{o} \to \mathtt{o}, x : \mathtt{o} \vdash f\,x : \mathtt{o} \Rightarrow \mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, x_{\mathtt{o}}}\ \textsc{App2}}{f : \mathtt{o} \to \mathtt{o}, x : \mathtt{o} \vdash \mathtt{a}(f\,x) : \mathtt{o} \Rightarrow \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, x_{\mathtt{o}})}\ \textsc{App2}}{f : \mathtt{o} \to \mathtt{o} \vdash \lambda x.\mathtt{a}(f\,x) : \mathtt{o} \to \mathtt{o} \Rightarrow \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, \mathtt{e})}\ \textsc{Abs2}}{\emptyset \vdash \lambda f.\lambda x.\mathtt{a}(f\,x) : (\mathtt{o} \to \mathtt{o}) \to \mathtt{o} \to \mathtt{o} \Rightarrow \lambda f_{\mathtt{o}\to\mathtt{o}}.\mathtt{br}\,\mathtt{a}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, \mathtt{e})}\ \textsc{Abs1}$$

Notice that the argument x has been removed, and the result of the transformation has type $\mathtt{o} \to \mathtt{o}$. The whole grammar is transformed to the grammar consisting of the following rules.

$$S_{\mathtt{o}} \to F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\,\mathtt{a} \qquad S_{\mathtt{o}} \to F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\,\mathtt{b}$$
$$A_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, \mathtt{e}) \qquad B_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to \mathtt{br}\,\mathtt{b}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, \mathtt{e})$$
$$F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to \mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\,(\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, \mathtt{e}) \qquad F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}(A_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}})$$
$$F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to F_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}(B_{(\mathtt{o}\to\mathtt{o})\to\mathtt{o}\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}}).$$

Here, we have omitted rules that are unreachable from $S_{\mathtt{o}}$. For example, the rule

$$F_{(\top\to\mathtt{o})\wedge(\mathtt{o}\to\mathtt{o})\to\mathtt{o}}\, f_{\top\to\mathtt{o}}\, f_{\mathtt{o}\to\mathtt{o}} \to \mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, f_{\top\to\mathtt{o}}$$

may be obtained from the following derivation, but it is unreachable from $S_{\mathtt{o}}$, since F is never called with an argument of type $(\top \to \mathtt{o}) \wedge (\mathtt{o} \to \mathtt{o})$.

$$\dfrac{\dfrac{\dfrac{}{f : \mathtt{o} \to \mathtt{o} \vdash f \Rightarrow f_{\mathtt{o}\to\mathtt{o}}}\ \textsc{Var} \quad \dfrac{\dfrac{}{f : \top \to \mathtt{o} \vdash f : \top \to \mathtt{o} \Rightarrow f_{\top\to\mathtt{o}}}\ \textsc{Var}}{f : \top \to \mathtt{o} \vdash f\,\mathtt{e} : \mathtt{o} \Rightarrow f_{\top\to\mathtt{o}}}\ \textsc{App1}}{f : \top \to \mathtt{o}, f : \mathtt{o} \to \mathtt{o} \vdash f(f\,\mathtt{e}) : \mathtt{o} \Rightarrow \mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, f_{\top\to\mathtt{o}}}\ \textsc{App2}}{\emptyset \vdash \lambda f.f(f\,\mathtt{e}) : (\top \to \mathtt{o}) \wedge (\mathtt{o} \to \mathtt{o}) \to \mathtt{o} \Rightarrow \lambda f_{\top\to\mathtt{o}}.\lambda f_{\mathtt{o}\to\mathtt{o}}.\mathtt{br}\, f_{\mathtt{o}\to\mathtt{o}}\, f_{\top\to\mathtt{o}}}\ \textsc{Abs1}$$

The following theorem states the correctness of the first transformation.

▶ **Theorem 7.** *Let $\mathcal{G}$ be an order-$(n+1)$ word grammar. If $\mathcal{G} \Rightarrow \mathcal{G}''$, then $\mathcal{G}''$ is an (extended) grammar of order at most n. Furthermore, $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}_{\mathtt{leaf}}(\mathcal{G}'')\uparrow_{\mathtt{e}}$.*

4 Step 2: removing dummy symbols

We now describe the second step for eliminating redundant symbols $\mathtt{e}$, which have been introduced by (Tr1-Abs2). By the remark at the end of Section 2, we assume that the result of the first transformation is an ordinary grammar, not containing extended terms. We also assume that $\mathtt{br}$ occurs only in the fully applied form. This does not lose generality, because otherwise we can replace $\mathtt{br}$ by a new non-terminal Br and add the rule $Br\,x\,y \to \mathtt{br}\,x\,y$.

The idea of the transformation is to use intersection types to distinguish between terms that generate trees consisting of only $\mathtt{br}$ and $\mathtt{e}$, and those that generate trees containing other arity-0 terminals. We assign the type $\mathtt{o}_\epsilon$ to the former terms, and $\mathtt{o}_+$ to the latter. A term $\mathtt{br}\,t_0\,t_1$ is transformed to (i) $\mathtt{br}\,t_0^\#\,t_1^\#$ if both t_0 and t_1 have type $\mathtt{o}_+$ (where $t_i^\#$ is the image of the transformation of t_i), (ii) $t_i^\#$ if t_i has type $\mathtt{o}_+$ and t_{1-i} has type $\mathtt{o}_\epsilon$, and (iii) $\mathtt{e}$ if both t_0 and t_1 have type $\mathtt{o}_\epsilon$. As in the transformation of the previous section, we replicate each non-terminal and variable for each intersection type. For example, the nonterminal $A : \mathtt{o} \to \mathtt{o}$ defined by $A\,x \to x$ would be replicated to $A_{\mathtt{o}_+\to\mathtt{o}_+}$ and $A_{\mathtt{o}_\epsilon\to\mathtt{o}_\epsilon}$.

We first define the set of intersection types by:

$$\xi ::= \mathtt{o}_\epsilon \mid \mathtt{o}_+ \mid \xi_1 \wedge \cdots \wedge \xi_k \to \xi$$

We assume some total order $<$ on intersection types, and require that whenever we write $\xi_1 \wedge \cdots \wedge \xi_k$, $\xi_1 < \cdots < \xi_k$ holds. We define the refinement relation $\xi :: \kappa$ inductively by: (i) $\mathtt{o}_\epsilon :: \mathtt{o}$, (ii) $\mathtt{o}_+ :: \mathtt{o}$, and (iii) $(\xi_1 \wedge \cdots \wedge \xi_k \to \xi) :: (\kappa_1 \to \kappa_2)$ if $\xi :: \kappa_2$ and $\xi_i :: \kappa_1$ for every $i \in \{1, \ldots, k\}$. We consider only types ξ such that $\xi :: \kappa$ for some κ. For example, we forbid an ill-formed type like $\mathtt{o}_+ \wedge (\mathtt{o}_+ \to \mathtt{o}_+) \to \mathtt{o}_+$.

We introduce a type-based transformation relation $\Xi \vdash t : \xi \Rightarrow u$, where Ξ is a type environment (i.e., a set of bindings of the form $x : \xi$), t is a source term, ξ is the type of t, and u is the result of transformation. The relation is defined inductively by the rules below.

$$\frac{}{\Xi, x : \xi \vdash x : \xi \Rightarrow x_\xi}\ (\textsc{Tr2-Var}) \qquad \frac{}{\Xi \vdash \mathtt{e} : \mathtt{o}_\epsilon \Rightarrow \mathtt{e}}\ (\textsc{Tr2-Const0}) \qquad \frac{\Sigma(a) = 0 \qquad a \neq \mathtt{e}}{\Xi \vdash a : \mathtt{o}_+ \Rightarrow a}\ (\textsc{Tr2-Const1})$$

$$\frac{\Xi \vdash t_0 : \xi_0 \Rightarrow u_0 \qquad \Xi \vdash t_1 : \xi_1 \Rightarrow u_1 \qquad (u, \xi) = \begin{cases} (\mathtt{br}\,u_0\,u_1, \mathtt{o}_+) & \text{if } \xi_0 = \xi_1 = \mathtt{o}_+ \\ (u_i, \mathtt{o}_+) & \text{if } \xi_i = \mathtt{o}_+ \text{ and } \xi_{1-i} = \mathtt{o}_\epsilon \\ (\mathtt{e}, \mathtt{o}_\epsilon) & \text{if } \xi_0 = \xi_1 = \mathtt{o}_\epsilon \end{cases}}{\Xi \vdash \mathtt{br}\,t_0\,t_1 : \xi \Rightarrow u}\ (\textsc{Tr2-Const2})$$

$$\frac{\xi :: \mathcal{N}(F) \qquad A\,x_1\,\cdots\,x_k \to t \in \mathcal{R} \qquad \emptyset \vdash \lambda x_1.\cdots\lambda x_k.t : \xi \Rightarrow \lambda y_1.\cdots\lambda y_\ell.u}{\Xi \vdash A : \xi \Rightarrow A_\xi}\ (\textsc{Tr2-NT})$$

$$\frac{\Xi \vdash s : \xi_1 \wedge \cdots \wedge \xi_k \to \xi \Rightarrow v \qquad \Xi \vdash t : \xi_i \Rightarrow U_i \text{ (for each } i \in \{1, \ldots, k\})}{\Xi \vdash st : \xi \Rightarrow vU_1 \cdots U_k}\ (\textsc{Tr2-App})$$

$$\frac{\Xi \vdash t : \xi \Rightarrow u_i \text{ (for each } i \in \{1, \ldots, k\}) \qquad k \geq 1}{\Xi \vdash t : \xi \Rightarrow \{u_1, \ldots, u_k\}}\ (\textsc{Tr2-Set})$$

$$\frac{\Xi, x : \xi_1, \ldots, x : \xi_k \vdash t : \xi \Rightarrow u}{\Xi \vdash \lambda x.t : \xi_1 \wedge \cdots \wedge \xi_k \to \xi \Rightarrow \lambda x_{\xi_1} \cdots \lambda x_{\xi_k}.u}\ (\textsc{Tr2-Abs})$$

The transformation of rewriting rules and grammars is defined by:

$$\frac{\emptyset \vdash \lambda x_1.\cdots\lambda x_k.t : \xi \Rightarrow \lambda x'_1.\cdots\lambda x'_\ell.t' \qquad \xi :: \mathcal{N}(A)}{(A \to \lambda x_1.\cdots\lambda x_k.t) \Rightarrow (A_\xi \to \lambda x'_1.\cdots\lambda x'_\ell.t')}\ (\textsc{Tr2-Rule})$$

$$\frac{\begin{array}{c}\mathcal{N}' = \{A_\xi : [\![\xi]\!] \mid \mathcal{N}(A) = \kappa \wedge \xi :: \kappa\} \\ \mathcal{R}' = \{r' \mid \exists r \in \mathcal{R}.r \Rightarrow r'\} \cup \{S' \to S_{\mathtt{o}_\epsilon}, S' \to S_{\mathtt{o}_+}\}\end{array}}{(\Sigma, \mathcal{N}, \mathcal{R}, S) \Rightarrow (\Sigma, \mathcal{N}', \mathcal{R}', S')}\ (\textsc{Tr2-Gram})$$

Here, $[\![\xi]\!]$ is defined by:

$$[\![\mathtt{o}_\epsilon]\!] = [\![\mathtt{o}_+]\!] = \mathtt{o} \qquad [\![\,\xi_1 \wedge \cdots \wedge \xi_k \to \xi]\!] = [\![\xi_1]\!] \to \cdots \to [\![\xi_k]\!] \to [\![\xi]\!]$$

We explain some key rules. In (Tr2-Var) we replicate a variable for each type, as in the first transformation. The rules (Tr2-Const0) and (Tr2-Const1) are for nullary constants, which are mapped to themselves. We assign type $\mathtt{o}_\epsilon$ to $\mathtt{e}$ and $\mathtt{o}_+$ to the other constants. The rule (Tr2-Const2) is for the binary tree constructor $\mathtt{br}$. As explained above, we eliminate terms that generate empty trees (those consisting of only $\mathtt{br}$ and $\mathtt{e}$). For example, if $\xi_0 = \mathtt{o}_\epsilon$ and $\xi_1 = \mathtt{o}_+$, then t_0 may generate an empty tree; thus, the whole term is transformed to u_1.

The rule (Tr2-NT) replicates a terminal for each type, as in the case of variables. The middle and rightmost premises require that there is some body t of A that can indeed be transformed according to type ξ. Without this condition, for example, A defined by the rule $A \to A$ would be transformed to $A_{\mathtt{o}_\epsilon}$ by $\emptyset \vdash A : \mathtt{o}_\epsilon \Rightarrow A_{\mathtt{o}_\epsilon}$, but $A_{\mathtt{o}_\epsilon}$ diverges and does not produce an empty tree. That would make the rule (Tr2-Const2) unsound: when a source term is $\mathtt{br}\,A\,\mathtt{a}$, it would be transformed to $\mathtt{a}$, but while the original term does not generate a tree, the result of the transformation does. In short, the two premises are required to ensure that whenever $\emptyset \vdash t : \mathtt{o}_\epsilon \Rightarrow u$ holds, t can indeed generate an empty tree. In (Tr2-App), the argument is replicated for each type. Unlike in the transformation in the previous section, type environments can be shared among the premises, since linearity does not matter here. The other rules for terms are analogous to those in the first transformation.

In rule (Tr2-Gram) for grammars, we prepare a start symbol S' and add the rules $S' \to S_{\mathtt{o}_\epsilon}, S' \to S_{\mathtt{o}_+}$. We remark that the rewriting rule for $S_{\mathtt{o}_\epsilon}$ (resp. $S_{\mathtt{o}_+}$) is generated only if the original grammar generates an empty (resp. non-empty) tree. For example, in the extreme case where $\mathcal{R} = \{S \to S\}$, we have $\mathcal{R}' = \{S' \to S_{\mathtt{o}_\epsilon}, S' \to S_{\mathtt{o}_+}\}$, without any rules to rewrite $S_{\mathtt{o}_\epsilon}$ or $S_{\mathtt{o}_+}$.

▶ **Example 8.** Let us consider the grammar $\mathcal{G}_3 = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ where $\mathcal{N} = \{S : \mathtt{o}, A : \mathtt{o} \to \mathtt{o}, B : \mathtt{o} \to \mathtt{o}, F : \mathtt{o} \to \mathtt{o}\}$, and $\mathcal{R}$ consists of:

$$S \to F\,\mathtt{a} \qquad S \to F\,\mathtt{b} \qquad A\,f \to \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\,f\,\mathtt{e}) \qquad B\,f \to \mathtt{br}\,\mathtt{b}\,(\mathtt{br}\,f\,\mathtt{e})$$
$$F\,f \to \mathtt{br}\,f\,(\mathtt{br}\,f\,\mathtt{e}) \qquad F\,f \to F(A\,f) \qquad F\,f \to F(B\,f)$$

It is the same as the grammar obtained in Example 6, except that redundant subscripts on non-terminals and variables have been removed. The body of the rule for A is transformed as follows.

$$\dfrac{\dfrac{}{f : \mathtt{o}_+ \vdash \mathtt{a} : \mathtt{o}_+ \Rightarrow \mathtt{a}}\ \textsc{Const1} \qquad \dfrac{\dfrac{\dfrac{}{f : \mathtt{o}_+ \vdash f : \mathtt{o}_+ \Rightarrow f_{\mathtt{o}_+}}\ \textsc{Var} \quad \dfrac{}{f : \mathtt{o}_+ \vdash \mathtt{e} : \mathtt{o}_\epsilon \Rightarrow \mathtt{e}}\ \textsc{Const0}}{f : \mathtt{o}_+ \vdash \mathtt{br}\,f\,\mathtt{e} : \mathtt{o}_+ \Rightarrow f_{\mathtt{o}_+}}\ \textsc{Const2}}{f : \mathtt{o}_+ \vdash \mathtt{br}\,\mathtt{a}\,(\mathtt{br}\,f\,\mathtt{e}) : \mathtt{o}_+ \Rightarrow \mathtt{br}\,\mathtt{a}\,f_{\mathtt{o}_+}}\ \textsc{Const2}}{\emptyset \vdash \lambda f.\mathtt{br}\,\mathtt{a}\,(\mathtt{br}\,f\,\mathtt{e}) : \mathtt{o}_+ \to \mathtt{o}_+ \Rightarrow \lambda f_{\mathtt{o}_+}.\mathtt{br}\,\mathtt{a}\,f_{\mathtt{o}_+}}\ \textsc{Abs}$$

The whole rules are transformed to:

$$S' \to S_{\mathtt{o}_+} \qquad S' \to S_{\mathtt{o}_\epsilon} \qquad S_{\mathtt{o}_+} \to F_{\mathtt{o}_+ \to \mathtt{o}_+}\,\mathtt{a} \qquad S_{\mathtt{o}_+} \to F_{\mathtt{o}_+ \to \mathtt{o}_+}\,\mathtt{b}$$
$$A_{\mathtt{o}_+ \to \mathtt{o}_+}\,f_{\mathtt{o}_+} \to \mathtt{br}\,\mathtt{a}\,f_{\mathtt{o}_+} \qquad B_{\mathtt{o}_+ \to \mathtt{o}_+}\,f_{\mathtt{o}_+} \to \mathtt{br}\,\mathtt{b}\,f_{\mathtt{o}_+} \qquad F_{\mathtt{o}_+ \to \mathtt{o}_+}\,f_{\mathtt{o}_+} \to \mathtt{br}\,f_{\mathtt{o}_+}\,f_{\mathtt{o}_+}$$
$$F_{\mathtt{o}_+ \to \mathtt{o}_+}\,f_{\mathtt{o}_+} \to F_{\mathtt{o}_+ \to \mathtt{o}_+}(A_{\mathtt{o}_+ \to \mathtt{o}_+}\,f_{\mathtt{o}_+}) \qquad F_{\mathtt{o}_+ \to \mathtt{o}_+}\,f_{\mathtt{o}_+} \to F_{\mathtt{o}_+ \to \mathtt{o}_+}(B_{\mathtt{o}_+ \to \mathtt{o}_+}\,f_{\mathtt{o}_+})$$

Here, we have omitted rules on non-terminals unreachable from S'.

The following theorem claims the correctness of the transformation. The proof is given in [2]. The main theorem (Theorem 4) follows from Theorems 7, 9, and the fact that any order-m grammar with $m < n$ can be converted to an order-n grammar by adding a dummy non-terminal of order n.

▶ **Theorem 9.** *Let $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$ be an order-n tree grammar. If $\mathcal{G} \Rightarrow \mathcal{G}'$, then $\mathcal{G}'$ is a tree grammar of order at most n, and $\mathcal{L}_{\mathtt{leaf}}(\mathcal{G})\uparrow_{\mathtt{e}} = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$.*

5 Applications

5.1 Unsafe order-2 word languages = safe order-2 word languages

As mentioned in Section 1, many of the earlier results on higher-order grammars [6, 10] were for the subclass called *safe* higher-order grammars. In safe grammars, the (simple) types of terms are restricted to *homogeneous types* [6] of the form $\kappa_1 \to \cdots \to \kappa_k \to \mathtt{o}$, where $\mathtt{order}(\kappa_1) \geq \cdots \geq \mathtt{order}(\kappa_k)$, and arguments of the same order must be supplied simultaneously. For example, if A has type $(\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}) \to \mathtt{o}$, then the term $f\,(A\,f\,f)$ where $f : \mathtt{o} \to \mathtt{o}$ is valid, but $g\,(A\,f)$ where $g : ((\mathtt{o} \to \mathtt{o}) \to \mathtt{o}) \to \mathtt{o}, f : \mathtt{o} \to \mathtt{o}$ is not: the partial application $A\,f$ is disallowed, since A expects another order-1 argument. *Unsafe* grammars (which are just called higher-order grammars in the present paper) are higher-order grammars without the safety restriction.

For order-2 word languages, Aehlig et al. [1] have shown that safety is not a genuine restriction. Our result in the present paper provides an alternative, short proof. Given an unsafe order-2 word grammar $\mathcal{G}$, we can obtain an equivalent order-1 grammar $\mathcal{G}'$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$. Note that $\mathcal{G}'$ is necessarily safe, since it is order-1 and hence there are no partial applications. Now, apply the backward transformation sketched in Section 2 to obtain an order-2 word grammar $\mathcal{G}''$ such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}'') = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$. By the construction of the backward transformation, $\mathcal{G}''$ is clearly a safe grammar: Since the type of each term occurring in $\mathcal{G}'$ is $\mathtt{o} \to \cdots \to \mathtt{o} \to \mathtt{o}$, the type of the corresponding term of $\mathcal{G}''$ is $(\mathtt{o} \to \mathtt{o}) \to \cdots \to (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o})$. Since all the arguments of type $\mathtt{o}$ are applied simultaneously in $\mathcal{G}'$, all the arguments of type $\mathtt{o} \to \mathtt{o}$ are also applied simultaneously in $\mathcal{G}''$. Thus, for any unsafe order-2 word grammar, there exists an equivalent safe order-2 word grammar.

5.2 Diagonal problem

The diagonal problem [5] asks, given a (word or tree) language L and a set S of symbols, whether for all n, there exists $w_n \in L$ such that $\forall a \in S. |w_n|_a \geq n$. Here, $|w|_a$ denotes the number of occurrences of a in w. A decision algorithm for the diagonal problem can be used for computing downward closures [21], which in turn have applications to program verification. Hague et al. [9] recently showed that the diagonal problem is decidable for safe higher-order word languages, and Clemente et al. [4] extended the result for unsafe tree languages. For the single letter case of the diagonal problem (where $|S| = 1$), we can obtain an alternative proof as follows. First, following the approach of Hague et al. [9], we can use logical reflection to reduce the single letter diagonal problem for an unsafe order-n tree language to that for the path language of an unsafe order-n tree language. We can then use our transformation to reduce the latter to the single letter diagonal problem for an unsafe order-$(n-1)$ tree language.

5.3 Context-sensitivity of order-3 word languages

By using the result of this paper and the context-sensitivity of order-2 tree languages [13], we can prove that any order-3 word language is context-sensitive, i.e., the membership problem for an order-3 word language can be decided in non-deterministic linear space. Given an order-3 word grammar $\mathcal{G}$, we first construct a corresponding order-2 tree grammar $\mathcal{G}'$ in

advance. Given a word w, we can construct a tree π whose frontier word is w one by one, and check whether $\pi \in \mathcal{L}(\mathcal{G}')$. Since the size of π is linearly bounded by the length $|w|$ of w, $\pi \stackrel{?}{\in} \mathcal{L}(\mathcal{G}')$ can be checked in space linear with respect to $|w|$. Thus, $w \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$ can be decided in non-deterministic linear space (with respect to the size of w).

6 Related Work

As already mentioned in Section 1, higher-order grammars have been extensively studied in 1980's [6, 7, 8], but most of those results have been for safe grammars. In particular, Damm [6] has shown an analogous result for safe grammars, but his proof does not extend to the unsafe case.

As also mentioned in Section 1, intersection types have been used in recent studies of (unsafe) higher-order grammars. In particular, type-based transformations of grammars and λ-terms have been studied in [14, 13, 4]. Clement et al. [4], independently from ours, gave a transformation from an order-$(n+1)$ "narrow" tree language (which subsumes a word language as a special case) to an order-n tree language; this transformation preserves the number of occurrences of each symbol in each tree. When restricted to word languages, our result is stronger in that our transformation is guaranteed to preserve the order of symbols as well, and does not add any additional leaf symbols (though they are introduced in the intermediate step); consequently, our proofs are more involved. They use different intersection types, but the overall effect of their transformation seems similar to that of our first transformation. Thus, it may actually be the case that their transformation also preserves the order of symbols, although they have not proved so.

7 Conclusion

We have shown that for any unsafe order-$(n+1)$ word grammar $\mathcal{G}$, there exists an unsafe order-n tree grammar $\mathcal{G}'$ whose frontier language coincides with the word language $\mathcal{L}_{\mathtt{w}}(\mathcal{G})$. The proof is constructive in that we provided (two-step) transformations that indeed construct $\mathcal{G}'$ from $\mathcal{G}$. The transformations are based on a combination of linear/non-linear intersection types, which may be interesting in its own right. As Damm [6] suggested, we expect the result to be useful for further studies of higher-order languages; in fact, we have discussed a few applications of the result.

Acknowledgments. We would like to thank Takeshi Tsukada for helpful discussions and thank Pawel Parys for information about the related work [4].

References

1 Klaus Aehlig, Jolie G. de Miranda, and C.-H. Luke Ong. Safety is not a restriction at level 2 for string languages. In *Proceedings of FoSSaCS 2005*, volume 3441 of *LNCS*, pages 490–504. Springer, 2005.

2 Kazuyuki Asada and Naoki Kobayashi. On word and frontier languages of unsafe higher-order grammars. *CoRR*, abs/1604.01595, 2016.

3 William Blum and C.-H. Luke Ong. The safe lambda calculus. *Logical Methods in Computer Science*, 5(1), 2009.

4 Lorenzo Clemente, Pawel Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recusion schemes is decidable. In *Proceedings of LICS 2016*, 2016.

5 Wojciech Czerwinski and Wim Martens. A note on decidable separability by piecewise testable languages. *CoRR*, abs/1410.1042, 2014. URL: `http://arxiv.org/abs/1410.1042`.

6 Werner Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982.

7 Joost Engelfriet. Iterated stack automata and complexity classes. *Info. Comput.*, 95(1):21–75, 1991.

8 Joost Engelfriet and Heiko Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Inf.*, 26(1/2):131–192, 1988.

9 Matthew Hague, Jonathan Kochems, and C.-H. Luke Ong. Unboundedness and downward closures of higher-order pushdown automata. In *Proceedings of POPL 2016*, pages 151–163, 2016. `doi:10.1145/2837614.2837627`.

10 Teodor Knapik, Damian Niwinski, and Pawel Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *TLCA 2001*, volume 2044 of *LNCS*, pages 253–267. Springer, 2001.

11 Naoki Kobayashi. Model checking higher-order programs. *Journal of the ACM*, 60(3), 2013.

12 Naoki Kobayashi. Pumping by typing. In *Proceedings of LICS 2013*, pages 398–407. IEEE Computer Society, 2013.

13 Naoki Kobayashi, Kazuhiro Inaba, and Takeshi Tsukada. Unsafe order-2 tree languages are context-sensitive. In *Proceedings of FoSSaCS 2014*, volume 8412 of *LNCS*, pages 149–163. Springer, 2014.

14 Naoki Kobayashi, Kazutaka Matsuda, Ayumi Shinohara, and Kazuya Yaguchi. Functional programs as compressed data. *Higher-Order and Symbolic Computation*, 2013.

15 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of LICS 2009*, pages 179–188. IEEE Computer Society Press, 2009.

16 Gregory M. Kobele and Sylvain Salvati. The IO and OI hierarchies revisited. *Inf. Comput.*, 243:205–221, 2015.

17 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS 2006*, pages 81–90. IEEE Computer Society Press, 2006.

18 Pawel Parys. How many numbers can a lambda-term contain? In *Proceedings of FLOPS 2014*, volume 8475 of *LNCS*, pages 302–318. Springer, 2014. `doi:10.1007/978-3-319-07151-0_19`.

19 Sylvain Salvati and Igor Walukiewicz. Typing weak MSOL properties. In Andrew M. Pitts, editor, *Proceedings of FoSSaCS 2015*, volume 9034 of *LNCS*, pages 343–357. Springer, 2015. `doi:10.1007/978-3-662-46678-0_22`.

20 Takeshi Tsukada and C.-H. Luke Ong. Compositional higher-order model checking via ω-regular games over böhm trees. In *Proceedings of CSL-LICS'14*, pages 78:1–78:10. ACM, 2014. `doi:10.1145/2603088.2603133`.

21 Georg Zetzsche. An approach to computing downward closures. In *Proceedings of ICALP 2015*, volume 9135 of *LNCS*, pages 440–451. Springer, 2015. `doi:10.1007/978-3-662-47666-6_35`.

The Schützenberger Product for Syntactic Spaces*†

Mai Gehrke[1], Daniela Petrişan[2], and Luca Reggio[3]

1 **IRIF, CNRS and Univ. Paris Diderot, Paris, France**
`mgehrke@liafa.univ-paris-diderot.fr`
2 **IRIF, CNRS and Univ. Paris Diderot, Paris, France**
`petrisan@liafa.univ-paris-diderot.fr`
3 **IRIF, CNRS and Univ. Paris Diderot, Paris, France**
`reggio@liafa.univ-paris-diderot.fr`

Abstract

Starting from Boolean algebras of languages closed under quotients and using duality theoretic insights, we derive the notion of *Boolean spaces with internal monoids* as recognisers for arbitrary formal languages of finite words over finite alphabets. This leads to recognisers and syntactic spaces in a setting that is well-suited for applying tools from Stone duality as applied in semantics.

The main focus of the paper is the development of topo-algebraic constructions pertinent to the treatment of languages given by logic formulas. In particular, using the standard semantic view of quantification as projection, we derive a notion of *Schützenberger product* for Boolean spaces with internal monoids. This makes heavy use of the Vietoris construction – and its dual functor – which is central to the coalgebraic treatment of classical modal logic.

We show that the unary Schützenberger product for spaces yields a recogniser for the language of all models of the formula $\exists x.\Phi(x)$, when applied to a recogniser for the language of all models of $\Phi(x)$. Further, we generalise global and local versions of the theorems of Schützenberger and Reutenauer characterising the languages recognised by the binary Schützenberger product.

Finally, we provide an equational characterisation of Boolean algebras obtained by local Schützenberger product with the one element space based on an Egli-Milner type condition on generalised factorisations of ultrafilters on words.

1998 ACM Subject Classification F. Theory of Computation; F.1.1 Models of Computation, F.4.1 Mathematical Logic; F.4.3 Formal Languages

Keywords and phrases Stone duality and Stone-Čech compactification, Vietoris hyperspace construction, logic on words, algebraic language theory beyond the regular setting

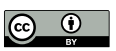
Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.112

1 Introduction

This contribution lies at the interface of two distinct areas: one in semantics concerned with modelling binding of variables, and the other in the theory of formal languages and the search for separation results for complexity classes based on a generalisation of the algebraic theory of regular languages [22, 12]. In semantics of propositional and modal logics, Stone duality and coalgebraic logic have had great success, but in the presence of quantifiers more

* A full version of this paper is available at `http://arxiv.org/abs/1603.08264`.

† This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No.670624).



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 112; pp. 112:1–112:14

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

general categorical semantics is required. Quantifiers change the set of free variables in a formula, leading to a notion of indexing formulas by their contexts of free variables. In the theory of regular languages, classes of models indexed by finite alphabets have long been studied in the form of varieties of languages [5]. There, one considers Boolean algebras of languages closed under quotients over a category of finite alphabets with monoid morphisms between the corresponding finitely generated monoids. This paper is intended as a first step towards establishing a connection between categorical semantics of logics and fibrational approaches in language theory.

We follow the line set by [7, 8] and [9], which exploits the connection between the algebraic theory of formal languages and Stone duality, see also [2, 1]. In this paper we are interested in the effect that first-order quantifiers have at the level of the algebraic recognisers. This is well understood in the regular case, where a plethora of powerful tools, in the form of Schützenberger, Maltsev, and block products of finite (and profinite) monoids, is used. Beyond the regular setting, we take as a departure point classes of languages equipped with actions of the free monoid over a finite set and the standard view of existential quantification as projection, and we derive – via Stone duality – our notion of recognisers and of unary Schützenberger product. Our analysis arrives at an extension of the Schützenberger product, which was originally introduced in [19] as a means of studying the concatenation product of regular languages and was further extended in [21] and [16] to arbitrary arity and to ordered monoids, respectively. Reutenauer [18], and Pin [15] in the ordered setting, have provided exact characterisations of the regular languages accepted by the Schützenberger product.

In the setting of regular languages, equations have played an essential rôle in providing decidability results for varieties of languages and various generalisations thereof. For classes of arbitrary languages decidability is not to be expected and separation of classes is the main focus. For this reason soundness becomes more important than completeness per se. However, complete axiomatisations are useful for obtaining decidability results for the class of regular languages within a fragment. See [9] for an example and for further motivation relative to the study of circuit complexity classes.

Contributions and Structure. After some preliminaries on Stone duality and actions by monoids, Section 3 introduces our notion of recognisers and main objects of study, the *Boolean spaces with internal monoids*. In Section 4 we analyse the relation between recognisers for a language L_Φ, corresponding to a formula Φ with one free first-order variable x, and recognisers for the existentially quantified language $L_{\exists x.\Phi}$. To this end, in Section 4.1 we introduce a unary version of the Schützenberger product, $\Diamond M$, for a discrete monoid M and prove that if M recognises L_Φ, then $\Diamond M$ recognises $L_{\exists x.\Phi}$. In Section 4.2 we extend the unary Schützenberger product, and the results in Section 4.1, to Boolean spaces with internal monoids (noting this can be done for semigroups as well). We end the section with a characterisation of the languages recognised by the unary Schützenberger product $(\Diamond X, \Diamond S)$ of a Boolean space with an internal semigroup (X, S) (see Theorem 14). In Section 5 we introduce the binary Schützenberger product of Boolean spaces with internal monoids. Theorems 16 and 18 extend results of Reutenauer in the regular setting and establish the connection with concatenation product for arbitrary languages. Finally, in Section 6 we provide a completeness result for the Boolean algebra recognised by the local version of the Schützenberger product of a space with the one element space.

2 Preliminaries

2.1 Stone duality for Boolean algebras

Let $(\mathcal{B}, \wedge, \vee, \neg, 0, 1)$ be a Boolean algebra. Recall that a subset $\mu \subseteq \mathcal{B}$ is a *filter* of $\mathcal{B}$ if it satisfies the following conditions:

- non-emptiness: $1 \in \mu$,
- upward closure: if $L \in \mu$ and $N \in \mathcal{B}$ satisfies $L \leq N$, then $N \in \mu$,
- closure under finite meets: if $L, N \in \mu$, then $L \wedge N \in \mu$.

A filter $\mu \subseteq \mathcal{B}$ is *proper* if $\mu \neq \mathcal{B}$. *Ultrafilters* are those for which $L \in \mu$ or $\neg L \in \mu$ for each $L \in \mathcal{B}$. In the Boolean algebra $\mathcal{P}(S)$, an example of an ultrafilter is given, for each $s \in S$, by the *principal ultrafilter* associated with the element s, namely

$$\uparrow s := \{b \in \mathcal{P}(S) \mid s \in b\}. \tag{1}$$

Let $X_{\mathcal{B}}$ be the collection of all the ultrafilters of $\mathcal{B}$. The fundamental insight of Stone is that, equipped with an appropriate topology, one may recover $\mathcal{B}$ from $X_{\mathcal{B}}$. For $L \in \mathcal{B}$ set

$$\widehat{L} := \{\mu \in X_{\mathcal{B}} \mid L \in \mu\}. \tag{2}$$

Then the family $\{\widehat{L} \mid L \in \mathcal{B}\}$ forms a basis of open sets for a topology σ on $X_{\mathcal{B}}$, and the topological space $(X_{\mathcal{B}}, \sigma)$ is called the *dual space* of the Boolean algebra $\mathcal{B}$. The topology σ is compact, Hausdorff, and admits a basis of *clopen* sets (i.e. sets that are both open and closed) since the complement of $\widehat{L}$ is $\widehat{\neg L}$. Compact Hausdorff spaces that admit a basis of clopen sets are known as *Boolean* (or *Stone*) *spaces*. The collection of clopens of a Boolean space X (equipped with set-theoretic operations) constitutes a Boolean algebra, known as the *dual algebra* of X. These processes are, up to natural equivalence, inverse to each other. Given a morphism of Boolean algebras $h\colon \mathcal{A} \to \mathcal{B}$, the inverse image map on their power sets $h^{-1}\colon \mathcal{P}(\mathcal{B}) \to \mathcal{P}(\mathcal{A})$ sends ultrafilters to ultrafilters and provides the continuous map from the dual space of $\mathcal{B}$ to the dual space of $\mathcal{A}$. Similarly, the inverse image map of a continuous map $f\colon X \to Y$ provides the morphism from the dual algebra of Y to that of X. In this correspondence, quotient algebras correspond to embeddings as (closed) subspaces, and inclusions as subalgebras correspond to quotient spaces. In category-theoretic terms, this establishes a contravariant equivalence between the category of Boolean spaces and continuous maps, and the category of Boolean algebras and their morphisms. This is the content of the celebrated Stone duality for Boolean algebras [20, Theorems 67 and 68].

We end this section with an example of a Boolean algebra and its dual space which will play a key rôle in the sequel. Let S be a set. Then $\mathcal{P}(S)$ is a Boolean algebra and its dual space, denoted by $\beta(S)$, is known as the *Stone-Čech compactification* of the set S. We remark that the map $\iota\colon S \to \beta(S)$, mapping an element s to the principal ultrafilter $\uparrow s$ of (1), is injective and embeds S, with the discrete topology, as a dense subspace of $\beta(S)$. Henceforth, we will consider S as a subspace of $\beta(S)$, identifying $s \in S$ with $\uparrow s$, thus suppressing the embedding ι. The space $\beta(S)$ is characterised by the following *universal property*: if X is a compact Hausdorff space and $f\colon S \to X$ is any function, then there is a (unique) continuous function $g\colon \beta(S) \to X$ such that the following diagram commutes.

$$\begin{array}{ccc} S & \hookrightarrow & \beta(S) \\ & \searrow^{f} & \downarrow{\scriptstyle g} \\ & & X \end{array} \tag{3}$$

Consequently, if T is a discrete space, any function $f\colon S \to T$ can be extended to a continuous map $\beta(f)\colon \beta(S) \to \beta(T)$. Explicitly, the latter is given, for each $\mu \in \beta(S)$ and $L \in \mathcal{P}(T)$, by

$$L \in \beta(f)(\mu) \quad \text{if, and only if,} \quad f^{-1}(L) \in \mu.$$

2.2 Monoid actions

Let $(M, \cdot, 1)$ be a monoid, and X be a set. A function $\lambda\colon M \times X \to X$ is called a *left action* of M on X provided

- for all $x \in X$, $\lambda(1, x) = x$,
- for all $m, m' \in M$ and $x \in X$, $\lambda(m \cdot m', x) = \lambda(m, \lambda(m', x))$.

Similarly, one can define a *right action* $\rho\colon X \times M \to X$ of M on X. For each $m \in M$, we refer to the function $\lambda_m\colon X \to X$ given by $\lambda_m(x) := \lambda(m, x)$ (respectively to the function $\rho_m\colon X \to X$ given by $\rho_m(x) := \rho(x, m)$) as the *component* of the action λ at m (respectively, of the action ρ at m). A pair consisting of left and right actions λ, ρ of M on X is said to be *compatible* if, for all $m, m' \in M$, $\lambda_m \circ \rho_{m'} = \rho_{m'} \circ \lambda_m$. We call such a pair of compatible actions a *biaction* of M on X (or an *M-biaction* on X).

▶ **Example 1.** Any monoid M can be seen as acting on itself on the left and on the right. The component of the left action at $m \in M$ is the multiplication on the left by m, and the component of the right action is the multiplication on the right by m. The compatibility of the two actions amounts precisely to the associativity of the monoid operation.

▶ **Example 2.** Consider $\mathbb{N}$, the free monoid on one generator. As observed in Example 1, for each $n \in \mathbb{N}$ we have components $\lambda_n, \rho_n\colon \mathbb{N} \to \mathbb{N}$ of compatible left and right actions of $\mathbb{N}$ on itself. By the universal property (3) of the Stone-Čech compactification, we obtain continuous components $\beta(\lambda_n), \beta(\rho_n)\colon \beta(\mathbb{N}) \to \beta(\mathbb{N})$ of a biaction of $\mathbb{N}$ on $\beta(\mathbb{N})$. However the set $\beta(\mathbb{N})$ is not equipped with a continuous monoid operation, see [11, Chapter 4].

3 Recognition by spaces with dense monoids

We start by showing how our main objects of study (see Definition 3 below) arise naturally by considering duals of Boolean algebras of languages closed under certain operations known as quotients by words. Let Σ be a finite alphabet. Instantiating the monoid in Example 1 with the free monoid Σ^* on Σ, we obtain a biaction of Σ^* on itself. The components of the left and right actions are given by concatenation, and they will be denoted by

$$\lambda_w\colon \Sigma^* \to \Sigma^*,\ u \mapsto wu \quad \text{and} \quad \rho_w\colon \Sigma^* \to \Sigma^*,\ u \mapsto uw.$$

By discrete duality, i.e. by applying the contravariant power set functor, we obtain right and left Σ^*-actions on $\mathcal{P}(\Sigma^*)$ given by $\lambda_w^{-1}\colon \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$, and respectively, by $\rho_w^{-1}\colon \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$. These are the well-known *left quotients* and *right quotients* of language theory given, respectively, by

$$L \mapsto \{u \mid wu \in L\} =: w^{-1}L \quad \text{and} \quad L \mapsto \{u \mid uw \in L\} =: Lw^{-1}.$$

It is immediate that the λ_w^{-1} and ρ_w^{-1} are homomorphisms and compatible Σ^*-actions.

Dualising again, we see that the space $\beta(\Sigma^*)$ is equipped with (compatible and continuous) left and right Σ^*-actions given, for all $w \in \Sigma^*$, by $\beta(\lambda_w)$ and $\beta(\rho_w)$, respectively. By abuse of notation and for ease of readability, we will denote these actions again by λ_w, respectively ρ_w. We notice that the pair $(\beta(\Sigma^*), \Sigma^*)$ exhibits the following structure:

- a Boolean space $\beta(\Sigma^*)$,
- a dense subspace Σ^* equipped with a monoid structure,
- a biaction of Σ^* on $\beta(\Sigma^*)$ with continuous components extending that of Σ^* on itself.

Now, consider a Boolean subalgebra $\mathcal{B}$ of $\mathcal{P}(\Sigma^*)$ closed under left and right quotients by words. Then the maps λ_w^{-1} and ρ_w^{-1} restrict to Boolean algebra morphisms on $\mathcal{B}$, yielding the following commutative diagrams.

$$\begin{array}{ccc} \mathcal{P}(\Sigma^*) & \xrightarrow{\lambda_w^{-1}} & \mathcal{P}(\Sigma^*) \\ \uparrow & & \uparrow \\ \mathcal{B} & \xrightarrow{\lambda_w^{-1}} & \mathcal{B} \end{array} \qquad \begin{array}{ccc} \mathcal{P}(\Sigma^*) & \xrightarrow{\rho_w^{-1}} & \mathcal{P}(\Sigma^*) \\ \uparrow & & \uparrow \\ \mathcal{B} & \xrightarrow{\rho_w^{-1}} & \mathcal{B} \end{array} \tag{4}$$

Let $X_{\mathcal{B}}$ denote the dual space of the Boolean algebra $\mathcal{B}$. The embedding $\mathcal{B} \hookrightarrow \mathcal{P}(\Sigma^*)$ dually corresponds to a quotient $\tau\colon \beta(\Sigma^*) \twoheadrightarrow X_{\mathcal{B}}$. The space $X_{\mathcal{B}}$ also admits left and right Σ^*-actions induced by the duals of the maps λ_w^{-1}, respectively ρ_w^{-1}, from (4). We thus obtain

$$\begin{array}{ccc} \beta(\Sigma^*) & \xrightarrow{\lambda_w} & \beta(\Sigma^*) \\ \tau\downarrow & & \downarrow\tau \\ X_{\mathcal{B}} & \xrightarrow{\lambda_w} & X_{\mathcal{B}} \end{array} \qquad \begin{array}{ccc} \beta(\Sigma^*) & \xrightarrow{\rho_w} & \beta(\Sigma^*) \\ \tau\downarrow & & \downarrow\tau \\ X_{\mathcal{B}} & \xrightarrow{\rho_w} & X_{\mathcal{B}} \end{array} \tag{5}$$

Then $M := \tau[\Sigma^*]$ is a dense subspace of $X_{\mathcal{B}}$, and we have the following commutative diagram.

$$\begin{array}{ccc} \beta(\Sigma^*) & \xrightarrow{\tau} & X_{\mathcal{B}} \\ \uparrow & & \uparrow \\ \Sigma^* & \xrightarrow{\tau} & M \end{array} \tag{6}$$

We observe that the pair $(X_{\mathcal{B}}, M)$ exhibits the same kind of structure as $(\beta(\Sigma^*), \Sigma^*)$:

- a Boolean space $X_{\mathcal{B}}$,
- a dense subspace M equipped with a monoid structure,
- a biaction of M on $X_{\mathcal{B}}$ with continuous components extending the biaction of M on itself.

Indeed, recall that $X_{\mathcal{B}}$ is equipped with left and right Σ^*-actions which are preserved by the map τ by commutativity of (5). The Σ^*-actions on $X_{\mathcal{B}}$ restrict to Σ^*-actions on M, which are preserved by the restriction of τ. The monoid structure on M is then defined as follows. For any $m \in M$ pick $w_m \in \Sigma^*$ satisfying $\tau(w_m) = m$. Such an element exists because M is the image of Σ^* by τ. For $m, m' \in M$, set $m \cdot m' := \lambda_{w_m}(m')$. It is easily seen that the latter operation is well-defined and provides a monoid structure on M which makes the restriction of τ a monoid morphism.

As first introduced in [8], we will be using dual spaces equipped with actions as recognisers. The examples above motivate the following definition.

▶ **Definition 3.** A *Boolean space with an internal monoid* is a pair (X, M) consisting of

- a Boolean space X,
- a dense subspace M equipped with a monoid structure,
- a biaction of M on X with continuous components extending the biaction of M on itself.

▶ Remark. The recognisers introduced in [8] are monoids equipped with a uniform space structure, namely the Pervin uniformity given by a Boolean algebra of subsets of the monoid,

so that the biaction of the monoid on itself has uniformly continuous components. Such an object was called a *semiuniform monoid*. One may show that the completion of a semiuniform monoid is a Boolean space with an internal monoid. Conversely, given a Boolean space with an internal monoid (X, M), the Pervin uniformity on M induced by the dual of X is a semiuniform monoid, and these two constructions are inverse to each other.

We are interested in maps between pairs (X, M) and (Y, N), i.e. continuous maps $X \to Y$ which preserve the additional structure.

▶ **Definition 4.** A *morphism* between two Boolean spaces with internal monoids (X, M) and (Y, N) is a continuous map $f\colon X \to Y$ such that f restricts to a monoid morphism $M \to N$.

Morphisms, as just defined, are in fact also biaction-preserving maps.

▶ **Lemma 5.** *Let $f\colon (X, M) \to (Y, N)$ be a morphism of Boolean spaces with internal monoids. Then f preserves the actions, i.e. for every $m \in M$*

$$f \circ \lambda_m = \lambda_{f(m)} \circ f \quad \text{and} \quad f \circ \rho_m = \rho_{f(m)} \circ f.$$

▶ **Example 6.** The map $\tau\colon (\beta(\Sigma^*), \Sigma^*) \to (X_{\mathcal{B}}, M)$ of (6) is a morphism of Boolean spaces with internal monoids.

▶ Remark. The map $L \mapsto \widehat{L}$ of (2) establishes a one-to-one correspondence between the elements of $\mathcal{P}(\Sigma^*)$ and the clopens of $\beta(\Sigma^*)$. Thus, we will sometimes blur the distinction between recognition of a language L and recognition of the corresponding clopen $\widehat{L}$.

▶ **Definition 7.** Let Σ be a finite alphabet, and let $L \subseteq \Sigma^*$ be a language. We say that L (or $\widehat{L}$) is *recognised by the morphism* $f\colon (\beta(\Sigma^*), \Sigma^*) \to (X, M)$ if there is a clopen $C \subseteq X$ such that $\widehat{L} = f^{-1}(C)$. Moreover, the language L is *recognised by the space* (X, M) if there is a morphism $(\beta(\Sigma^*), \Sigma^*) \to (X, M)$ recognising L. Similarly, we say that a morphism (or a space) recognises a Boolean algebra if it recognises all its elements.

▶ Remark. In general, a morphism $(\beta(\Sigma^*), \Sigma^*) \to (X, M)$ with *infinite* M, recognises (in the sense of Definition 7) far less languages than the induced monoid morphism $\Sigma^* \to M$. On the other hand, a finite monoid M may be seen as a space with an internal monoid, in which the space component is the monoid itself, equipped with the discrete topology. A morphism $(\beta(\Sigma^*), \Sigma^*) \to (M, M)$ yields in particular a monoid morphism $\Sigma^* \to M$. Conversely, a monoid morphism $h\colon \Sigma^* \to M$ extends uniquely to a continuous map $\beta h\colon \beta(\Sigma^*) \to M$ whose restriction to Σ^* is a monoid morphism. Thus the notion of recognition introduced here extends the usual notion for regular languages, but is finer-grained in the non-regular setting.

4 A unary variant of the Schützenberger product

4.1 Logical motivation: existentially quantified languages

Consider the free monoid Σ^* over a finite alphabet Σ. A word $w \in \Sigma^*$ may be seen as a structure based on the set $\{0, \dots, |w| - 1\}$,[1] equipped minimally with a unary predicate for each letter $a \in \Sigma$, which holds at i if and only if $w_i = a$. Now given a formula Φ (in a language interpretable over words as structures), assumed for simplicity to have only one free first-order variable x, we will see the set L_Φ of all words satisfying Φ as a language in the

[1] Here, as usual, $|w| \in \mathbb{N}$ denotes the length of the word $w = w_0 \cdots w_{|w|-1} \in \Sigma^*$.

extended alphabet $\Sigma \times 2$. In the terminology of [22], L_Φ consists of $\{x\}$-structures, which correspond to words in the subset $(\Sigma \times \{0\})^*(\Sigma \times \{1\})(\Sigma \times \{0\})^*$ of the free monoid $(\Sigma \times 2)^*$. An $\{x\}$-structure satisfies Φ provided the underlying word in the alphabet Σ satisfies Φ under the interpretation in which x points to the unique position marked with a 1. Notice that $(\Sigma \times \{0\})^*(\Sigma \times \{1\})(\Sigma \times \{0\})^*$ is isomorphic to the set $\Sigma^* \otimes \mathbb{N}$ of words in Σ^* with a marked spot defined by

$$\Sigma^* \otimes \mathbb{N} := \{(w, i) \in \Sigma^* \times \mathbb{N} \mid i < |w|\}.$$

Throughout this section we will make use of the following three maps

$$\gamma_0 \colon \Sigma^* \to (\Sigma \times 2)^*, \qquad \gamma_1 \colon \Sigma^* \otimes \mathbb{N} \to (\Sigma \times 2)^*, \qquad \pi \colon \Sigma^* \otimes \mathbb{N} \to \Sigma^*.$$

- The map $\gamma_0 \colon \Sigma^* \to (\Sigma \times 2)^*$ is the embedding given by $w \mapsto w^0$, where w^0 has the same length as w and

$$(w^0)_j := (w_j, 0) \quad \text{for each} \quad j < |w|.$$

- The map $\gamma_1 \colon \Sigma^* \otimes \mathbb{N} \to (\Sigma \times 2)^*$ is the embedding given by $(w, i) \mapsto w^{(i)}$, where $w^{(i)}$ has the same length as w and

$$(w^{(i)})_j := \begin{cases} (w_j, 0) & \text{if} \quad i \neq j < |w| \\ (w_i, 1) & \text{if} \quad i = j. \end{cases}$$

- The map $\pi \colon \Sigma^* \otimes \mathbb{N} \to \Sigma^*$ is the projection on the first coordinate.

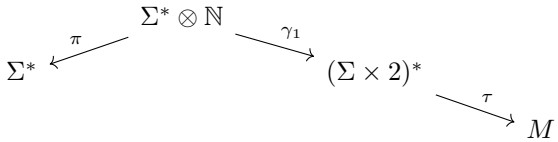
▶ Remark. The language $L_{\exists x.\Phi}$ is obtained as $\pi[\gamma_1^{-1}(L_\Phi)]$. More generally, given a language $L \subseteq (\Sigma \times 2)^*$, we shall denote $\pi[\gamma_1^{-1}(L)] \subseteq \Sigma^*$ by $L_\exists$.

▶ Remark. Notice that, unlike γ_0, the maps γ_1 and π are not monoid morphisms. Indeed, $\Sigma^* \otimes \mathbb{N}$ does not have a suitable monoid structure. However, $\Sigma^* \otimes \mathbb{N}$ does have a Σ^*-biaction structure. For $v \in \Sigma^*$, the components of the left and right actions are given by

$$\lambda_v(w, i) := (vw, i + |v|),$$
$$\rho_v(w, i) := (wv, i).$$

It is clear that both γ_1 and π preserve the Σ^*-actions.

Assume that the language L_Φ is recognised by a monoid morphism $\tau \colon (\Sigma \times 2)^* \to M$. We have the following pair of functions[2] with domain $\Sigma^* \otimes \mathbb{N}$



which gives rise to a relation $R \colon \Sigma^* \nrightarrow M$ given by

$$(w, m) \in R \quad \text{if, and only if,} \quad \exists (w, i) \in \pi^{-1}(w).\ (\tau \circ \gamma_1)(w, i) = m.$$

[2] Notice that this is not a relational morphism in the sense of Tilson's definition given in [5], since the domain $\Sigma^* \otimes \mathbb{N}$ does not have a compatible monoid structure.

Though π is not injective, it does have *finite preimages*. As will be crucial in what follows, this allows us to represent R as a function (which, in general, is not a monoid morphism)

$$\xi_1 \colon \Sigma^* \to \mathcal{P}_{fin}(M), \quad w \mapsto \{\tau(w^{(i)}) \mid 0 \leq i < |w|\} \tag{7}$$

where $\mathcal{P}_{fin}(M)$ denotes the set of finite subsets of M. Consider the monoid structure on $\mathcal{P}_{fin}(M)$ with union as the multiplication, and the empty set as unit. Notice that the monoid M acts on $\mathcal{P}_{fin}(M)$ both to the left and to the right, and the two actions are compatible. The left action $M \times \mathcal{P}_{fin}(M) \to \mathcal{P}_{fin}(M)$ is given, for $m \in M$ and $S \in \mathcal{P}_{fin}(M)$, by $m \cdot S := \{m \cdot s \mid s \in S\}$. Similarly, the right action is given by $S \cdot m := \{s \cdot m \mid s \in S\}$.

▶ **Definition 8.** We define the *unary Schützenberger product* $\Diamond M$ of M as the bilateral semidirect product $\mathcal{P}_{fin}(M) * M$ of the monoids $(\mathcal{P}_{fin}(M), \cup)$ and $(M, \cdot)$. Explicitly, the underlying set of this monoid is the Cartesian product $\mathcal{P}_{fin}(M) \times M$, and the multiplication $*$ on $\mathcal{P}_{fin}(M) * M$ is given by

$$(S, m) * (T, n) := (S \cdot n \cup m \cdot T, m \cdot n).$$

Note that the projection onto the second coordinate, $\pi_2 \colon \Diamond M \to M$, is a monoid morphism.

▶ **Proposition 9.** *If $\tau \colon (\Sigma \times 2)^* \to M$ is a monoid morphism recognising L_Φ, then there exists a monoid morphism*

$$\xi \colon \Sigma^* \to \Diamond M$$

that recognises the language $L_{\exists x.\Phi}$ and makes the following diagram commute.

$$\begin{array}{ccc} \Sigma^* & \xrightarrow{\ \xi\ } & \Diamond M \\ {\scriptstyle \gamma_0}\downarrow & & \downarrow{\scriptstyle \pi_2} \\ (\Sigma \times 2)^* & \xrightarrow{\ \tau\ } & M \end{array}$$

Proof idea. The map ξ is obtained by pairing $\xi_1 \colon \Sigma^* \to \mathcal{P}_{fin}(M)$ of (7) and $\tau \circ \gamma_0 \colon \Sigma^* \to M$. Explicitly,

$$w \mapsto (\{\tau(w^{(i)}) \mid 0 \leq i < |w|\}, \tau(w^0)).$$

One may show that the map ξ is a monoid morphism with respect to the concatenation on Σ^* and the multiplication $*$ on the semidirect product $\mathcal{P}_{fin}(M) * M$. Now let V be a subset of M such that $L_\Phi = \tau^{-1}(V)$, and consider the set $\diamond V \subseteq \mathcal{P}_{fin}(M)$ defined as $\{S \in \mathcal{P}_{fin}(M) \mid S \cap V \neq \emptyset\}$. Then $\xi^{-1}(\diamond V \times M)$ is precisely $L_{\exists x.\Phi}$. ◀

▶ Remark. In [21] Straubing generalised the Schützenberger product for any finite number of monoids. Using his construction, the unary Schützenberger product of M is simply M, and hence is different from $\Diamond M$ introduced above. For the connection between closure under concatenation product and first-order quantification in the regular setting, see [14].

▶ Remark. For lack of space, we have chosen to just 'pull Definition 8 (and consequently also the upcoming Definition 11) out of a hat'. However, by a careful analysis of how quotients in $\mathcal{P}(\Sigma^*)$ of languages $L_\exists$ are calculated, relative to corresponding calculations in $\mathcal{P}((\Sigma \times 2)^*)$, one may simply derive by duality that the operation given here is the right one.

4.2 The Schützenberger product for one space $\Diamond X$

In this section we assume that the language $L_\Phi \subseteq (\Sigma \times 2)^*$ is recognised by a morphism of Boolean spaces with internal monoids $\tau\colon (\beta(\Sigma \times 2)^*, (\Sigma \times 2)^*) \to (X, M)$. Notice that in this case we have a pair of continuous maps

$$\begin{array}{ccccc} & & \beta(\Sigma^* \otimes \mathbb{N}) & & \\ & \overset{\beta\pi}{\swarrow} & & \overset{\beta\gamma_1}{\searrow} & \\ \beta(\Sigma^*) & & & & \beta(\Sigma \times 2)^* \\ & & & & \overset{\tau}{\searrow} \\ & & & & X \end{array} \tag{8}$$

which, as before, yields a relation $\beta(\Sigma^*) \nrightarrow X$. We would like to describe this relation as a continuous map on $\beta(\Sigma^*)$. To this end, we need an analogue for spaces of the finite power set construction. This is provided by the *Vietoris space construction* (see [10, Section B.1] for further details).

▶ **Definition 10.** Let X be a Boolean space. The *Vietoris space* $\mathcal{V}(X)$ is the Boolean space with underlying set $\{K \subseteq X \mid K \text{ is closed in } X\}$, and topology generated by the subbasis consisting of the sets, for V clopen in X, of the form

$$\Box V := \{K \in \mathcal{V}(X) \mid K \subseteq V\} \quad \text{and} \quad \Diamond V := \{K \in \mathcal{V}(X) \mid K \cap V \neq \emptyset\}.$$

Just as in the monoid case, diagram (8) yields a map

$$\xi_1\colon \beta(\Sigma^*) \to \mathcal{V}(X)$$

defined as the composition $\tau \circ \beta\gamma_1 \circ (\beta\pi)^{-1}$, or equivalently as the unique continuous extension of the map $\xi_1\colon \Sigma^* \to \mathcal{P}_{fin}(M)$ defined in (7).

▶ **Definition 11.** We define the *unary Schützenberger product* of a Boolean space with an internal monoid (X, M) as the pair $(\Diamond X, \Diamond M)$, where $\Diamond X$ is the space $\mathcal{V}(X) \times X$ equipped with the product topology and $\Diamond M$ is as in Definition 8.

▶ **Lemma 12.** *The unary Schützenberger product $(\Diamond X, \Diamond M)$ of (X, M) is a Boolean space with an internal monoid.*

Proof Idea. Recall that M is a dense subspace of X. It follows by [13, Theorem 4, p. 163] that $\mathcal{P}_{fin}(M)$ is a dense subspace of $\mathcal{V}(X)$. Thus the monoid $\Diamond M$ is a dense subspace of $\Diamond X$. Next we define the actions of $\Diamond M$ on $\Diamond X$ as follows:

$$l_{(S,m)}(T, x) := (\{\lambda_s(x) \mid s \in S\} \cup \lambda_m[T], \lambda_m(x)),$$
$$r_{(S,m)}(T, x) := (\{\rho_s(x) \mid s \in S\} \cup \rho_m[T], \rho_m(x)).$$

It is not difficult to see that the above maps are the unique continuous extensions to $\Diamond X$ of the multiplication by (S, m), to the left and to the right, on $\Diamond M$. ◀

The projection $\pi_2\colon \Diamond X \to X$ is a morphism of Boolean spaces with internal monoids.

▶ **Proposition 13.** *If $\tau\colon (\beta(\Sigma \times 2)^*, (\Sigma \times 2)^*) \to (X, M)$ is a morphism of Boolean spaces with internal monoids recognising L_Φ, then there is a morphism $\xi\colon (\beta(\Sigma^*), \Sigma^*) \to (\Diamond X, \Diamond M)$ recognising $L_{\exists x.\Phi}$ and such that the following diagram commutes.*

$$\begin{array}{ccc} \beta(\Sigma^*) & \xrightarrow{\xi} & \Diamond X \\ {\scriptstyle \beta\gamma_0}\downarrow & & \downarrow{\scriptstyle \pi_2} \\ \beta(\Sigma \times 2)^* & \xrightarrow{\tau} & X \end{array}$$

All the constructions introduced so far can be carried out for semigroups. In particular, we can consider Boolean spaces with internal semigroups as recognisers of languages in $\mathcal{P}(\Sigma^+)$. Along the lines of Definition 8, we introduce the unary Schützenberger product $\Diamond S$ of a semigroup S as the bilateral semidirect product of the semigroups $(\mathcal{P}^+_{fin}(S), \cup)$ and $(S, \cdot)$, where $\mathcal{P}^+_{fin}(S)$ denotes the family of finite non-empty subsets of S. Similarly, at the level of spaces, in the Vietoris construction we will consider only non-empty closed subsets.

Now, write $\mathcal{B}(X, \Sigma)$ for the Boolean algebra of languages in $\mathcal{P}(\Sigma^+)$ recognised by the Boolean space with an internal semigroup (X, S), and note that the latter Boolean algebra is always closed under quotients. Moreover, given a language $L \subseteq (\Sigma \times 2)^+$, recall that $L_\exists$ denotes the language $\pi[\gamma_1^{-1}(L)]$.

▶ **Theorem 14.** *Let (X, S) be a Boolean space with an internal semigroup, and let $\mathcal{B}(X, \Sigma{\times}2)_\exists$ denote the Boolean subalgebra closed under quotients of $\mathcal{P}(\Sigma^+)$ generated by the family $\{L_\exists \mid L \in \mathcal{B}(X, \Sigma \times 2)\}$. Then $\mathcal{B}(\Diamond X, \Sigma)$ coincides with the Boolean algebra generated by the union of $\mathcal{B}(X, \Sigma)$ and $\mathcal{B}(X, \Sigma \times 2)_\exists$.*

The proof of this theorem hinges on the fact that the first components of the recognising morphisms evaluate to non-empty subsets. An analogous statement can be formulated for monoids, but we would have to restrict the recognising morphisms when defining $\mathcal{B}(\Diamond X, \Sigma)$.

5 A variant of the Schützenberger product for two spaces

Given two monoids $(M, \cdot), (N, \cdot)$, the Schützenberger product $\Diamond(M, N)$ can be defined as the monoid $\mathcal{P}_{fin}(M \times N) \times M \times N$ whose operation is given by

$$(S, m_1, n_1) \cdot (T, m_2, n_2) := (m_1 \cdot T \cup S \cdot n_2, m_1 \cdot m_2, n_1 \cdot n_2).$$

Now, consider two Boolean spaces with internal monoids (X, M) and (Y, N). We define the space $\Diamond(X, Y)$ as the product $\mathcal{V}(X \times Y) \times X \times Y$. It is clear that the monoid $\Diamond(M, N)$ is dense in $\Diamond(X, Y)$. Moreover, the left action of $\Diamond(M, N)$ on itself can be extended to $\Diamond(X, Y)$ by setting, for any $(S, m_1, n_1) \in \Diamond(M, N)$,

$$\lambda_{(S,m_1,n_1)} \colon \Diamond(X, Y) \to \Diamond(X, Y),\ (Z, x, y) \mapsto (m_1 Z \cup Sy, \lambda_{m_1}(x), \lambda_{n_1}(y)), \tag{9}$$

where

$$m_1 Z := \{(\lambda_{m_1}(x), y) \in X \times Y \mid (x, y) \in Z\} \quad \text{and} \quad Sy := \{(m, \lambda_n(y)) \in X \times Y \mid (m, n) \in S\}.$$

Similarly, the right action can be defined by

$$\rho_{(S,m_1,n_1)} \colon \Diamond(X, Y) \to \Diamond(X, Y),\ (Z, x, y) \mapsto (Z n_1 \cup xS, \rho_{m_1}(x), \rho_{n_1}(y)), \tag{10}$$

where

$$Z n_1 := \{(x, \rho_{n_1}(y)) \in X \times Y \mid (x, y) \in Z\} \quad \text{and} \quad xS := \{(\rho_m(x), n) \in X \times Y \mid (m, n) \in S\}.$$

It is easy to see that we obtain a biaction of $\Diamond(M, N)$ on $\Diamond(X, Y)$. Furthermore,

▶ **Lemma 15.** *The biaction of $\Diamond(M, N)$ on $\Diamond(X, Y)$ defined in (9) and (10) has continuous components. Thus $(\Diamond(X, Y), \Diamond(M, N))$ is a Boolean space with an internal monoid.*

The next three results establish the connection between concatenation of possibly non-regular languages and the Schützenberger product of Boolean spaces with internal monoids. We thus extend the theorems of Schützenberger [19] and Reutenauer [18].

▶ **Theorem 16** (Reutenauer's theorem, global version). *Consider Boolean spaces with internal monoids (X, M) and (Y, N). Let $\mathcal{L}$ be the Boolean algebra generated by all the Σ^*-languages of the form L_1, L_2 and $L_1 a L_2$, where L_1 (respectively L_2) is recognised by X (respectively Y) and $a \in \Sigma$. Then a Σ^*-language is recognised by $X \Diamond Y$ if, and only if, it belongs to $\mathcal{L}$.*

Proof Idea. Suppose the languages L_1, L_2 are recognised by morphisms $\phi_1 \colon (\beta(\Sigma^*), \Sigma^*) \to (X, M)$ and $\phi_2 \colon (\beta(\Sigma^*), \Sigma^*) \to (Y, N)$, respectively, and fix $a \in \Sigma$. By abuse of notation, call $\phi_1 \times \phi_2 \colon \beta(\Sigma^* \times \{a\} \times \Sigma^*) \to X \times Y$ the unique continuous extension of the product map $\Sigma^* \times \{a\} \times \Sigma^* \to X \times Y$ whose components are $(w, a, w') \mapsto \phi_1(w)$ and $(w, a, w') \mapsto \phi_2(w')$. Let $\zeta_a \colon \beta(\Sigma^*) \to \mathcal{V}(X \times Y)$ be the continuous function induced by the diagram

$$\begin{array}{ccc} & \beta(\Sigma^* \times \{a\} \times \Sigma^*) & \\ {\scriptstyle \beta c}\swarrow & & \searrow{\scriptstyle \phi_1 \times \phi_2} \\ \beta(\Sigma^*) & & X \times Y \end{array} \tag{11}$$

just as for diagram (8), where $c \colon \Sigma^* \times \{a\} \times \Sigma^* \to \Sigma^*$ is the concatenation map $(w, a, w') \mapsto waw'$. One can prove that the map ζ_a is a morphism recognising L_1, L_2 and $L_1 a L_2$.

Conversely, for any morphism $\langle \zeta, \phi_1, \phi_2 \rangle \colon (\beta(\Sigma^*), \Sigma^*) \to (X \Diamond Y, M \Diamond N)$ and clopens $C_1 \subseteq X$, $C_2 \subseteq Y$, we must prove that $\zeta^{-1}(\Diamond(C_1 \times C_2)) \cap \Sigma^* \in \mathcal{L}$. One observes that each

$$L_{C_1 \times C_2, a} := \{w \in \Sigma^* \mid \exists u, v \in \Sigma^* \text{ s.t. } w = uav \text{ and } \phi_1(u)\zeta(a)\phi_2(v) \in \Diamond(C_1 \times C_2)\}$$

is in the Boolean algebra $\mathcal{L}$. Then $\zeta^{-1}(\Diamond(C_1 \times C_2)) \cap \Sigma^* = \bigcup_{a \in \Sigma} L_{C_1 \times C_2, a}$. ◀

The next corollary follows at once by Theorem 16, by noting that $L_1 L_2 = \bigcup_{a \in \Sigma} L_1 a (a^{-1} L_2)$ whenever L_2 does not contain the empty word and $L_1 L_2 = \bigcup_{a \in \Sigma} L_1 a (a^{-1} L_2) \cup L_1$ otherwise.

▶ **Corollary 17.** *The Boolean space with an internal monoid $(\Diamond(X, Y), \Diamond(M, N))$ recognises the concatention $L_1 L_2$ of languages L_1, L_2 recognised by (X, M) and (Y, N), respectively.*

Finally, the following local statement is a direct consequence of the proof of Theorem 16.

▶ **Theorem 18** (Reutenauer's theorem, local version). *Consider morphisms $\phi_1 \colon (\beta(\Sigma^*), \Sigma^*) \to (X, M)$ and $\phi_2 \colon (\beta(\Sigma^*), \Sigma^*) \to (Y, N)$. Let $\mathcal{L}$ be the Boolean algebra generated by all the Σ^*-languages of the form L_1, L_2 and $L_1 a L_2$, where L_1 (respectively L_2) is recognised by ϕ_1 (respectively ϕ_2) and $a \in \Sigma$. Then a Σ^*-language is recognised by the morphism*

$$\langle \langle \zeta_a \rangle_{a \in \Sigma}, \phi_1, \phi_2 \rangle \colon \beta(\Sigma^*) \to \mathcal{V}(X \times Y)^{\Sigma} \times X \times Y$$

where $\zeta_a \colon \beta(\Sigma^*) \to \mathcal{V}(X \times Y)$ is induced by diagram (11) if, and only if, it belongs to $\mathcal{L}$.

6 Ultrafilter equations

Identifying simple equational bases for the Boolean algebras of languages recognised by Schützenberger products, in terms of the equational theories of the input Boolean algebras, is an important step in studying classes built up by repeated application of quantification or language concatenation. See e.g. [17, 3] for examples of such work in the regular setting.

As a proof-of-concept and first step, we provide a fairly easy completeness result for the Boolean algebra recognised by the local version of a Schützenberger product of a space with the one element space. First we introduce notation for the dual construction, see Theorem 18.

▶ **Definition 19.** Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be Boolean algebras of Σ^*-languages closed under quotients. We define the *binary Schützenberger sum* of $\mathcal{B}_1$ and $\mathcal{B}_2$ to be the Boolean algebra of languages

$$\mathcal{B}_1 \Diamond\!\!\!\!\!+ \mathcal{B}_2 := \langle \mathcal{B}_1 \cup \mathcal{B}_2 \cup \{L_1 a L_2 \mid L_1 \in \mathcal{B}_1,\ L_2 \in \mathcal{B}_2,\ a \in \Sigma\}\rangle.$$

Note that this Boolean algebra is also closed under quotients.

Let $\mathcal{B} \subseteq \mathcal{P}(\Sigma^*)$ be a Boolean algebra closed under quotients. We give equations for $\mathcal{B}\Diamond\!\!\!\!\!+2$. Recall that an equation for a Boolean subalgebra of $\mathcal{P}(\Sigma^*)$ is a pair $\mu \approx \nu$, where $\mu, \nu \in \beta(\Sigma^*)$, and that $L \in \mathcal{P}(\Sigma^*)$ *satisfies the ultrafilter equation* $\mu \approx \nu$ provided

$$L \in \mu \quad \text{if, and only if,} \quad L \in \nu.$$

A Boolean subalgebra of $\mathcal{P}(\Sigma^*)$ satisfies an ultrafilter equation provided each of its elements satisfies it. For background and more details on equations see e.g. [7, 9, 6]. Now, set

$$f_a \colon \Sigma^* \otimes \mathbb{N} \to \Sigma^*,\ (w,i) \mapsto w(a@i) \quad \text{and} \quad f_r \colon \Sigma^* \otimes \mathbb{N} \to \Sigma^*,\ (w,i) \mapsto w_{|i} = w_0 \cdots w_{i-1}$$

where $a \in \Sigma$ and $w(a@i)$ denotes the word obtained by replacing the ith letter of the word $w = w_0 \cdots w_{|w|-1}$ by an a.

The intuition is that the extension βf_a will allow us to *factor* an ultrafilter at an occurrence of the letter a, whereas the extension βf_r gives us access to the prefix of this factorisation.

▶ **Definition 20.** Let $\mathcal{E}(\mathcal{B}\Diamond\!\!\!\!\!+2)$ denote the set of all equations $\mu \approx \nu$ so that

- $\mu \approx \nu$ holds in $\mathcal{B}$;
- for each $\gamma \in \beta(\Sigma^* \otimes \mathbb{N})$ so that $\mu = \beta f_a(\gamma)$, there exists $\delta \in \beta(\Sigma^* \otimes \mathbb{N})$ such that $\nu = \beta f_a(\delta)$ and the equation $\beta f_r(\gamma) \approx \beta f_r(\delta)$ holds in $\mathcal{B}$;
- for each $\delta \in \beta(\Sigma^* \otimes \mathbb{N})$ so that $\nu = \beta f_a(\delta)$, there exists $\gamma \in \beta(\Sigma^* \otimes \mathbb{N})$ such that $\mu = \beta f_a(\gamma)$ and the equation $\beta f_r(\gamma) \approx \beta f_r(\delta)$ holds in $\mathcal{B}$.

▶ **Theorem 21.** *The ultrafilter equations in $\mathcal{E}(\mathcal{B}\Diamond\!\!\!\!\!+2)$ characterise the Boolean algebra $\mathcal{B}\Diamond\!\!\!\!\!+2$.*

The proof of Theorem 21 relies on the following two lemmas.

▶ **Lemma 22.** *Let $\gamma \in \beta(\Sigma^* \otimes \mathbb{N})$. If $\mu = \beta f_a(\gamma)$ and $L \in \beta f_r(\gamma)$, then $La\Sigma^* \in \mu$.*

▶ **Lemma 23.** *Let $\mathscr{F} \subseteq \mathcal{P}(\Sigma^*)$ be a proper filter, $\mu \in \beta(\Sigma^*)$ and $a \in \Sigma$. If $La\Sigma^* \in \mu$ for all $L \in \mathscr{F}$, then there exists $\gamma \in \beta(\Sigma^* \otimes \mathbb{N})$ such that $\mu = \beta f_a(\gamma)$ and $\mathscr{F} \subseteq \beta f_r(\gamma)$.*

Proof Idea for Theorem 21. Soundness follows easily from the lemmas. For completeness notice that, by repeated use of compactness, $K \in \mathcal{P}(\Sigma^*)$ belongs to $\mathcal{B}\Diamond\!\!\!\!\!+2$ if and only if for each $\mu \in \widehat{K}$, the clopen $\widehat{K}$ extends the set

$$C_\mu := \bigcap\{\widehat{L} \mid L \in \mathcal{B},\ L \in \mu\} \cap \bigcap\{\widehat{La\Sigma^*} \mid a \in \Sigma, L \in \mathcal{B}, La\Sigma^* \in \mu\} \cap \bigcap\{(\widehat{La\Sigma^*})^c \mid a \in \Sigma, L \in \mathcal{B}, La\Sigma^* \notin \mu\}.$$

Finally one shows, again using the lemmas, that $\mu \approx \nu \in \mathcal{E}(\mathcal{B}\Diamond\!\!\!\!\!+2)$ for any $\nu \in C_\mu$. ◀

7 Conclusion

The concepts of recognition and of syntactic monoid, stemming from the algebraic theory of regular languages, inherently arise in the setting of Stone/Priestley duality for Boolean algebras and lattices with additional operations, see [7]. Reasoning by analogy, this led in

[8] to generalisations of recognition and syntactic objects for arbitrary languages of finite words. In *loc. cit.* this was achieved in the setting of monoids equipped with uniform space structures, the so called *semiuniform monoids*. In this paper we naturally arrive at an isomorphic notion of recogniser – Boolean spaces with internal monoids – which is however more amenable to existing tools from duality theory.

Our first contribution is setting up the right framework that allows us to extend to the non-regular setting algebraic constructions whose logical counterpart is adding a layer of quantifier depth. We should mention that both the Schützenberger and the block product are algebraic constructions that can be used for this purpose in the regular case. However, for technical reasons, extending the former to Boolean spaces with internal monoids is more natural. The unary Schützenberger product that we introduce (which actually does not appear in the (pro)finite monoid literature to the best of our knowledge) arises naturally via duality for the Boolean algebra with quotients generated by the languages $L_\exists$, for L coming from some Boolean algebra $\mathcal{B}$. Moreover, our framework can be easily extended to the case of bounded distributive lattices, one would just need to use the Vietoris functor on spectral spaces instead. A comparison between our unary Schützenberger product and the block product introduced in [12] for finitely typed monoids remains a topic for future investigation.

Furthermore, Theorem 14 of Section 4.2 and Theorem 16 of Section 5, provide characterisations of the languages accepted by our unary and binary Schützenberger products of Boolean spaces. Finally, in Section 6 we derive a preliminary result on equations. Theorem 21 on equational completeness is by no means the final word, but rather a first stepping stone in this direction. In the regular setting, as well as in the special cases treated in [9] and [4], much smaller subsets of $\mathcal{E}(\mathcal{B}\Diamond\!\!\!\!\!\!+2)$ have been shown to provide complete axiomatisations. We expect that a notion akin to the derived categories of profinite monoid theory [23] have to be developed, and we expect the remainder of the Stone-Čech compactification to play a key rôle in this.

Acknowledgements. We are grateful to Olivier Carton, Thomas Colcombet, Andreas Krebs and Jean-Éric Pin for helpful discussions on the block and Schützenberger products, and to Jean-Éric Pin for sharing an unpublished note on profinite equations and one-step products.

References

1 J. Adámek, R. Myers, H. Urbat, and S. Milius. Varieties of languages in a category. In *LICS*, pages 414–425. IEEE, 2015.

2 F. Bonchi, M. Bonsangue, H. Hansen, P. Panangaden, J. Rutten, and A. Silva. Algebra-coalgebra duality in Brzozowski's minimization algorithm. *ACM Trans. Comput. Logic*, 15(1):3:1–3:29, 2014.

3 M. Branco and J.-É. Pin. Equations defining the polynomial closure of a lattice of regular languages. In Albers et al, editor, *ICALP 2009*, volume 5556 of *Lecture Notes In Computer Science*, pages 115–126. Springer-Verlag, 2009.

4 S. Czarnetzki and A. Krebs. Using duality in circuit complexity. *arXiv*, abs/1510.04849, 2015. To appear in LATA 2016. URL: `http://arxiv.org/abs/1510.04849`.

5 S. Eilenberg. *Automata, languages, and machines. Vol. B.* Academic Press, New York-London, 1976.

6 M. Gehrke. Stone duality, topological algebra, and recognition. *J. Pure and Appl. Algebra*, 2016.

7 M. Gehrke, S. Grigorieff, and J.-É. Pin. Duality and equational theory of regular languages. In *Automata, languages and programming II*, volume 5126 of *Lecture Notes in Comput. Sci.*, pages 246–257. Springer, Berlin, 2008.

8 M. Gehrke, S. Grigorieff, and J.-É. Pin. A topological approach to recognition. In *Automata, languages and programming II*, volume 6199 of *Lecture Notes in Comput. Sci.*, pages 151–162. Springer, Berlin, 2010.

9 M. Gehrke, A. Krebs, and J.-É. Pin. Ultrafilters on words for a fragment of logic. *Theoret. Comput. Sci.*, 610(part A):37–58, 2016.

10 M. Gehrke, D. Petrişan, and L. Reggio. The Schützenberger product for syntactic spaces. *arXiv*, abs/1603.08264, 2016.

11 N. Hindman and D. Strauss. *Algebra in the Stone-Čech compactification*. de Gruyter, 2012.

12 A. Krebs, K.-J. Lange, and S. Reifferscheid. Characterizing TC^0 in terms of infinite groups. *Theory Comput. Syst.*, 40(4):303–325, 2007.

13 K. Kuratowski. *Topology. Vol. I*. Academic Press, New York-London; Państwowe Wydawnictwo Naukowe, Warsaw, 1966.

14 R. McNaughton and S. Papert. *Counter-free automata*. The M.I.T. Press, Cambridge, Mass.-London, 1971. With an appendix by William Henneman, M.I.T. Research Monograph, No. 65.

15 J.-É. Pin. Arbres et hierarchies de concatenation. In *ICALP*, volume 154 of *Lecture Notes in Computer Science*, pages 617–628. Springer, 1983.

16 J.-É. Pin. Algebraic tools for the concatenation product. *Theoretical Computer Science*, 292(1):317–342, 2003. Selected Papers in honor of Jean Berstel.

17 J.-É. Pin and P. Weil. Profinite semigroups, Malcev products, and identities. *J. of Algebra*, 182(3):604–626, 1996.

18 C. Reutenauer. *Theoretical Computer Science 4th GI Conference: Aachen*, chapter Sur les varietes de langages et de monoïdes, pages 260–265. Springer, 1979.

19 M.-P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

20 M. H. Stone. The theory of representations for Boolean algebras. *Trans. Amer. Math. Soc.*, 40(1):37–111, 1936.

21 H. Straubing. A generalization of the Schützenberger product of finite monoids. *Theoret. Comput. Sci.*, 13(2):137–150, 1981.

22 H. Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhauser, 1994.

23 B. Tilson. Categories as algebra: an essential ingredient in the theory of monoids. *J. Pure Appl. Algebra*, 48(1-2):83–198, 1987.

Logic of Local Inference for Contextuality in Quantum Physics and Beyond*

Kohei Kishida†

Department of Computer Science, University of Oxford, Oxford, United Kingdom
Kohei.Kishida@cs.ox.ac.uk

Abstract

Contextuality in quantum physics provides a key resource for quantum information and computation. The topological approach in [Abramsky and Brandenburger, New J. Phys., 2011, Abramsky et al., CSL 2015, 2015] characterizes contextuality as "global inconsistency" coupled with "local consistency", revealing it to be a phenomenon also found in many other fields. This has yielded a logical method of detecting and proving the "global inconsistency" part of contextuality. Our goal is to capture the other, "local consistency" part, which requires a novel approach to logic that is sensitive to the topology of contexts. To achieve this, we formulate a logic of local inference by using context-sensitive theories and models in regular categories. This provides a uniform framework for local consistency, and lays a foundation for high-level methods of detecting, proving, and moreover using contextuality as computational resource.

1998 ACM Subject Classification F.4.1 Mathematical Logic, I.2.3 Deduction and Theorem Proving

Keywords and phrases Contextuality, quantum mechanics, regular category, regular logic, separated presheaf

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.113

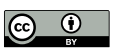
1 Introduction

Quantum physics provides quantum computing with immense advantage over classical computing. Among its non-classical properties, non-locality is known to be a basis for quantum communication (see [19]). It is in fact a special case of a property called *contextuality*, which recent studies [22, 10] suggest is an essential source of the computational power of quantum computers. This motivates the search for structural, higher-level expressions of contextuality that are independent of the formalism of quantum mechanics.

The conception of contextuality that originated in [11] exploits the structure of *presheaf*: As was shown in [11], a certain type of contextuality of a quantum system amounts to the absence of global sections from presheaves modelling the behaviors of the system. The recent, "sheaf-theoretic" approach [3] expands this insight by viewing contextuality in more general terms, as a matter of *topology* in data of measurements and outcomes: A wider range of contextuality is then characterized as the "global inconsistency" of the "locally consistent".

* Earlier versions were presented at Queen Mary University London EECS Theory Group Seminar, Birmingham Theoretical Computer Science Seminar, Amsterdam Quantum Logic Workshop 2015 (sponsored by the Netherlands Organization for Scientific Research and the University of Amsterdam), and Workshop on Information and Processes 2016 (sponsored by Université Paris Diderot, the Pendergraft Herbert Buchanan Professorship Research Fund, and the U.S. AFOSR); grateful acknowledgment goes to the audiences and sponsors. The author also thanks the anonymous reviewers for their helpful comments and suggestions.

† This work has been supported by the grant FA9550-12-1-0136 of the U.S. AFOSR.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 113; pp. 113:1–113:14

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

This has on the one hand shown that contextual phenomena can be found in various other fields such as relational database theory (see [1]), and on the other hand made it possible to apply various tools – sheaf theory, cohomology, linear algebra, for instance – to contextuality. One idea that has emerged is to formulate contextuality argument in logical terms [2]: One describes a presheaf model using logical formulas, and proves its contextuality by deriving contradiction from the formulas.

This method, however, only shows the global inconsistency of a given set of formulas; we know them to be locally consistent only because they describe a locally consistent model that is given. Nonetheless, when designing ways of exploiting contextuality, we may well first obtain a set of formulas or a specification, and then check if there is a model satisfying it. This requires a logic in which consistency means local consistency. The chief goal of this paper is to deliver such a new logic of local inference. The two logics – one for global inconsistency and the other for local consistency – together lay a foundation for high-level logical methods of not only showing but also using contextuality as resource.

Section 2 reviews the sheaf approach to contextuality, which takes presheaves valued in **Sets**. Then section 3 defines what we call "inchworm logic", a novel logic of local inference for contextual models. We formulate this on the basis of regular logic, since its vocabulary captures the essence of local inference. Semantics is provided for this logic in section 4, where we generalize **Sets**-valued presheaf models to ones valued in any regular category **S**. This encompasses cases that prove useful and powerful in applications: E.g., presheaves of abelian groups, R-modules, etc. serve the purpose of cohomology; indeed, Čech cohomology is used to detect the contextuality of **Sets**-valued presheaves [2]. This paper gives a uniform way of using **S**-valued presheaves directly as models of contextual logic.

2 Contextual Models

We first review the idea behind the formalism of [2], stressing that it applies to more settings than just quantum ones. The idea captures contextuality as a matter of topological nature, which we illustrate with a simplicial formulation equivalent to the presheaf formulation of [2]. We also present a modification of the latter that can readily be generalized in section 4.

2.1 Topological Models for Contextuality

The formalism of [3] concerns variables and values in general. Their bare-bones structure consists of a set X of variables and, for each $x \in X$, a set A_x of possible values of x. So we have an X-indexed family of sets A_x. This can model various settings in which we make queries against a system and it answers, as observed in [1, 2]; e.g.,

- We measure properties $x \in X$ of a physical system and it gives back outcomes $a \in A_x$.
- A relational database has attributes $x \in X$, and $a \in A_x$ are possible data values for x.
- $x \in X$ are sentences of propositional logic and a set of models assign to them boolean values $a \in A_x = \mathbf{2}$. Or $x \in X$ may simply be boolean variables.

We often make a query regarding several variables in combination; a set $U \subseteq X$ of variables the query concerns forms a *context* in which the system gives back a result. Contexts play essential rôles in the following two kinds of constraints, (a) on answers and (b) on queries.

(a) When we make a query in a context U, the system returns (one or a set of) tuples $s \in \prod_{x \in U} A_x$ of values. It then has the subset $A_U \subseteq \prod_{x \in U} A_x$ of "admissible" tuples that can be part of query results, and it is often the information on A_U that we want. E.g.,

- From a relational database we retrieve data with an attribute list U, and the database returns the relation A_U on sets A_x $(x \in U)$ as a table.

- Given a set of models and a set U of sentences, A_U is the set of combinations of values that U can take; e.g., a pair $\varphi, \neg\varphi \in U$ only take values $(1,0)$ or $(0,1)$.
- We may measure a physical system in various states and find that some set U of quantities always satisfy a certain equation that characterizes A_U.

(A tuple $s \in \prod_{x\in U} A_x$ is a dependent function, so it is formally a set of the form $\{\, (x, s(x)) \mid x \in U \text{ and } s(x) \in A_x \,\}$; but we may refer to it as "$(s(x), s(y), \dots)$ over $(x, y, \dots)$".)

(b) We have the family $\mathcal{C} \subseteq \mathcal{P}X$ of contexts in which queries can be made and answered. We may not be able to make a query in a context $V \subseteq X$ (i.e. $V \notin \mathcal{C}$) for reasons such as:

- V may have too many variables to deal with feasibly.
- A database schema may have no table encompassing all the attributes in V.
- Quantum mechanics may deem it impossible to measure all the properties in V at once.

In these examples, if queries can be made in a context U, they can be in any $V \subseteq U$; we also assume that queries can be made in $\{x\}$ for any $x \in X$, but only in finite U. So $\mathcal{C}$ is an (abstract) *simplicial complex* on X, i.e. a $\subseteq$-downward closed subfamily of $\mathcal{P}_{\text{fin}}X$ with $\bigcup_{U\in\mathcal{C}} U = X$. Also, if a tuple s of values is admissible, so is any $t \subseteq s$. Hence, whenever $V \subseteq U \in \mathcal{C}$, the projection of tuples $-|_V : \prod_{x\in U} A_x \to \prod_{x\in V} A_x :: s \mapsto s|_V$ restricts to $A_{V\subseteq U} : A_U \to A_V$. Thus $A : \mathcal{C}^{\text{op}} \to \mathbf{Sets}$ forms a presheaf on the poset $\mathcal{C}$.

In fact, A is a *separated* presheaf. Generally, for any subfamily $\mathcal{C}$ of $\mathcal{P}X$ closed under binary intersection, whenever $\bigcup_i U_i = U \in \mathcal{C}$ for $U_i \in \mathcal{C}$, a presheaf P on $\mathcal{C}$ has the map $\langle P_{U_i\subseteq U}\rangle_i : P_U \to \prod_i P_{U_i} :: s \mapsto (s|_{U_i})_i$ land in the set of *matching families* for $(U_i)_i$,

$$\text{Match}_{(U_i)_i,P} = \{\, (t_i)_i \in \textstyle\prod_i P_{U_i} \mid t_j|_{U_j\cap U_k} = t_k|_{U_j\cap U_k} \text{ for every pair } j,k \,\}.$$

Then P is called separated if each of these $\langle P_{U_i\subseteq U}\rangle_i$ is injective, and a *sheaf* if each of those $\langle P_{U_i\subseteq U}\rangle_i : P_U \to \text{Match}_{(U_i)_i,P}$ is bijective (see [17]). Yet, on a simplicial complex $\mathcal{C}$, separated presheaves and sheaves have simpler descriptions:

▶ **Fact 1.** A presheaf P on a simplicial complex $\mathcal{C}$ is a sheaf iff $P_U = \prod_{x\in U} P_x$ for all $U \in \mathcal{C}$. And P is separated iff it is a subpresheaf of a sheaf, i.e. iff $P_U \subseteq \prod_{x\in U} P_x$ for all $U \in \mathcal{C}$.

This shows that our A above is a separated presheaf, but not generally a sheaf. So let us write $\mathbf{sPsh}(\mathcal{C})$ for the full subcategory of $\mathbf{Sets}^{\mathcal{C}^{\text{op}}}$ of separated presheaves. Note that every sheaf F has $F_\varnothing = 1$, a singleton. A separated presheaf P has $P_\varnothing = 1$, too, unless it is the empty presheaf $U \mapsto \varnothing$, i.e. the model is inconsistent in every context (hence modelling, e.g., a physical system that never produces outcomes in any context of measurements).

2.2 Presheaves and Bundles

The constraints (a) and (b) above are, indeed, matters of topology; this idea will be useful in subsection 2.3. Given a separated presheaf A as in subsection 2.1, its underlying family of X-indexed sets $(A_x)_{x\in X}$ is equivalent to a set over X, viz. $\pi : \sum_{x\in X} A_x \to X :: (x, a) \mapsto x$, by $\mathbf{Sets}^X \simeq \mathbf{Sets}/X$. The base X comes with a simplicial complex $\mathcal{C}$, but so does $\sum_{x\in X} A_x$, taking tuples $s \in A_U$ as simplices, i.e. $\mathcal{A} = \bigcup_{U\in\mathcal{C}} A_U$. And π is a *simplicial map*, or a *bundle* of simplicial complexes, since $s \in A_U \subseteq \mathcal{A}$ implies $\pi[s] = U \in \mathcal{C}$. On the other hand, any given bundle $\pi : \mathcal{A} \to \mathcal{C}$ has a family of $A_U = \{\, s \in \mathcal{A} \mid \pi[s] = U \,\}$ and $A_{V\subseteq U} : s \mapsto s|_V$.

A simplicial map $\pi : \mathcal{A} \to \mathcal{C}$ is called *non-degenerate* if $\pi|_s$ is injective for every $s \in \mathcal{A}$. Our π above is non-degenerate, because every $s \in A_U$ is a *local section* of the bundle π, meaning $s : U \to \sum_{x\in X} A_x$ such that $\pi \circ s = 1_U$. Let us write **Simp** and **ndSimp** for the categories of simplicial maps and of non-degenerate ones, respectively. It is easy to check that for every simplicial complex $\mathcal{C}$, the slice category $\mathbf{ndSimp}/\mathcal{C}$ is a full subcategory of

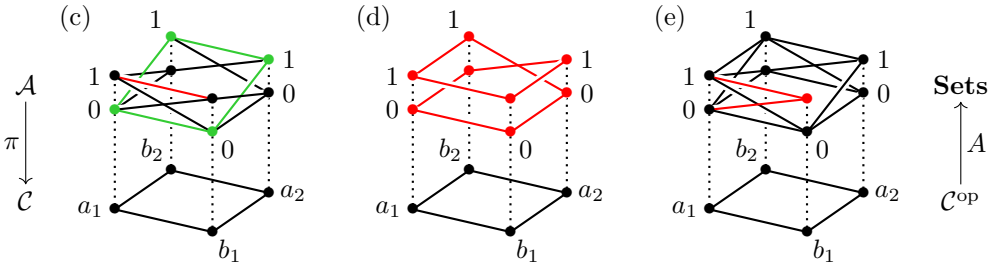


Figure 1 Bundles for (c) the Hardy model and (d) the PR-box

Simp/$\mathcal{C}$; i.e., it is the category of non-degenerate bundles and simplicial maps over $\mathcal{C}$. Then, extending $\mathbf{Sets}^X \simeq \mathbf{Sets}/X$, the correspondence described above gives

▶ **Fact 2.** $\mathbf{sPsh}(\mathcal{C}) \simeq \mathbf{ndSimp}/C$ for any simplicial complex $\mathcal{C}$.

So here is a topological reading of (a) and (b). Each $U \in \mathcal{C}$ is a local, small enough region of the space X of variables. The topology on the space $\mathcal{A}$ of values then distinguishes those tuples $s \in \prod_{x \in U} A_x$ in A_U from the others and deems the former to be continuous sections. We refer to objects of $\mathbf{sPsh}(\mathcal{C})$ and $\mathbf{ndSimp}/C$ interchangeably as *topological models*.

2.3 Contextuality in Physics, Databases, and More

Given a non-degenerate bundle $\pi : \mathcal{A} \to \mathcal{C}$ over a simplicial complex $\mathcal{C}$ on X, consider a *global section* of it, i.e. $g \in \prod_{x \in X} A_x$ such that $g|_U \in A_U$ for all $U \in \mathcal{C}$. It is an assignment of values to all the variables that satisfies every constraint on combinations of values. E.g., in classical logic, the models are exactly the global sections; so the consistency of a sentence x means that $(x, 1)$ is part of a global section. Then, in the physical setting, it may seem natural to similarly think of global sections g as states of the system, assigning values to all the quantities – so, although we can only make a query locally in a context $U \in \mathcal{C}$, the system in a state g actually has a value $g(x)$ assigned to every quantity x, and the answer we receive in the context U is simply $g|_U$. This assumption, that any section we observe is part of a context-independent global section, holds not just in classical logic but also in classical physics – but breaks down in quantum physics, precisely when contextuality arises.

Figure 1 shows "Bell-type" scenarios in which Alice and Bob measure properties of a system, perhaps a quantum one. The base $\mathcal{C}$ expresses constraints of type (b) above: Alice can make at most one of two measurements a_1 and a_2 at a time, so she chooses one; similarly Bob chooses from b_1 and b_2 – so there are four possible combinations of measurements, indicated by the four edges of $\mathcal{C}$. Alice and Bob repeat measurements in different contexts, and learn that each $x \in X = \{a_1, a_2, b_1, b_2\}$ has two possible outcomes 0 and 1, but that some combinations of outcomes are never obtained. $\mathcal{A}$ expresses these constraints, of type (a), with edges indicating possible combinations. E.g., $\mathcal{A}$ of (c) deems every joint outcome of (a_1, b_1) possible, with $A_{\{a_1,b_1\}} = \mathbf{2} \times \mathbf{2}$; but $(0, 0)$ is not a possible joint outcome of (a_2, b_2).

The models in Figure 1 all violate the classical assumption above, and are examples of

▶ **Definition 3** ([2])**.** A topological model is said to be *logically contextual* if not all of its local sections extend to global ones, and *strongly contextual* if it has no global section at all.

(c) of Figure 1 represents an example of logical contextuality due to [9] that is realizable in quantum physics. It has several global sections, e.g. the one marked in green; call it g. So, when Alice and Bob measure (a_1, b_1) and observe $(0, 0)$, the classical explanation is possible

that the system was in the state g and had outcomes $g(x)$ assigned to all the measurements $x \in X$, and that Alice and Bob have simply retrieved that information on U. On the other hand, the local section in red, $(1,1)$ over (a_1, b_1), does not extend to any global section. This means that the classical explanation is simply impossible for this joint outcome. Furthermore, the classical explanation is never possible in the strongly contextual (d). This model, called the *PR box* [21], is not quantum-realizable (though it plays an important rôle in the quantum information literature), but quantum physics exhibits many instances of strong contextuality.

The upshot is that contextuality consists in *global inconsistency coupled with local consistency*: A section $s \in A_U$ is consistent locally, in the sense of satisfying the constraint on query results in the context U, but it may be inconsistent globally, in the sense of contradicting all the other constraints and thereby failing to extend to a global section.

The general definition of contextuality in terms of global sections can also be applied to relational databases: Contextuality then corresponds exactly to the absence of a universal relation [1]. In fact, the natural join $\bowtie_{U \in \mathcal{C}} A_U = \{\, g \in \prod_{x \in X} A_x \mid g|_U \in A_U \text{ for all } U \in \mathcal{C} \,\}$ of relations A_U (which is the largest of universal relations if there are any) is, simply by definition, the set of global sections.

2.4 No-Signalling Principle

Not just there being local sections, local consistency involves more – viz. a constraint that is called the *no-signalling* principle in the physical setting [7]. For a topological model A, it amounts to the condition that every $A_{U \subseteq V} : A_V \to A_U :: s \mapsto s|_U$ is a surjection.

An example violating no-signalling is (e) of Figure 1: $A_{\{b_1\} \subseteq \{a_2, b_1\}} : A_{\{a_2, b_1\}} \to A_{\{b_1\}}$ is not surjective. Suppose Alice and Bob make measurements, Bob chooses b_1, and he observes 1, which is not in the image of $A_{\{b_1\} \subseteq \{a_2, b_1\}}$. This means that Bob has received the signal from Alice (no matter how far away she may be!) that she has chosen a_1 and not a_2.

To see why no-signalling should be part of local consistency, regard $\mathcal{A}$ in (e) as representing a relational database. It has tables $A_{\{a_1, b_1\}}$ and $A_{\{a_2, b_1\}}$; but, when queried about the attribute b_1, they yield different results of projection, differing in whether 1 is in or not. Thus, no-signalling means the consistency of projections (see [1]). Indeed, as we will see in subsection 4.3, no-signalling means a sort of coherence of A as a semantic model of logic.

▶ **Definition 4.** We say that a separated presheaf $A : \mathcal{C}^{\mathrm{op}} \to \mathbf{Sets}$ is no-signalling if it satisfies (1), and that a non-degenerate bundle $\pi : \mathcal{A} \to \mathcal{C}$ is no-signalling if it satisfies (2):

(1) Every $A_{U \subseteq V} : A_V \to A_U$ is a surjection.

(2) If $\pi[s] \subseteq U$ for $s \in \mathcal{A}$ and $U \in \mathcal{C}$, then there is some $t \in \mathcal{A}$ such that $s \subseteq t$ and $\pi[t] = U$.

Clearly, (1) and (2) coincide via $\mathbf{sPsh}(\mathcal{C}) \simeq \mathbf{ndSimp}/\mathcal{C}$. Hence their full subcategories of no-signalling models are equivalent. Note that (1) or (2) implies $A_U \neq \varnothing$ for all $U \in \mathcal{C}$, if $A_\varnothing \neq \varnothing$. So, while the empty model is no-signalling, all the other, nonempty no-signalling models (which are, essentially, the "empirical models" of [2]) are locally consistent.

3 Contextual Logics

3.1 Contextuality Argument: Logic of Global Inconsistency

Viewing A_U as representing a constraint on assignments of values to variables $x \in U$, we can describe a topological model A using formulas in contexts $U \in \mathcal{C}$ of variables. E.g., the assignments of $(0,0)$ and $(1,1)$ to (x, y) satisfy the equation $x \oplus y = 0$, where $\oplus$ is for XOR,

i.e. addition modulo 2; the assignments $(0,1)$ and $(1,0)$ satisfy $x \oplus y = 1$. Therefore the PR box, (d) of Figure 1, satisfies the following set of equations:

$$a_1 \oplus b_1 = 0, \quad a_1 \oplus b_2 = 0, \quad a_2 \oplus b_1 = 0, \quad a_2 \oplus b_2 = 1 \tag{3}$$

These are in fact inconsistent: Their right-hand sides sum to 1, but the left to 0 regardless of the values of variables (since each variable occurs twice). This is to say that no global assignment of values satisfies all the constraints of A_U, i.e., that A is strongly contextual.

A family of arguments of this sort, using XOR (or parity) equations, has been given to show the strong contextuality of a range of quantum examples; the first instance in literature was in [18] for the GHZ state [8]. This sort of so-called "all-vs-nothing argument" was formalized and generalized in [2]. On the other hand, one may also adopt more expressive languages, such as Boolean formulas, to express a wider range of constraints.

Formulas can also be used to show logical (and not strong) contextuality. E.g., the Hardy model, (c) of Figure 1, satisfies the antecedents of

$$\neg a_1 \vee \neg b_2, \quad \neg a_2 \vee \neg b_1, \quad a_2 \vee b_2 \quad \vdash \quad \neg a_1 \vee \neg b_1 \tag{4}$$

but not the consequent, due to the contextual section $(1,1)$ over (a_1, b_1). This shows that this local section can be part of no global assignment satisfying all the constraints.

Yet this kind of contextuality argument needs some reflection. The inconsistency of a set Γ of formulas, $\Gamma \vdash \bot$, does not mean that Γ has no model; in fact, the PR box, (d) of Figure 1, satisfies all the equations in (3). In the same vein, the derivability $\Gamma \vdash \varphi$ does not mean that every model of Γ satisfies φ; the Hardy model (c) satisfies Γ but not φ of (4). So the logic of $\vdash$ here is *not* sound with respect to contextual models – indeed, that is the whole point of the argument. Invalidating $\vdash$ precisely means contextuality: $\Gamma \vdash \bot$ really means that no global section satisfies Γ; it is why any model of Γ has no global section. $\Gamma \vdash \varphi$ means that every global section satisfying Γ satisfies φ; it is why any model satisfying Γ but not φ must have local sections (viz. ones not satisfying φ) that fail to extend to global sections.

In this sense, the logic of $\vdash$ here is a "global logic" of global sections. We should then note that this logic, by itself, says very little about local consistency. To see this, consider:

$$a_1 \oplus b_1 = 0, \quad a_1 \oplus b_1 = 1 \tag{5}$$

This set of equations is, like (3), inconsistent. It is, however, inconsistent not just globally but also locally: Not only does no global section satisfy both equations, no local section over the context $\{a_1, b_1\}$ does; a model A satisfies (5) only if $A_{\{a_1,b_1\}} = \varnothing$ (the physical system can give no outcomes to the measurements a_1, b_1). Yet the global logic does not tell us why (3) is locally consistent whereas (5) is not. Thus the kind of argument above is really a "global-inconsistency argument": It shows contextuality only because we already know the formulas to be locally consistent, having obtained them as descriptions of some model.

3.2 "Inchworm Logic" of Local Inference

$\Gamma \vdash \bot$ of (5) means local inconsistency over $\{a_1, b_1\}$ since both equations in Γ are in the context $\{a_1, b_1\}$. Turning $\Gamma, \varphi \vdash \bot$ into the form of inference, if Γ, φ are in the context U, $\Gamma \vdash \neg\varphi$ gives local entailment over U. E.g., the antecedents of $a_1 = 0$, $b_1 = 0 \vdash a_1 \oplus b_1 = 0$ rule out all the sections over (a_1, b_1) except $(0,0)$, which satisfies the consequent.

Indeed, local inference can be carried out across different contexts, validly in no-signalling models, subject to one constraint. To see this, expand the base $\mathcal{C}$ in Figure 1 from (f) of

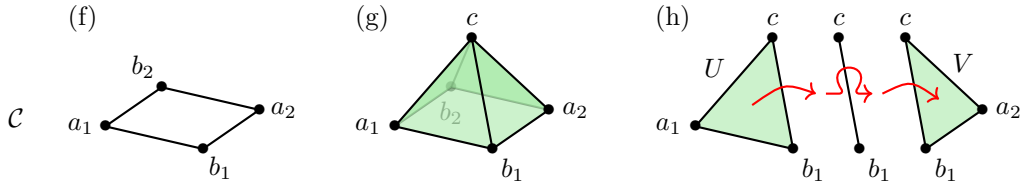


Figure 2 Charlie and an inchworm

Figure 2 to (g), where the four triangles are in $\mathcal{C}$ – so a new experimenter, Charlie, can make his measurement c along with Alice and Bob.

Now rewrite the locally consistent (3) in the inference form (6) (we replace $x \oplus y = 0$ with simpler $x = y$) and compare it to (7):

$$a_1 = b_1, \quad a_1 = b_2, \quad a_2 = b_1 \quad \vdash \quad a_2 = b_2 \tag{6}$$

$$a_1 = b_1, \quad a_1 = c, \quad a_2 = b_1 \quad \vdash \quad a_2 = c \tag{7}$$

(7) is valid in no-signalling models, whereas (6) is not (the PR box is a countermodel, as it is a model for (3)). The only difference is c replacing b_2 – this tiny difference, however, enables us to obtain (7) in the following two steps:

$$\dfrac{\dfrac{a_1 = b_1 \qquad a_1 = c}{b_1 = c}\,U \qquad a_2 = b_1}{a_2 = c}\,V \tag{8}$$

The first step is within the context $U = \{a_1, b_1, c\}$, hence valid locally: Every section over U satisfying the antecedents satisfies the consequent. Similarly, the second step is valid within $V = \{a_2, b_1, c\}$. The key aspect is that the formula in the middle, $b_1 = c$, can be in the context $U \cap V$ and so in U or in V. The upshot is that information gets passed on from a larger context U to a smaller $U \cap V$ and then to another larger V – just like the locomotion of an inchworm, if (h) of Figure 2 helps to visualize it. Crucially, the no-signalling property is essential when the inchworm moves from a larger context to a smaller: E.g., the first step of (8) concludes that every section over $\{a_1, b_1, c\}$ satisfies $b_1 = c$; but then, in the absence of no-signalling, there may be a section over $\{b_1, c\}$ violating $b_1 = c$ without extending to $\{a_1, b_1, c\}$. We will discuss the semantic rôle of no-signalling further in subsection 4.3.

3.3 Formalizing the Inchworm

We formalize and generalize the idea of "inchworm inference". As in the example in subsection 3.2, an inchworm logic is obtained by constraining a global logic. We assume this logic to be (at least) regular, i.e. to have $\top$, $\wedge$, and $\exists$, for the reasons explained shortly.

▶ **Definition 5.** Let $\mathcal{L}$ be a language of regular logic (or richer) whose variables include X. For each $x \in X$, write T_x for the type of x, and then, for each $\bar{x} \subseteq X$, write $\Phi_{\bar{x}}$ for the set of formulas in the context $\bar{x} : T_{\bar{x}}$. Given a simplicial complex $\mathcal{C}$ on X, the *$\mathcal{C}$-contextual fragment* of $\mathcal{L}$ is $\Phi_{\mathcal{C}} = \bigcup_{U \in \mathcal{C}} \Phi_U$. By a *$\mathcal{C}$-contextual language* $\mathcal{L}_{\mathcal{C}}$, we simply mean a pair of such $\mathcal{L}$ and $\Phi_{\mathcal{C}}$. Now let $\mathbb{T}$ be a regular theory in $\mathcal{L}$ given by an entailment relation $\vdash$ (which is *not* required to be binary). Then the inchworm fragment of $\vdash$ in $\mathcal{L}_{\mathcal{C}}$ is the entailment relation $\vdash_{\mathcal{C}}$ on $\Phi_{\mathcal{C}}$ defined inductively by the following. We write $\Gamma_U = \Gamma \cap \Phi_U$.

(9) $\Gamma \vdash_{\mathcal{C}} \varphi$ if there is $U \in \mathcal{C}$ such that $\varphi \in \Phi_U$ and $\Gamma_U \vdash \varphi$.

(10) If $\Gamma \vdash_{\mathcal{C}} \varphi$ and $\Delta, \varphi \vdash_{\mathcal{C}} \psi$ then $\Gamma, \Delta \vdash_{\mathcal{C}} \psi$.

(9) expresses the idea that $\vdash$ within a single context is valid locally as well. In (10), note that the first two instances of $\vdash_{\mathcal{C}}$ may be witnessed by different contexts. Observe also that $\Gamma \vdash_{\mathcal{C}} \varphi$ entails $\Gamma \vdash \varphi$; thus $\vdash_{\mathcal{C}}$ is a fragment of $\vdash$.

▶ **Example 6.** Let $\mathcal{L}$ have T as a basic type; 0, 1 be constants of type T; and $\oplus$ be a function symbol of type $\oplus : T \times T \to T$. Let $T_x = T$ for all $x \in X$. So, e.g., $x : T, y : T \mid x \oplus y = 1$ makes sense, and is in $\Phi_{\{x,y\}}$. This gives equations of the kind relevant to the examples in subsection 3.1. Note that $\Phi_{\mathcal{C}}$ is a union. E.g., for (f) of Figure 2, $a_i = 0$ is in $\Phi_{\{a_i\}} \subseteq \Phi_{\mathcal{C}}$ for both $i = 1, 2$, but $a_1 = 0 \wedge a_2 = 0$ is *not* in $\Phi_{\mathcal{C}}$ since $\{a_1, a_2\} \notin \mathcal{C}$.

We assume $\mathcal{L}$ to have $\top$ and $\wedge$, so that pieces of information can be combined within the same context. The inchworm moves from a smaller context U to a larger V via the order embedding $i : (\Phi_U, \vdash_U) \hookrightarrow (\Phi_V, \vdash_V)$, and from V to U via the left adjoint $\exists_{V \setminus U}$ of i. Then $\exists_{V \setminus U} \dashv i$ means that, for any $\varphi \in \Phi_V$, $\exists_{V \setminus U}.\varphi \in \Phi_U$ encapsulates all and only the information that φ entails on U. We also have $\exists_{V \setminus U} \circ i \cong 1$, so a piece of information that can be both about U and about V undergoes no change when carried across U and V.

4 Contextual Semantics in Regular Categories

Our definition of model using a sheaf generalizes by replacing **Sets** with any category $\mathbf{S}$ with finite limits, since the base $\mathcal{C}$ is a simplicial complex. Yet, for the sake of no-signalling, we moreover need $\mathbf{S}$ to be regular. References on regular categories and their categorical logic include [20, 5, 12]. We then lay out how to model the inchworm logic in $\mathbf{S}$.

4.1 Topological Models in Regular Categories

Let $\mathcal{C}$ be a simplicial complex on a set X, and $\mathbf{S}$ be a category with finite limits. By a presheaf on $\mathcal{C}$ valued in $\mathbf{S}$, we mean any contravariant functor $P : \mathcal{C}^{\mathrm{op}} \to \mathbf{S}$. Then the definitions of separated presheaf and sheaf generalize straightforwardly to

▶ **Definition 7.** We say that an $\mathbf{S}$-valued presheaf P on $\mathcal{C}$ is separated if the arrow $\langle P_{U_i \subseteq U} \rangle_i : P_U \to \prod_i P_{U_i}$ is monic whenever $\bigcup_i U_i = U$, and a sheaf if, whenever $\bigcup_i U_i = U$, $\langle P_{U_i \subseteq U} \rangle_i$ is an equalizer as follows, where $p_j : \prod_i P_{U_i} \to P_{U_j}$ and $p_k : \prod_i P_{U_i} \to P_{U_k}$ are the projections.

$$P_U \rightarrowtail \prod_i P_{U_i} \rightrightarrows \prod_{j,k} P_{U_j \cap U_k}$$

with the parallel pair $\langle P_{U_j \cap U_k \subseteq U_j} \circ p_j \rangle_{j,k}$ and $\langle P_{U_j \cap U_k \subseteq U_k} \circ p_k \rangle_{j,k}$.

Again, every sheaf F has $F_\varnothing = 1$, the terminal object of $\mathbf{S}$, and every separated presheaf P has $P_\varnothing \rightarrowtail 1$. Also, for simpler descriptions, Fact 1 generalizes to

▶ **Fact 8.** An $\mathbf{S}$-valued presheaf P on a simplicial complex $\mathcal{C}$ is a sheaf iff $P_U = \prod_{x \in U} P_x$ for all $U \in \mathcal{C}$. And P is separated iff it is a subpresheaf of a sheaf, i.e. iff each $\langle P_{\{x\} \subseteq U} \rangle_{x \in U} : P_U \to \prod_{x \in U} P_x$ is monic, i.e. iff each P_U is a relation in $\mathbf{S}$ on $(P_x)_{x \in U}$.

Next we define the no-signalling property for $\mathbf{S}$-valued separated presheaves. In doing so, we need to choose from several generalizations of the notion of surjection in (1) of Definition 4; the one that serves our purpose is the one that provides semantics for $\exists : \Phi_V \rightleftarrows \Phi_U : i$ in the inchworm logic. This is the principal reason we need $\mathbf{S}$ to be regular; then, in $\mathbf{S}$, every arrow $f : C \to D$ gives rise to the adjoint pair $\exists_f \dashv f^{-1}$, $\exists_f : \mathrm{Sub}_{\mathbf{S}}(C) \rightleftarrows \mathrm{Sub}_{\mathbf{S}}(D) : f^{-1}$ (e.g. [5, Lemma 2.5]), and moreover $\exists_f \circ f^{-1} = 1_{\mathrm{Sub}(D)}$ (and so f^{-1} is an order embedding) if f is a regular epi (essentially, [12, Corollary D1.2.8]). Therefore the right generalization of Definition 4 is the following Definition 9, with an alternative description in Fact 10.

▶ **Definition 9.** A separated presheaf A on a simplicial complex $\mathcal{C}$ valued in a regular category $\mathbf{S}$ is said to be no-signalling if every $A_{U\subseteq V} : A_V \to A_U$ is a regular epi.

▶ **Fact 10.** Let F be a sheaf on a simplicial complex $\mathcal{C}$. Then a family $(i_U : A_U \rightarrowtail F_U)_{U\in\mathcal{C}}$ of subobjects forms a subpresheaf of F, and hence a separated presheaf, iff $A_V \leqslant {F_{U\subseteq V}}^{-1}(A_U)$, or equivalently $\exists_{F_{U\subseteq V}}(A_V) \leqslant A_U$, whenever $U \subseteq V \in \mathcal{C}$. Moreover, a separated presheaf $i : A \rightarrowtail F$ is no-signalling iff $\exists_{F_{U\subseteq V}}(A_V) = A_U$ whenever $U \subseteq V \in \mathcal{C}$.

4.2 Global Inconsistency in Regular Categories

Definition 3 of contextuality for **Sets**-valued presheaves can now extend to ones valued in any regular category $\mathbf{S}$. Let A be an $\mathbf{S}$-valued separated presheaf on a simplicial complex $\mathcal{C}$ on X. It is a subpresheaf of a sheaf F on $\mathcal{C}$. In fact, let us assume, just in this subsection, that X is finite (or that $\mathbf{S}$ is complete); then, by Fact 8 (or a straightforward generalization), F extends uniquely to a sheaf on $\mathcal{P}X$, viz. $F : U \mapsto \prod_{x\in U} F_x$. Then the set of global sections of A – i.e. the natural join $\bowtie A$ of the relations $A_U \subseteq F_U$ – generalizes to the $\mathbf{S}$-valued case:

▶ **Fact 11.** Given any $\mathbf{S}$-valued separated presheaf A, let F be a sheaf such that $i : A \rightarrowtail F$ and, using A_U as predicates in the internal language of $\mathbf{S}$, define

$$\bowtie A = [\![\, \bar{x} : F_X \mid \textstyle\bigwedge_{U\in\mathcal{C}} A_U(F_{U\subseteq X}\bar{x}) \,]\!] = \textstyle\bigwedge_{U\in\mathcal{C}} {F_{U\subseteq X}}^{-1}(A_U) \rightarrowtail F_X.$$

Then $\bowtie A$ is the limit of A as a diagram in $\mathbf{S}$.

For each $U \in \mathcal{C}$, write $\rho_U : \bowtie A \to A_U$ for the restriction of $F_{U\subseteq X}$ to $\bowtie_A$; it generalizes the restriction of global sections to local sections over U. Definition 3 then extends to

▶ **Definition 12.** An $\mathbf{S}$-valued separated presheaf A is said to be logically contextual if not every $\rho_U : \bowtie A \to A_U$ is a regular epi. A is moreover said to be strongly contextual if $\bowtie A$ is not well-supported, i.e. if the unique arrow $!_{\bowtie A} : \bowtie A \to 1$ is not a regular epi.

Rewriting this in the internal language of $\mathbf{S}$, the strong contextuality of A means that $\mathbf{S}$ fails $\exists\bar{x} : F_X.\bowtie A(\bar{x})$, i.e., that no global section $\bar{x}$ satisfies all the constraints A_U. The logical contextuality means that $\bar{x} : F_V \mid A_V(\bar{x}) \vdash \exists\bar{y} : F_{X\setminus V}.\bowtie A(\langle\bar{x},\bar{y}\rangle)$ fails in $\mathbf{S}$ for some $V \in \mathcal{C}$, i.e., that not every local section $\bar{x}$ over V satisfying A_V extends to a global section $\langle\bar{x},\bar{y}\rangle$ satisfying all A_U.

4.3 Contextual Interpretation

In Definition 5 we defined a contextual language $\mathcal{L}_\mathcal{C}$ and logic $\vdash_\mathcal{C}$ simply as a global language $\mathcal{L}$ and logic $\vdash$ paired with their contextual fragments. Our definition of an interpretation of them in regular categories goes in parallel.

▶ **Definition 13.** Given a contextual language $\mathcal{L}_\mathcal{C} = (\mathcal{L}, \Phi_\mathcal{C})$, an interpretation of it in a regular category $\mathbf{S}$ is simply an interpretation $[\![-]\!]$ of $\mathcal{L}$ in $\mathbf{S}$. The images of T_x and Φ_U then play special rôles: For each $\bar{x} \in \mathcal{C}$, we have

- $[\![T_{\bar{x}}]\!] = \prod_{x\in\bar{x}} [\![T_x]\!]$; therefore $[\![T_-]\!] : \mathcal{C}^{\mathrm{op}} \to \mathbf{S}$ forms a sheaf by Fact 8.
- Moreover, $[\![\, \bar{x} : T_{\bar{x}} \mid \varphi \,]\!] \rightarrowtail [\![T_{\bar{x}}]\!]$ for each $\varphi \in \Phi_{\bar{x}}$.

So we may write $([\![-]\!], F)$ for the interpretation $[\![-]\!]$, where F is the sheaf $F : \bar{x} \mapsto [\![T_{\bar{x}}]\!]$. We may also write $[\![\varphi]\!]_{\bar{x}} \rightarrowtail F_{\bar{x}}$ for $[\![\, \bar{x} : T_{\bar{x}} \mid \varphi \,]\!]$.

▶ **Example 14.** Expanding Example 6, take $[\![-]\!]$ in **Sets** with $[\![T]\!] = \mathbf{2}$ and the obvious $[\![0]\!]$, $[\![1]\!]$, and $[\![\oplus]\!]$. Then we have a sheaf $[\![T_-]\!] : U \mapsto \mathbf{2}^U$ and, e.g., $[\![\, x : T, y : T \mid x \oplus y = 0 \,]\!] \rightarrowtail [\![T_{\{x,y\}}]\!] = \mathbf{2} \times \mathbf{2}$ is an equalizer of $[\![\oplus]\!], [\![0]\!] \circ ! : \mathbf{2} \times \mathbf{2} \to \mathbf{2}$.

An interpretation $[\![-]\!]$ of $\mathcal{L}$ is said to model a sequent $\Gamma \vdash \varphi$ if some finite $\Delta \subseteq \Gamma$ has $\bigwedge_{\psi\in\Delta}[\![\psi]\!]_U \leqslant [\![\varphi]\!]_U$. This makes sense whether $U \in \mathcal{C}$ or not. Nevertheless, if $U \notin \mathcal{C}$, then $\bigwedge_{\psi\in\Delta}[\![\psi]\!]_U \leqslant [\![\varphi]\!]_U$ only means the global entailment and not the local one. Take, e.g.,

▶ **Example 15.** Expanding Example 14, the model A of the PR box, (d) of Figure 1, is a subpresheaf of $[\![T\_]\!]$ described by (3): E.g. $A_{\{a_i,b_1\}} = [\![\, a_1 \oplus b_1 = 0 \,]\!]_{\{a_1,b_1\}} \rightarrowtail \mathbf{2}\times\mathbf{2}$. Then the global inconsistency, and strong contextuality in particular, of the equations Γ in (3) means $\bowtie A = \bigcap_{\varphi\in\Gamma}[\![\varphi]\!]_X \subseteq [\![\bot]\!]_X = \varnothing$. Yet Γ is locally consistent, modelled by the PR box.

This is why, to model the inchworm logic of local inference, we need a presheaf on different contexts $U \in \mathcal{C}$, as opposed to an intersection in a single context $V \notin \mathcal{C}$, to the left of $\leqslant$.

▶ **Definition 16.** Suppose $([\![-]\!], F)$ is an interpretation of a contextual language $\mathcal{L}_\mathcal{C}$. Then let us say that a subpresheaf $A \rightarrowtail F$ is a *pre-model* in $([\![-]\!], F)$ of a formula $\varphi \in \Phi_U$ in a context $U \in \mathcal{C}$, and write $A \vDash_U \varphi$, to mean that $A_U \leqslant [\![\varphi]\!]_U$.

▶ **Fact 17.** If $A \leqslant B$ for subpresheaves A and B of F, then $B \vDash_U \varphi$ implies $A \vDash_U \varphi$.

Note, however, that this notion of pre-model is context-dependent and concerns formulas in contexts as opposed to formulas *per se*. When $U \subseteq V \in \mathcal{C}$ and $\varphi \in \Phi_U$, Fact 10 yields

(11) $A \vDash_U \varphi$ entails $A \vDash_V \varphi$ (because $\exists_{F_{U\subseteq V}}(A_V) \leqslant A_U \leqslant [\![\varphi]\!]_U$ entails $A_V \leqslant {F_{U\subseteq V}}^{-1}[\![\varphi]\!]_U = [\![\varphi]\!]_V$).

(12) Suppose A is no-signalling. Then $A \vDash_V \varphi$ entails $A \vDash_U \varphi$. (This is because $A_V \leqslant [\![\varphi]\!]_V = {F_{U\subseteq V}}^{-1}[\![\varphi]\!]_U$ implies $A_U = \exists_{F_{U\subseteq V}}(A_V) \leqslant [\![\varphi]\!]_U$.)

If A is not no-signalling, (12) may fail, and then inchworm inference fails. E.g., in (8), the first step purports to show that, if $A \vDash_U a_1 = b_1$ and $A \vDash_U a_1 = c$, then $A \vDash_U b_1 = c$ and so $A \vDash_{U\cap V} b_1 = c$; but the "and so" step here requires (12). In this sense, no-signalling means the context-independent coherence of a presheaf as a model of formulas. Therefore

▶ **Definition 18.** A pre-model A is called a (no-signalling) *model* if it is no-signalling. Then we say that A is a model of a formula $\varphi \in \Phi_\mathcal{C}$, and write $A \vDash \varphi$, to mean that A is a pre-model of φ in any suitable context, i.e., that $A_U \leqslant [\![\varphi]\!]_U$ for every $U \in \mathcal{C}$ such that $\varphi \in \Phi_U$.

▶ **Theorem 19.** *Let $[\![-]\!]$ be an interpretation of $\mathcal{L}_\mathcal{C}$ that models a theory $\vdash$ in $\mathcal{L}$. Then the inchworm logic $\vdash_\mathcal{C}$ of $\vdash$ is sound with respect to the no-signalling models in $[\![-]\!]$: If $\Gamma \vdash_\mathcal{C} \varphi$, then $A \vDash \varphi$ for every no-signalling model A of Γ in $[\![-]\!]$.*

4.4 The Inchworm and No-Signalling

Subsection 4.3 primarily concerned how given presheaves modelled formulas. We showed in particular that no-signalling validated inchworm inference. Let us discuss, on the other hand, how the description by given formulas yields a model. This shows the other direction of the connection between no-signalling and the inchworm, from the latter to the former.

We say a set $\Gamma \subseteq \Phi_\mathcal{C}$ of formulas of $\mathcal{L}_\mathcal{C}$ is $\mathcal{C}$-*finite* if Γ_U is finite for each $U \in \mathcal{C}$.

▶ **Definition 20.** Let $([\![-]\!], F)$ be an interpretation of $\mathcal{L}_\mathcal{C}$. Given any $\mathcal{C}$-finite $\Gamma \subseteq \Phi_\mathcal{C}$, define $\mathbb{M}_F(\Gamma)$ as a family $(\mathbb{M}_F(\Gamma)_U = \bigwedge_{\varphi\in\Gamma_U}[\![\varphi]\!]_U \rightarrowtail F_U)_{U\in\mathcal{C}}$ of subobjects of F_U.

▶ **Fact 21.** $\mathbb{M}_F(\Gamma)$ is the largest subpresheaf A of F such that $A \vDash_U \Gamma_U$ for each $U \in \mathcal{C}$.

$\mathbb{M}_F(\Gamma)$ generally fails to be no-signalling (take the first step of (8) as an example again). Yet the description by Γ sometimes manages to give a no-signalling $\mathbb{M}_F(\Gamma)$.

▶ **Fact 22.** Let $(\llbracket - \rrbracket, F)$ be an interpretation of $\mathcal{L}_\mathcal{C}$ that models a theory $\vdash$ in $\mathcal{L}$. We say $\Gamma \subseteq \Phi_\mathcal{C}$ is *inchworm-saturated* if $\Gamma_V \vdash \varphi$ implies $\Gamma_U \vdash \exists_{V \setminus U}. \varphi$ whenever $U \subseteq V \in \mathcal{C}$ and $\varphi \in \Phi_V$. Now, if a $\mathcal{C}$-finite Γ is inchworm-saturated, then $\mathbb{M}_F(\Gamma)$ is no-signalling.

When Γ is inchworm-saturated, it may not be deductively closed, but the inchworm cannot bring a new piece of information ψ to a context U from another V, since ψ follows from the information Γ_U that U already has. Fact 22 means that, if Γ is inchworm-saturated and if each Γ_U finite and consistent (and has $\mathbb{M}_F(\Gamma)_U$ nonempty or well-supported), then Γ is locally consistent, modelled by a no-signalling model $\mathbb{M}_F(\Gamma)$. E.g., (3) is inchworm-saturated, with each context consistent, so it gives the PR box, (d) of Figure 1, as $\mathbb{M}_F(\Gamma)$.

On the other hand, even when a description Γ is not inchworm-saturated and $\mathbb{M}_F(\Gamma)$ fails to be no-signalling, the inchworm can carve out the "no-signalling interior" of $\mathbb{M}_F(\Gamma)$, if Γ can be saturated in finite (or $\mathcal{C}$-finite) steps.

▶ **Theorem 23.** *Let $(\llbracket - \rrbracket, F)$ be an interpretation of $\mathcal{L}_\mathcal{C}$ that models a theory $\vdash$ in $\mathcal{L}$. Given $\Gamma \subseteq \Phi_\mathcal{C}$, suppose there is a $\mathcal{C}$-finite and inchworm-saturated $\Delta \subseteq \Phi_\mathcal{C}$ such that $\Gamma \subseteq \Delta$ and $\Gamma \vdash_\mathcal{C} \varphi$ for all $\varphi \in \Delta$. Then $\mathbb{M}_F(\Delta)$ is the largest no-signalling subpresheaf of $\mathbb{M}_F(\Gamma)$.*

E.g., A in (e) of Figure 1 is $\mathbb{M}_F(\Gamma)$ given by $\Gamma = \{\varphi\}$ for $\varphi = (a_2 \wedge \neg b_1) \vee (\neg a_2 \wedge \neg b_1)$; since φ cannot be in the context $\{b_1\}$, $\Gamma_{\{b_1\}} = \varnothing$ and $A_{\{b_1\}} = \mathbf{2}$. Yet $\varphi \vdash \neg b_1$, so $\Gamma \vdash_\mathcal{C} \neg b_1$, and $\Delta = \Gamma \cup \{\neg b_1\}$ is inchworm-saturated, with $\neg b_1 \in \Delta_{\{b_1\}}$. Hence, by Theorem 23, the inchworm carves out a no-signalling $\mathbb{M}_F(\Delta)$ by removing the red sections from (e). Indeed, in many applications (e.g. all the examples in sections 2 and 3), the theory $\vdash$ satisfies

(13) Given any $\Gamma \subseteq \Phi_\mathcal{C}$ (that may not be $\mathcal{C}$-finite), for each $U \in \mathcal{C}$ there is a finite $\Delta_U \subseteq \Gamma_U$ such that $\Delta_U \vdash \varphi$ for all $\varphi \in \Gamma_U$.

This guarantees the supposition of Theorem 23: Given any $\mathcal{C}$-finite $\Gamma \subseteq \Phi_\mathcal{C}$, take its $\vdash_\mathcal{C}$-deductive closure $\Gamma^* = \{\, \varphi \mid \Gamma \vdash_\mathcal{C} \varphi \,\}$ as Γ in (13) and obtain Δ_U; then $\Delta = \Gamma \cup \bigcup_{U \in \mathcal{C}} \Delta_U$ is such as in Theorem 23. Therefore Theorem 23 applies and leads to a family of completeness results organized by Lemma 24, which transfers a completeness theorem of a global theory to its inchworm fragment. It yields, e.g., Theorem 25, since any (global) regular theory has a "conservative model" in a "classifying category" (e.g. [5, Proposition 6.4]).

▶ **Lemma 24.** *Suppose that a theory $\vdash$ in $\mathcal{L}$ satisfies* (13)*, and that $\llbracket - \rrbracket$ is a conservative model of $\vdash$, meaning that, for any $\Gamma \subseteq \Phi_\mathcal{C}$, $\bigwedge_{\psi \in \Delta} \llbracket \psi \rrbracket_U \leqslant \llbracket \varphi \rrbracket_U$ for some $\Delta \subseteq \Gamma$ if but also only if $\Gamma \vdash \varphi$. Then $\Gamma \vdash_\mathcal{C} \varphi$ iff $A \vDash \varphi$ for every no-signalling model A of Γ in $\llbracket - \rrbracket$.*

▶ **Theorem 25.** *Let $\vdash$ be a regular theory satisfying* (13)*. Then, for any $\Gamma \subseteq \Phi_\mathcal{C}$, $\Gamma \vdash_\mathcal{C} \varphi$ iff $A \vDash \varphi$ for every no-signalling model A of Γ in every model $\llbracket - \rrbracket$ of $\vdash$ in any regular category.*

4.5 Completion for Completeness

Generally, (13) may fail and the inchworm saturation may not be attained in finite steps.

▶ **Example 26.** In Figure 1, replace each $A_x = \mathbf{2}$ with $\mathbb{Z}$, and let $\Gamma = \{\, a_1 = b_2, b_1 = a_1, a_2 = b_1, b_2 = a_2 + 1, b_2 > 0 \,\}$ in the obvious $\mathcal{L}$ and $\vdash$. Then $\Gamma \vdash_\mathcal{C} a_1 > 0, b_1 > 0, a_2 > 0, b_2 > 1, \ldots, x > n$ for every $x \in X$ and $n \in \mathbb{N}$, whereas $\Gamma \nvdash_\mathcal{C} \bot$ (although the empty presheaf is the only no-signalling model of Γ). So there cannot be any such Δ as in Theorem 23. (Note that the topology of $\mathcal{C}$ is essential: E.g., if we take $\mathcal{C} = \mathcal{P}X$ instead, then $\Gamma \vdash_\mathcal{C} \bot$ by $\Gamma \vdash \bot$.)

Thus, even if Γ is finite, the set $\{\, \llbracket \varphi \rrbracket_U \mid \varphi \in \Gamma^*{}_U \,\}$ may have no minimum (though it is lowerbounded by $\exists_{F_U \subseteq X} (\bigwedge_{\psi \in \Gamma} \llbracket \psi \rrbracket_X)$ if X is finite); then, in a regular category in general, $\bigwedge_{\varphi \in \Gamma^*{}_U} \llbracket \varphi \rrbracket_U$ may not exist. So, instead of the semilattice $\mathrm{Sub}_\mathbf{S}(F_U)$ of subobjects, let us

use a completion of it, viz. the semilattice $\mathrm{Filt}(\mathrm{Sub}_{\mathbf{S}}(F_U))$ of filters in $\mathrm{Sub}_{\mathbf{S}}(F_U)$, and assign a filter of subobjects, instead of a subobject, to each $U \in \mathcal{C}$.

▶ **Definition 27.** Suppose $([\![-]\!], F)$ is an interpretation of a contextual language $\mathcal{L}_{\mathcal{C}}$. Then, by a *filter model* in $([\![-]\!], F)$, we mean a presheaf $G : \mathcal{C}^{\mathrm{op}} \to \mathbf{Sets}$ such that

- $G_U \in \mathrm{Filt}(\mathrm{Sub}_{\mathbf{S}}(F_U))$ for every $U \in \mathcal{C}$.
- For $U \subseteq V \in \mathcal{C}$, $G_U = \{\, S \rightarrowtail F_U \mid {F_{U\subseteq V}}^{-1}(S) \in G_V \,\} = \{\, \exists_{F_{U\subseteq V}}(S) \rightarrowtail F_U \mid S \in G_V \,\}$, so $G_{U\subseteq V} : G_V \to G_U :: S \mapsto \exists_{F_{U\subseteq V}}(S)$ is a surjection.

We say G models φ, and write $G \vDash \varphi$, to mean that $[\![\varphi]\!]_U \in G_U$ whenever $\varphi \in \Phi_U$.

Then we have the filter versions of Theorem 19, Fact 22, and completeness results organized by Lemma 24. Observe that every (no-signalling) model A is a "principal" filter model, $U \mapsto \uparrow A_U = \{\, S \rightarrowtail F \mid A_U \leqslant S \,\}$; so Theorem 28 is stronger than Theorem 19.

▶ **Theorem 28.** *Let $[\![-]\!]$ be a model of a theory $\vdash$ in $\mathcal{L}$. Then the inchworm logic $\vdash_{\mathcal{C}}$ of $\vdash$ is sound with respect to the filter models in $[\![-]\!]$: If $\Gamma \vdash_{\mathcal{C}} \varphi$, then $G \vDash \varphi$ for every filter model G of Γ in $[\![-]\!]$.*

▶ **Fact 29.** Let $([\![-]\!], F)$ be a model of a theory $\vdash$ in $\mathcal{L}$. Given any $\Gamma \subseteq \Phi_{\mathcal{C}}$, the family $\mathbb{FM}_F(\Gamma) = (\{\, S \rightarrowtail F_U \mid [\![\varphi]\!]_U \leqslant S \text{ for some } \varphi \in {\Gamma^*}_U \,\})_{U\in\mathcal{C}}$ is a filter model of Γ in $([\![-]\!], F)$. Moreover, for any filter model G of Γ in $([\![-]\!], F)$, $\mathbb{FM}_F(\Gamma)_U \subseteq G_U$ for each $U \in \mathcal{C}$.

▶ **Lemma 30.** *Suppose $[\![-]\!]$ is a conservative model of a theory $\vdash$ in $\mathcal{L}$. Then $\Gamma \vdash_{\mathcal{C}} \varphi$ iff $G \vDash \varphi$ for every filter model G of Γ in $[\![-]\!]$.*

5 Conclusion

Let us conclude the paper by discussing connections and applications between the framework of this paper and other approaches or other fields as future work. First of all, categorical logic has a long tradition (since [16]) of viewing local truth as a modal operator. Indeed, the logic of local information in this paper is closely related to the dynamic-logical characterization of contextuality in [13]. There is also a connection to model theory. For instance, the similarity between inchworm inference and Craig interpolation should be obvious; indeed, by defining Φ_U more generally as a "language in the vocabulary U", we can prove a stronger version of Robinson's joint consistency theorem (see [6, subsection 4.1.1]) that is sensitive to the topology of $\mathcal{C}$.

As explained in section 2, presheaf models can model Boolean valuations. This enables us to transfer and apply techniques from satisfiability problems to quantum contextuality as computational resource. Another connection is to the structure of valuation algebra, which is used for local computation [14]. In fact, our presheaf models can also be formulated in terms of valuation algebras, as a $\mathcal{C}$-indexed family of valuations satisfying certain conditions. We can expect these connections to help extend local computation to situations in classical computing where contextual phenomena arise.

The generality of taking presheaves in regular categories is also expected to facilitate applications. In cohomology, it is typical to use presheaves valued in regular categories, such as presheaves of abelian groups, R-modules, etc. Therefore the framework of this paper applies to the logic of local inference within such presheaves. One can also take regular categories of structures that are used for other purposes such as modelling processes in quantum physics. In addition, the connection to logical paradoxes [2] is also relevant. As shown in [15, 4], regular categories provide background for self-referential and other fixpoint paradoxes; so our formalism will unify the two perspectives on paradoxes.

References

1 Samson Abramsky. Relational databases and Bell's theorem. In Val Tannen, Limsoon Wong, Leonid Libkin, Wenfei Fan, Wang-Chiew Tan, and Michael P. Fourman, editors, *In Search of Elegance in the Theory and Practice of Computation: Essays Dedicated to Peter Buneman*, volume 8000 of *Lecture Notes in Computer Science*, pages 13–35. Springer, 2013. doi:10.1007/978-3-642-41660-6_2.

2 Samson Abramsky, Rui Soares Barbosa, Kohei Kishida, Raymond Lal, and Shane Mansfield. Contextuality, cohomology and paradox. In Stephan Kreutzer, editor, *24th EACSL Annual Conference on Computer Science Logic, CSL 2015, September 7–10, 2015, Berlin, Germany*, volume 41 of *LIPIcs*, pages 211–228. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2015. URL: http://www.dagstuhl.de/dagpub/978-3-939897-90-3, doi:10.4230/LIPIcs.CSL.2015.211.

3 Samson Abramsky and Adam Brandenburger. The sheaf-theoretic structure of non-locality and contextuality. *New J. Phys.*, 13:113036–113075, 2011. doi:10.1088/1367-2630/13/11/113036.

4 Samson Abramsky and Jonathan A. Zvesper. From Lawvere to Brandenburger-Keisler: Interactive forms of diagonalization and self-reference. *J. Comput. Syst. Sci.*, 81(5):799–812, 2015. doi:10.1016/j.jcss.2014.12.001.

5 Carsten Butz. Regular categories and regular logic. BRICS Lecture Series LS-98-2, BRICS, 1998.

6 Dov M. Gabbay and Larisa Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logics*. Oxford Logic Guides. Clarendon Press, Oxford, 2005.

7 Giancarlo Ghirardi, Alberto Rimini, and Tullio Weber. A general argument against superluminal transmission through the quantum mechanical measurement process. *Lettere al Nuovo Cimento*, 27:293–298, Mar 1980. doi:10.1007/BF02817189.

8 Daniel M. Greenberger, Michael A. Horne, and Anton Zeilinger. Going beyond Bell's theorem. In M. Kafatos, editor, *Bell's Theorem, Quantum Theory, and Conceptions of the Universe*, pages 69–72. Kluwer, 1989.

9 Lucien Hardy. Nonlocality for two particles without inequalities for almost all entangled states. *Phys. Rev. Lett.*, 71:1665–1668, Sep 1993. doi:10.1103/PhysRevLett.71.1665.

10 Mark Howard, Joel Wallman, Victor Veitch, and Joseph Emerson. Contextuality supplies the 'magic' for quantum computation. *Nature*, 510:351–355, June 2014. doi:10.1038/nature13460.

11 Chris J. Isham and Jeremy Butterfield. Topos perspective on the Kochen–Specker theorem: I. Quantum states as generalized valuations. *Int. J. Theor. Phys.*, 37:2669–2733, November 1998. doi:10.1023/A:1026680806775.

12 Peter T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium*, volume 2 of *Oxford Logic Guides*. Clarendon Press, Oxford, 2002.

13 Kohei Kishida. Stochastic relational presheaves and dynamic logic for contextuality. In Bob Coecke, Ichiro Hasuo, and Prakash Panangaden, editors, *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL 2014), Kyoto, Japan, 4–6th June 2014*, volume 172 of *EPTCS*, pages 115–132, 2014. URL: http://eptcs.web.cse.unsw.edu.au/content.cgi?QPL2014, doi:10.4204/EPTCS.172.9.

14 Jürg Kohlas, Marc Pouly, and Cesar Schneuwly. Generic local computation. *J. Comput. Syst. Sci.*, 78(1):348–369, 2012. doi:10.1016/j.jcss.2011.05.012.

15 F. William Lawvere. Diagonal arguments and cartesian closed categories. *Lecture Notes in Mathematics*, 92:134–145, 1969.

16 F. William Lawvere. Quantifiers and sheaves. *Actes, Congrès intern. math.*, 1:329–334, 1970.

17 Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Springer, 1992.

18 N. David Mermin. Extreme quantum entanglement in a superposition of macroscopically distinct states. *Phys. Rev. Lett.*, 65:1838–1840, Oct 1990. doi:10.1103/PhysRevLett.65.1838.

19 Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.

20 Jaap van Oosten. Basic category theory. BRICS Lecture Series LS-95-1, BRICS, 1995.

21 Sandu Popescu and Daniel Rohrlich. Quantum nonlocality as an axiom. *Found. Phys.*, 24:379–385, Mar 1994. doi:10.1007/BF02058098.

22 Robert Raussendorf. Contextuality in measurement-based quantum computation. *Phys. Rev. A*, 88:022322, Aug 2013. doi:10.1103/PhysRevA.88.022322.

Minimizing Resources of Sweeping and Streaming String Transducers*

Félix Baschenis[1], Olivier Gauwin[2], Anca Muscholl[3], and Gabriele Puppis[4]

1 **University of Bordeaux, LaBRI, CNRS, Bordeaux, France**
2 **University of Bordeaux, LaBRI, CNRS, Bordeaux, France**
3 **University of Bordeaux, LaBRI, CNRS, Bordeaux, France; and Institute for Advanced Study of the Technical University of Munich, Munich, Germany**
4 **University of Bordeaux, LaBRI, CNRS, Bordeaux, France**

Abstract

We consider minimization problems for natural parameters of word transducers: the number of passes performed by two-way transducers and the number of registers used by streaming transducers. We show how to compute in ExpSpace the minimum number of passes needed to implement a transduction given as sweeping transducer, and we provide effective constructions of transducers of (worst-case optimal) doubly exponential size. We then consider streaming transducers where concatenations of registers are forbidden in the register updates. Based on a correspondence between the number of passes of sweeping transducers and the number of registers of equivalent concatenation-free streaming transducers, we derive a minimization procedure for the number of registers of concatenation-free streaming transducers.

1998 ACM Subject Classification F.4.3 Formal Languages, F.1.1 Models of Computation, F.2.0 Analysis of Algorithms and Problem Complexity – General

Keywords and phrases word transducers, streaming, 2-way, sweeping transducers, minimization

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.114

1 Introduction

Regular word functions extend the robust family of regular languages, preserving many of its characterizations and algorithmic properties. A word function maps words over a finite input alphabet to words over a finite output alphabet. Regular word functions have been studied in the early seventies, in the form of (deterministic) two-way finite state automata with output [1]. Engelfriet and Hoogeboom [8] later showed that monadic second-order definable graph transductions, restricted to words, are an equivalent model – this justifies the notation "regular" word functions, in the spirit of classical results in automata theory and logic by Büchi, Elgot, Rabin and others. Recently, Alur and Cerný [2] proposed an enhanced version of one-way transducers called streaming transducers, and showed that they equivalent to the two previous models. A streaming transducer processes the input word from left to right, and stores (partial) output words in finitely many, write-only registers. A variant of streaming transducers extended by stacks has been introduced in [3] and shown to capture precisely the monadic-second order definable tree transductions.

* This work was partially supported by the ExStream project (ANR-13-JS02-0010) and the TUM-IAS, funded by the German Excellence Initiative and the EU 7th Framework Programme (grant 291763).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 114; pp. 114:1–114:14

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

Two-way and streaming transducers raise new and challenging questions about storage requirements. The classical storage measure for automata is state complexity. State space minimization of two-way transducers is however poorly understood, even in the simpler setting of automata (cf. related work). But there are more meaningful parameters for transducer minimization. One such parameter for streaming transducers is the number of registers, and for two-way transducers it is the number of times the transducer needs to re-process the input word. These parameters measure the required storage capacity in a more realistic way than the number of control states. For example, a two-way transducer that needs to process a very large input with several passes has much larger memory requirements in practice than the memory needed for storing the states. Ideally, the input is processed one-way, hence in one pass only, as in the streaming setting. But not every transduction can be implemented by a one-way, finite state transducer without additional memory.

The register minimization problem has been considered by Alur and Raghothaman in [4], for a special family of deterministic streaming transducers: the output alphabet is unary, and the updates are additions/subtractions of registers by constants. For two-way transducers, Filiot et al. showed how to decide whether a transducer is equivalent to some one-way transducer [10]. The decision procedure of [10] is non-elementary, and we provided in [5] an elementary decision procedure and construction of equivalent one-way transducers in the special case of *sweeping* transducers: head reversals are only allowed at the extremities of the input. Sweeping transducers are strictly less expressive than two-way transducers, as shown e.g. by the transduction mapping inputs of the form $u_1\#u_2\#\cdots\#u_n$, where the words u_i contain no occurrence of $\#$, to $u_n\cdots u_2u_1$.

In this paper we extend our results from [5] by showing how to compute in ExpSpace the *minimal number of passes* needed by a non-deterministic, functional sweeping transducer. It turns out that sweeping transducers have the same expressive power as bounded-reversal two-way transducers, and as *concatenation-free* streaming transducers – transducers where concatenation of registers is not allowed in the updates. Since the transformations between the sweeping and streaming models preserve the relationship between the number of passes and the number of registers, we reduce the *minimization problem for registers* of concatenation-free streaming transducers to the minimization of the number of passes of sweeping transducers, and thus solve the former problem.

Related work. As already mentioned, succinctness questions about two-way automata are still challenging. A longstanding open problem is whether non-deterministic two-way automata are exponentially more succinct than deterministic two-way automata. It is only known that this is the case for deterministic sweeping automata [13].

Regular transductions behave also nicely in terms of expressiveness: first-order definable transductions are known to be equivalent to transductions defined by aperiodic streaming transducers [11] and by aperiodic two-way transducers [6].

Besides [4], the closest work to ours is [7], that shows how to compute the minimal number of registers of deterministic streaming transducers with register updates of the form $x := y \cdot v$, where v is a word and x, y are registers. These transducers are as expressive as one-way transducers. However, the focus of [7] is different from ours, since the outputs can be formed over any infinitary group. Moreover, the works [4, 7] consider deterministic transducers, which require in general more registers than non-deterministic functional ones. The proof techniques are based on variants of a property that has been studied for one-way transducers (the twinning property), and are quite different from ours.

Overview. After introducing two-way and streaming transducers in Section 2, and showing some basic properties, we recall in Section 3 the key characterization of one-way definability from [5]. Section 4 presents the main result on minimization of sweeping transducers. Finally, Section 5 concludes with a logical characterization for sweeping transducers. A longer version of the paper is available at `https://hal.archives-ouvertes.fr/hal-01274992`.

2 Preliminaries

Here we introduce the transducers we are interested in: two-way and streaming transducers.

Two-way transducers. A *two-way transducer* is a tuple $T = (Q, \Sigma, \Delta, I, E, F)$, where Q is a finite set of states, Σ (resp. Δ) is a finite input (resp. output) alphabet, I (resp. F) is a subset of Q representing the initial (resp. final) states, and $E \subseteq Q \times \Sigma \times \Delta^\star \times Q \times \{\mathsf{left}, \mathsf{right}\}$ is a finite set of transition rules describing, for each state and input symbol, the possible output string, target state, and direction of movement. To enable distinguished transitions at the extremities of the input word, we use two special symbols $\rhd$ and $\lhd$ and assume that the input of a two-way transducer is of the form $u = a_1 \dots a_n$, with $n \geq 2$, $a_1 = \rhd$, $a_n = \lhd$, and $a_i \neq \rhd, \lhd$ for all $i = 2, \dots, n-1$.

Given an input word u, we call *positions* the places between the symbols of u, where the head of a transducer can lie. We can identify the positions of $u = a_1 \dots a_n$ with the numbers $1, \dots, n-1$, where each number x is seen as the position between a_x and a_{x+1}. Since here we deal with *two-way* devices, a position can be visited several times along a run. Formally, we associate the states of the transducer with *locations*, namely, with pairs (x, y), where x is a position and y is a non-negative integer, called *level*. For convenience, we assume that, from a location at even level, the transducer can either move to the next position to the right, without changing the level, or perform a reversal, that is, increment the level by 1 and keep the same position; symmetrically, from a location at odd level, the transducer can either move leftward, without changing the level, or perform a reversal. Locations are ordered according to the following order: $\ell \leq \ell'$ if $\ell = (x, y), \ell' = (x', y')$ and one of the following holds: (1) $y < y'$, or (2) $y = y'$ even and $x \leq x'$, or (3) $y = y'$ odd and $x \geq x'$.

Formally, we define a run on $u = a_1 \dots a_n$ as a sequence of locations, labeled by states and connected by edges, hereafter called *transitions*. The state at a location $\ell = (x, y)$ of a run ρ is denoted $\rho(\ell)$. The transitions must connect pairs of locations $\ell \leq \ell'$ that are either at adjacent positions and on the same level, or at the same position and on adjacent levels. Each transition is labeled with a pair a/v consisting of an input symbol a and a word v produced as output. There are four types of transitions:

$$(x, 2y+1) \xleftarrow{a_{x+1}/v} (x+1, 2y+1) \qquad (x, 2y) \xrightarrow{a_{x+1}/v} (x+1, 2y)$$

$$a_{x+1}/v \circlearrowright \begin{matrix}(x+1, 2y+2)\\(x+1, 2y+1)\end{matrix} \qquad \begin{matrix}(x, 2y+1)\\(x, 2y)\end{matrix} \circlearrowleft a_{x+1}/v$$

The upper left (resp. upper right) transition can occur in a run ρ of T on u provided that $\big(\rho(x+1, 2y+1), a_{x+1}, v, \rho(x, 2y+1), \mathsf{left}\big)$ (resp. $\big(\rho(x, 2y), a_{x+1}, v, \rho(x+1, 2y), \mathsf{right}\big)$) is a valid transition rule of T and a_{x+1} is the $(x+1)$-th symbol of u (assuming that first symbol is $\rhd$). Similarly, the lower left (resp. lower right) transition are called *reversals*, and can occur in a run ρ if $\big(\rho(x+1, 2y+1), a_{x+1}, v, \rho(x+1, 2y+2), \mathsf{right}\big)$ (resp. $\big(\rho(x, 2y), a_{x+1}, v, \rho(x, 2y+1), \mathsf{left}\big)$) is a valid transition rule of T and a_{x+1} is the $(x+1)$-th symbol of u.

We say that a run on $u = a_1 \dots a_n$ is *successful* if it starts with an initial state, either at location $(1, 0)$ or at location $(n-1, 1)$, and ends in a final state, at some location of the form $(1, y_{\mathsf{max}})$ or $(n-1, y_{\mathsf{max}})$. The output produced by a run ρ is the concatenation of the words produced by its transitions, and it is denoted by $\mathsf{out}(\rho)$.

Crossing sequences. An important notion associated with runs of two-way automata is that of crossing sequence. Intuitively, this is a tuple of states that label those locations of a run that visit the same position. Formally, given a successful run ρ of a two-way transducer on input $u = a_1 \dots a_n$, the *crossing sequence of ρ at a position $x \in \{1, \dots, n-1\}$* is the tuple $\rho|x = \big(\rho(x, y_0), \dots, \rho(x, y_h)\big)$, where $y_0 < \dots < y_h$ are all and only the levels of the locations of ρ at position x. The classical transformation of two-way finite state automata into equivalent one-way automata [12] uses crossing sequences.

Properties of two-way transducers. We say that a two-way transducer is

- *sweeping* if every run performs the reversals only at the extremities of the input word, i.e. when reading the symbols $\triangleright$ or $\triangleleft$;
- *L-sweeping* if it is sweeping and all successful runs start at the leftmost location $(1, 0)$;
- *R-sweeping* if it is sweeping and all successful runs start at the rightmost location $(n–1, 1)$;
- *k-pass* if every successful run visits every position of the input at most k times;
- *k-reversal* if every successful run performs at most k reversals;
- *one-way* if it is 1-pass, L-sweeping.

A transducer is *functional* if it produces at most one output on each input. It is called *unambiguous* if it admits at most one successful run on each input. These notions will have the same meaning for streaming transducers, defined later. Clearly, every unambiguous transducer is functional. The converse is not true in general, but we will see later that we can transform the functional transducers considered in this paper so as to enforce unambiguity.

It is easy to see that every unambiguous transducer with n states is $2n$-pass. For functional transducers we can restrict ourselves to considering only *normalized* runs, namely, runs that never visit the same position twice with the same state and the same direction. The reason is that functionality guarantees that every factor of a successful run that starts and ends at the same position and with the same state produces the empty output.

Hereafter, we silently assume that all transducers are *functional* and all successful runs are *normalized*. As a consequence the length of the crossing sequences of the successful runs of a transducer can be bounded by $2n$, where n is the number of states of the transducer.

For streaming transducers, we can observe the following. Every k-pass R-sweeping transducer can be transformed into an equivalent $(k+1)$-pass L-sweeping transducer. It is also easy to disambiguate functional sweeping transducers, that is, transform them into equivalent unambiguous sweeping transducers, without increasing the number of passes. For this it suffices to fix a total order on the successful runs, e.g. the lexicographic order, and restrict to runs that are minimal among those over the same input.

The following proposition shows an interesting correspondence between the number of passes of sweeping transducers and the number of reversals of two-way transducers.

▶ **Proposition 1.** *Every k-pass sweeping transducer is also $(k-1)$-reversal. Conversely, every $(k-1)$-reversal two-way transducer can be transformed in* 2ExpTime *into an equivalent unambiguous k-pass sweeping transducer. The transformation can be performed in* ExpTime *if the $(k-1)$-reversal transducer is unambiguous.*

Streaming transducers. Streaming transducers can implement the same transductions as two-way transducers [2, 8], but they do so using a single left-to-right pass and a fixed set of registers that can store words over the output alphabet.

Formally, a *streaming transducer* is a tuple $T = (Q, \Sigma, \Delta, R, U, I, E, F)$, where Q is a finite set of states, Σ (resp. Δ) is a finite input (resp. output) alphabet, R is a finite set of registers disjoint from Δ, U is a finite set of *updates* for the registers, namely, functions from R to $(R \uplus \Delta)^\star$, I is a subset of Q representing the initial states, $E \subseteq Q \times \Sigma \times U \times Q$ is a finite set of transition rules, describing, for each state and input symbol, the possible updates and target states, and $F : Q \rightharpoonup (R \uplus \Delta)^\star$ is a partial output function.

A well-behaved class of streaming transducers [2] is obtained by restricting the allowed types of updates and partial output functions to be copyless. A streaming transducer $T = (Q, \Sigma, \Delta, R, U, I, E, F)$ is *copyless* if (1) for every update $f \in U$, every register $z \in R$ appears at most once in $f(z_1) \cdot \ldots \cdot f(z_k)$, where $R = \{z_1, \ldots, z_k\}$, and (2) for every state $q \in Q$, every register $z \in R$ appears at most once in $F(q)$. Hereafter we assume that all streaming transducers are copyless.

To define the semantics of a streaming transducer $T = (Q, \Sigma, \Delta, R, U, I, E, F)$, we introduce *valuations* of registers in R. These are functions of the form $g : R \to \Delta^\star$. Valuations can be homomorphically extended to words over $R \cup \Delta$ and to updates, as follows. For every valuation $g : R \to \Delta^\star$ and every word $w \in (R \cup \Delta)^\star$, we let $g(w)$ be the word over Δ obtained from w by replacing every occurrence of a register z with its valuation $g(z)$. Similarly, for every valuation $g : R \to \Delta^\star$ and every update $f : R \to (R \cup \Delta)^\star$, we denote by $g \circ f$ the valuation that maps each register z to the word $g(f(z))$.

A *configuration* of T is a pair state-valuation (q, g). This configuration is said to be initial if $q \in I$ and $g(z) = \varepsilon$ for all registers $z \in R$. When reading a symbol a, the transducer can move from a configuration (q, g) to a configuration (q', g') if there exists a transition rule $(q, a, f, q') \in E$ such that $g' = g \circ f$. We denote this by $(q, g) \xrightarrow[T]{a} (q', g')$.

A *run* of T on $u = a_1 \ldots a_n$ is a sequence of configurations and transitions of the form $\sigma = (q_0, g_0) \xrightarrow[T]{a_1} (q_1, g_1) \xrightarrow[T]{a_2} \ldots \xrightarrow[T]{a_n} (q_n, g_n)$. The run ρ is *successful* if the partial output function F is defined on the last state q_n. In this case, the *output* of T on u is $g_n(F(q_n))$.

Properties and relationships with sweeping transducers. Functional and unambiguous streaming transducers are defined as in the two-way case. A streaming transducer is *k-register* if it uses at most k registers. As we did for two-way transducers, we assume that all streaming transducers are functional.

It is known that (functional) streaming transducers capture precisely the transductions definable by deterministic two-way transducers or, equally, by monadic second-order logic (so-called MSO transductions) [2, 8]. Moreover, differently from two-way transducers, non-deterministic streaming transducers can be determinized. This happens at the cost of increasing the number of registers.

▶ **Definition 2.** A streaming transducer $T = (Q, \Sigma, \Delta, R, U, I, E, F)$ is *concatenation-free* if $f(z) \in \Delta^\star \cdot (R \cup \{\varepsilon\}) \cdot \Delta^\star$, for all registers $z \in R$ and all updates $f \in U$.

Intuitively, a concatenation-free streaming transducer forbids register updates with two or more registers inside a right-hand side. We note that concatenation-free streaming transducers can also be determinized effectively. Moreover, it is easy to see that allowing boundedly many updates with concatenations does not change the expressiveness of the model, as one can remove any occurrence of an update with concatenations by introducing new registers.

The following proposition shows a tight correspondence between the number of registers of the concatenation-free streaming transducers and the number of passes of the sweeping transducers. Note that the proposition considers sweeping transducers that start from the rightmost position. A slightly weaker correspondence holds for L-sweeping transducers, since any sweeping transducer can be made L-sweeping (resp. R-sweeping) by increasing the number of passes by 1.

▶ **Proposition 3.** *Every concatenation-free streaming transducer with k registers can be transformed in* ExpTime *into an equivalent unambiguous $2k$-pass R-sweeping transducer. The transformation is in* Ptime *if the streaming transducer is unambiguous.*

Conversely, every k-pass R-sweeping transducer can be transformed in 2ExpTime *into an equivalent unambiguous concatenation-free streaming transducer with $\lceil \frac{k}{2} \rceil$ registers. The transformation is in* ExpTime *if the sweeping transducer is unambiguous.*

Based on the above proposition, the problem of minimizing the number of registers in a concatenation-free streaming transducer reduces to the problem of minimizing the number of passes performed by a sweeping transducer. We will thus focus on the latter problem: in Section 4, we consider the decidability and complexity of the following problem, called *k-pass sweeping definability problem*: given a functional sweeping transducer S and a number $k \in \mathbb{N}$, decide whether S has an equivalent k-pass sweeping transducer.

3 One-way definability

In [5] we gave an effective characterization of sweeping transducers that are *one-way definable*, i.e., equivalent to some one-way transducer. This can be seen as a special case of the problem that we are considering here, and some of the technical tools developed in [5] will be used later. We briefly recall some definitions and results related to this characterization. Hereafter we assume that S is an L-sweeping transducer and ρ a successful run of S.

Intercepted factors. An *interval* of positions of the run ρ has the form $I = [x_1, x_2]$, with $x_1 < x_2$. We say that an interval $I = [x_1, x_2]$ *contains* (resp., *strongly contains*) another interval $I' = [x'_1, x'_2]$ if $x_1 \leq x'_1 \leq x'_2 \leq x_2$ (resp., $x_1 < x'_1 \leq x'_2 < x_2$). We say that a factor of ρ is *intercepted* by an interval $I = [x_1, x_2]$ if it is maximal among the factors of ρ that visit only positions in I and that never make a reversal (recall that reversals in sweeping transducers can only occur at the extremities of the input word).

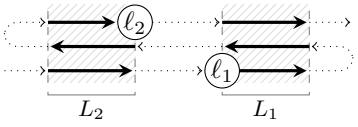
Pumping loops. A *loop* of a run ρ is an interval $L = [x_1, x_2]$ of positions such that the crossing sequences $\rho|x_1$ and $\rho|x_2$ are equal. If L is a loop of ρ, we can obtain new runs by replicating any number of times the factors of ρ intercepted by L and, simultaneously, the factor of the input word u between positions x_1 and x_2. This operation is called *pumping* and is formally defined as follows. Let $L = [x_1, x_2]$ be a loop of a run ρ on u. The run obtained by pumping n times the loop L is the sequence $\mathsf{pump}^n_L(\rho) = \underbrace{\alpha_1 \, \boldsymbol{\beta_1^n} \, \gamma_1}_{\text{1st pass}} \; \underbrace{\alpha_2 \, \boldsymbol{\beta_2^n} \, \gamma_2}_{\text{2nd pass}} \; \cdots \; \underbrace{\alpha_k \, \boldsymbol{\beta_k^n} \, \gamma_h}_{k\text{-th pass}}$ where k is the number of passes performed by ρ, β_i is the factor intercepted by L at the i-th level, α_i is the factor intercepted either by $[1, x_1]$ or by $[x_2+1, |u|]$ at the i-th level, depending on whether this level is even or odd, and, symmetrically, γ_i is the factor intercepted either by $[x_2+1, |u|]$ or by $[1, x_1]$ at the i-th level, depending on whether this level is even or odd. We also define $\mathsf{pump}^n_L(u) = u[1, x_1] \cdot \boldsymbol{\big(u[x_1+1, x_2]\big)^n} \cdot u[x_2+1, |u|]$ and we observe that $\mathsf{pump}^n_L(\rho)$ is a valid run on $\mathsf{pump}^n_L(u)$.

It is convenient to introduce some notation for pumping runs on multiple loops. If the loops are pairwise non-overlapping this can be done by simply pumping each loop separately, since the order in which we pump the loops does not really matter. The situation is a bit more complicated when some loops overlap. In particular, when pumping a loop L of ρ, several copies of the original locations of ρ are introduced, and with this several copies of other loops may appear (think, for example, of a loop L' that is contained in L). We say that a location $\tilde{\ell}$ in $\mathsf{pump}^n_L(\rho)$ *corresponds* to ℓ in ρ if $\tilde{\ell}$ is one of the copies of ℓ that is introduced when pumping ρ on L. We extend this correspondence to sets of locations and loops. With a slight abuse of notation, we denote by $\mathsf{pump}^{n_2}_{L_2}(\mathsf{pump}^{n_1}_{L_1}(\rho))$ the run obtained by first pumping n_1 times the loop L_1 in ρ, and then pumping n_2 times *every loop* that corresponds to L_2 in $\mathsf{pump}^{n_1}_{L_1}(\rho)$ (note that the copies of L_2 in $\mathsf{pump}^{n_1}_{L_1}(\rho)$ are pairwise non-overlapping). It is routine to check that the two runs $\mathsf{pump}^{n_2}_{L_2}(\mathsf{pump}^{n_1}_{L_1}(\rho))$ and $\mathsf{pump}^{n_1}_{L_1}(\mathsf{pump}^{n_2}_{L_2}(\rho))$ are isomorphic. This allows us to use the shorthand $\mathsf{pump}^{\overline{n}}_{\overline{L}}(\rho)$ to denote runs obtained from ρ by pumping the loops $\overline{L} = L_1, \ldots, L_m$ with the numbers $\overline{n} = n_1, \ldots, n_m$, respectively.

Inversions. The notion of inversion is crucial for characterizing one-way definability [5]. Let L be a loop of ρ. A location ℓ_1 is called an *entry point of* L if it is the first location of a factor intercepted by L. Similarly, a location ℓ_2 is called an *exit point of* L if it is the last location of a factor intercepted by L. Note that every entry/exit point of $L = [x_1, x_2]$ occurs either at position x_1 or at position x_2.

▶ **Definition 4.** An *inversion* of a run ρ is a pair of locations ℓ_1 and ℓ_2 for which there exist two loops $L_1 = [x_1, x'_1]$ and $L_2 = [x_2, x'_2]$ such that (also refer to the figure on the right):

- ℓ_1 is an entry point of L_1 and ℓ_2 is an exit point of L_2,
- $\ell_1 < \ell_2$ and $x_2 \leq x'_1$,
- for both $i = 1$ and $i = 2$, the factor intercepted by L_i and visiting ℓ_i has non-empty output, and no other loop strongly contained in L_i has the same property as L_i w.r.t. this factor.



We say that the loops L_1 and L_2 are the *witnessing loops* of the inversion (ℓ_1, ℓ_2).

Periodic words. A word w is said to have *period* p if $w \in u^* v$ for some word u of length p and some prefix v of u. For example, $w = abc\,abc\,ab$ has period $p = 3$.

We are interested into factors of the outputs of S that are periodic, with uniformly bounded periods. To do this, we fix the constant $e_S = c_S \cdot |Q|^{2|Q|}$, where c_S is the maximum number of symbols output by a single transition of S and Q is the state space of S. The crux in [5] is the following property:

▶ **Proposition 5** (Prop. 7 in [5]). *If S is a one-way definable L-sweeping transducer and (ℓ_1, ℓ_2) is an inversion of a successful run ρ of S, then $\mathsf{out}(\rho[\ell_1, \ell_2])$ has period at most e_S.*

The above result justifies the following definition: let $L_S \subseteq \mathrm{dom}(S)$ be the language of those words u that induce a successful run ρ of S such that, for all inversions (ℓ_1, ℓ_2) of ρ, $\mathsf{out}(\rho[\ell_1, \ell_2])$ is periodic with period at most e_S. We denote by $S|_{L_S}$ the transducer S restricted to inputs from L_S. One-way definability is characterized as follows:

▶ **Theorem 6** (Th. 1 in [5]). *An L-sweeping transducer S is one-way definable if and only if $L_S = dom(S)$. Moreover, given an L-sweeping transducer S, one can construct in doubly exponential time a one-way transducer T that is equivalent to $S|_{L_S}$.*

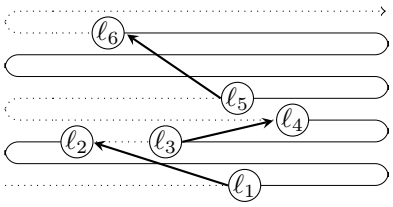
4 k-pass sweeping definability

We begin by defining the objects that need to be considered for characterizing *k-pass definability*, i.e., whether a sweeping transducer is equivalent to some k-pass sweeping transducer. Let S be an L-sweeping transducer. The idea is to consider factors of runs of S that can be simulated alternatively from left to right and from right to left. We begin by introducing a notion of inversion that looks symmetric to Definition 4: a *co-inversion* is defined as above, with $x_1 \leq x'_2$ replacing $x_2 \leq x'_1$. In other words, for an inversion we exclude the case where L_2 is *after* L_1, whereas for a co-inversion we exclude that L_2 is *before* L_1. We then combine inversions and co-inversions, as follows:

▶ **Definition 7.** A *k-inversion* of ρ is a sequence $\bar{\ell} = (\ell_1, \ell_2), \ldots, (\ell_{2k-1}, \ell_{2k})$ such that:

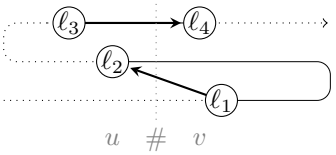
- $\ell_1 < \ell_2 < \ldots < \ell_{2k-1} < \ell_{2k}$ are distinct locations in ρ,
- for all even $i \in \{0, \ldots, k-1\}$, $(\ell_{2i+1}, \ell_{2i+2})$ is an inversion of ρ,
- for all odd $i \in \{0, \ldots, k-1\}$, $(\ell_{2i+1}, \ell_{2i+2})$ is a co-inversion of ρ.

An example of a 3-inversion is depicted to the right.



We say that $\bar{\ell}$ is *safe* if $\mathsf{out}(\rho[\ell_{2i+1}, \rho_{2i+2}])$ has period at most e_S, for *some* $i \in \{0, \ldots, k-1\}$. We denote by $L_S^{(k)}$ the language of words $u \in \mathrm{dom}(S)$ such that all k-inversions of all successful runs of S on u are safe.

▶ **Example 8.** Consider the 3-pass transducer that on input $u\#v$, with $u, v \in \{a, b\}^*$, outputs $(ab)^{|uvv|}(ba)^{|uuv|}$. This transduction can also be realized in 2 passes. This means that every 2-inversion is safe. For example, the 2-inversion depicted to the right is safe, as the output $\rho[\ell_3, \ell_4]$ is periodic.



Note that the definition of 1-inversion is the same as Definition 4, and hence $L_S^{(1)} = L_S$. In particular, by Theorem 6, we know that S is one-way definable iff $L_S^{(1)} = \mathrm{dom}(S)$. The generalization of this result is provided in Theorem 9 below: k-pass definability is equivalent to k-inversions being all safe, in the same way as one-way definability is equivalent to all inversions having periodic output.

▶ **Theorem 9.** *A sweeping transducer S is k-pass L-sweeping definable iff $L_S^{(k)} = dom(S)$, and this can be decided in* ExpSpace*. Moreover, given a sweeping transducer S, one can construct in* 2ExpTime *an unambiguous k-pass L-sweeping transducer T equivalent to $S|_{L_S^{(k)}}$.*

An analogous result for deciding k-pass *R-sweeping* definability can be derived by symmetry, by mirroring the input and reversing the computation. We also observe that, for $k = 1$, the above theorem improves the previous 2ExpSpace upper bound from [5] for deciding one-way definability of a sweeping transducer S. Concerning the doubly exponential size of an equivalent k-pass L-sweeping transducer, we observe that this is optimal, as in [5] we have shown that there are sweeping transducers S such that any equivalent one-way transducer has size at least doubly exponential in S.

Before turning to the proof of Theorem 9, we list some simple consequences of this theorem and of Propositions 1 and 3.

▶ **Corollary 10.**

- *One can compute in* ExpSpace *the minimum number of passes needed to implement a transduction given as a sweeping transducer.*

- *One can compute in* 3ExpSpace *the minimum number of reversals needed to implement a transduction given as a bounded-reversal two-way transducer. The complexity is* 2ExpSpace *if the given two-way transducer is unambiguous.*
- *One can compute in* 2ExpSpace *the minimum number of registers needed to implement a transduction given as a concatenation-free streaming transducer. The complexity is* ExpSpace *if the given streaming transducer is unambiguous.*

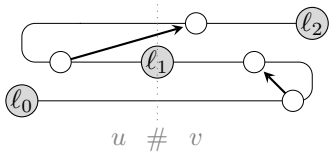
The proof of Theorem 9 is split into two parts. The first part, called "soundness", deals with the construction of the k-pass L-sweeping transducer T of the second claim. Since $L_S^{(k)} = \mathrm{dom}(S)$ implies that T is equivalent to S, this construction also proves the right-to-left direction of the first claim. Moreover, as a side result, we prove that whether $L_S^{(k)} = \mathrm{dom}(S)$ holds is decidable in ExpSpace. The second part, called "completeness", deals with the left-to-right direction of the first claim.

Soundness. We show how to construct from S a k-pass L-sweeping transducer T equivalent to $S|_{L_S^{(k)}}$. The idea is to consider a successful run ρ of S on a word $u \in L_S^{(k)}$, and divide it into k factors. We then simulate each factor of the run in a single pass, alternatively from left to right and from right to left, using [5]. First we need the notion of *k-factorizations*:

▶ **Definition 11.** A *k-factorization* of a successful run ρ of S is any sequence of locations $\overline{\ell} = \ell_0, \ell_1, \ldots, \ell_k$ of ρ such that:

- $\ell_0 \leq \ell_1 \leq \ldots \leq \ell_k$, ℓ_0 is the first location of ρ, and ℓ_k is the last location of ρ,
- for all even indexes i, with $0 \leq i < k$, and all inversions (ℓ, ℓ') of ρ, with $\ell_i \leq \ell \leq \ell' \leq \ell_{i+1}$, the word $\mathsf{out}(\rho[\ell, \ell'])$ has period at most e_S,
- for all odd indexes i, with $1 \leq i < k$, and all co-inversions (ℓ, ℓ') of ρ, with $\ell_i \leq \ell \leq \ell' \leq \ell_{i+1}$, the word $\mathsf{out}(\rho[\ell, \ell'])$ has period at most e_S.

▶ **Example 12.** We consider the transducer of Example 8 and we depict a 2-factorization of a run of it (gray nodes). All the inversions between ℓ_0 and ℓ_1, and all the co-inversions between ℓ_1 and ℓ_2, must be periodic. Note that the run does contain non-periodic inversions (e.g. those crossing ℓ_1), and hence it does not admit a 1-factorization.



The following lemma shows that we can reason equally in terms of safe k-inversions (Definition 7) and in terms of k-factorizations.

▶ **Lemma 13.** *For every word $u \in dom(S)$, we have that $u \in L_S^{(k)}$ if and only if all successful runs of S on u admit k-factorizations.*

Next we show that being a k-factorization is a *regular* property. To formalize this, we need to explain how to encode runs and sequences of locations as annotations of the underlying input. Formally, given a word $u \in \mathrm{dom}(S)$, a successful run ρ of S on u, and a tuple of locations $\overline{\ell} = \ell_1, \ldots, \ell_m$ in ρ, we denote by $\langle u, \rho, \overline{\ell} \rangle$ the word obtained by annotating each position $1 \leq x < |u|$ of u with the crossing sequence $\rho|x$ and with the m-tuple $\overline{y} = (y_1(x), \ldots, y_m(x))$, where each $y_i(x)$ is either the level of ℓ_i or $\perp$, depending on whether ℓ_i is at position x or not. Based on this encoding, we can define the language $F_S^{(k)}$ of all words of the form $\langle u, \rho, \overline{\ell} \rangle$, where ρ is a successful run of S on u and $\overline{\ell} = \ell_0, \ldots, \ell_k$ is a k-factorization of ρ. Lemma 14 below proves that this language is regular. In fact, in order to better handle the complexity of our characterization, the lemma shows that both $F_S^{(k)}$ and its complement $\overline{F}_S^{(k)}$ are recognized by automata of doubly exponential size.

▶ **Lemma 14.** *The language $F_S^{(k)}$ and its complement $\overline{F}_S^{(k)}$ are recognized by non-deterministic finite state automata of size double exponential w.r.t. S.*

Using the above encodings, we can also relativize the outputs produced by the transducer S to factors of successful runs. More precisely, we denote by S_{factors} the transducer that reads words of the form $\langle u, \rho, \ell_1, \ell_2\rangle$ and outputs words of the form $\mathsf{out}(\rho[\ell_1, \ell_2])$, provided that ρ is a successful run of S on u and ℓ_1, ℓ_2 are two locations in it. Note that S_{factors} does not check that the input is well-formed, in particular, that ρ is a successful run of S on u. Because of this, the number of states of S_{factors} is polynomial in the number of states of S, and a succinct representation of S_{factors} can be produced in polynomial time.

Now, it is easy to construct a k-pass L-sweeping transducer T equivalent to $S|_{L_S^{(k)}}$, as claimed in Theorem 9. The idea is that, on reading the input u, the transducer T guesses a successful run ρ on u and a k-factorization $\overline{\ell} = \ell_0, \ldots, \ell_k$ of ρ – this can be done using the encoding $\langle u, \rho, \overline{\ell}\rangle$ and Lemma 14. While guessing these objects, T performs k passes and outputs $T_0(\langle u, \rho, \ell_0, \ell_1\rangle) \cdot T_1(\langle u, \rho, \ell_1, \ell_2\rangle) \cdot \ldots \cdot T_{k-1}(\langle u, \rho, \ell_{k-1}, \ell_k\rangle)$, where each T_i is the 1-pass sweeping transducer obtained by applying Theorem 6 to S_{factor} (as usual, some mirroring is required for dealing with the odd indexes i). The only technical detail, here, is that different objects $\rho, \overline{\ell}$ may be guessed along the different passes of T. If this happens, the output produced by T might not be equal to that of S. We can overcome this problem by exploiting disambiguation, namely, by guessing canonical encodings $\langle u, \rho, \overline{\ell}\rangle$ in the language $F_S^{(k)}$. For example, we can fix a lexicographic ordering on these encodings and commit to always guessing the least encoding among those that agree on the input word u. This requires reasoning with both the language $F_S^{(k)}$ and its complement $\overline{F}_S^{(k)}$. By Lemma 14, the two languages are recognized by automata of doubly exponential size in S, and hence T can be constructed in doubly exponential time from S. As a matter of fact, the transducer T that we just constructed is also unambiguous.

We conclude this part by showing how to decide in exponential space if $L_S^{(k)} = \text{dom}(S)$. In fact, as we already know that $L_S^{(k)} \subseteq \text{dom}(S)$, it suffices to decide only the containment $L_S^{(k)} \supseteq \text{dom}(S)$. We know from Lemma 13 that the language $L_S^{(k)}$ coincides with the projection of $F_S^{(k)}$ on the underlying words u. Thus, we have $L_S^{(k)} \supseteq \text{dom}(S)$ if and only if $\overline{F}_S^{(k)} \cap D = \emptyset$, where $D = \{\langle u, \rho, \overline{\ell}\rangle \ : \ u \in \text{dom}(S)\}$. A close inspection of the construction of the automaton for $\overline{F}_S^{(k)}$ shows that the emptiness of $\overline{F}_S^{(k)} \cap D$ can be decided in ExpSpace.

Completeness. Here we prove the left-to-right direction of the first claim of Theorem 9. We suppose that S is an L-sweeping transducer and T is an equivalent k-pass L-sweeping transducer. We fix, once and for all, a successful run ρ of S on u and a k-inversion $\overline{\ell} = (\ell_1, \ell_2), \ldots, (\ell_{2k-1}, \ell_{2k})$ of ρ.

The goal is to prove that $\overline{\ell}$ is safe, namely, that the factor of the output produced between the locations of some (co-)inversion $(\ell_{2i+1}, \ell_{2i+2})$ of $\overline{\ell}$ is periodic, with uniformly bounded period. The main idea is to try to find a factor $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ that is entirely covered by the output produced along a single pass of the equivalent transducer T, and apply a suitable generalization of Proposition 5. Informally, this works by pumping the output $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ through repeating the witnessing loops of $(\ell_{2i+1}, \ell_{2i+2})$. In a similar way, we pump the output produced along a single pass of T. Then, by analyzing how the former outputs are covered by the latter outputs, we deduce the periodicity of $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$.

The main difficulty in formalizing the above idea lies in the fact that the k passes of the supposed transducer T cannot be identified directly on the run ρ of S. Therefore we need to reason in a proper way about *families* of factors associated with (co-)inversions inside pumped runs. Below, we introduce some terminology and notation to ease this task.

Recall that $\overline{\ell} = (\ell_1, \ell_2), \ldots, (\ell_{2k-1}, \ell_{2k})$ is a k-inversion of the run ρ. For all $i = 0, \ldots, k-1$, let L_{2i+1} and L_{2i+2} be the witnessing loops of $(\ell_{2i+1}, \ell_{2i+2})$. For a given tuple of numbers $\overline{n} = (n_1, \ldots, n_{2k}) \in \mathbb{N}^{2k}$, we define $\rho^{\overline{n}} = \mathsf{pump}^{\overline{n}}_{\overline{L}}(\rho)$, where $\overline{L} = L_1, \ldots, L_{2k}$ and $\overline{n} = n_1, \ldots, n_{2k}$ (recall that this is the run obtained by pumping the loops $L_1, \ldots, L_{2k}$ respectively $n_1, \ldots, n_{2k}$ times, as described in Section 3). Similarly, we denote by $u^{\overline{n}}$ the word parsed by the pumped run $\rho^{\overline{n}}$.

We would like to map the inversions and co-inversions of $\overline{\ell}$ on the pumped runs $\rho^{\overline{n}}$. Consider an inversion $(\ell_{2i+1}, \ell_{2i+2})$, for some $i \in \{1, \ldots, 2k\}$ (the case of a co-inversion is similar). Recall that when pumping loops in ρ, several copies of the original locations may be introduced. In particular, among the copies of the inversion $(\ell_{2i+1}, \ell_{2i+2})$ that appear in the pumped run $\rho^{\overline{n}}$, we will consider the maximal one, which is identified by taking the *first* copy $\tilde{\ell}_{2i+1}$ of ℓ_{2i+1} and the *last* copy $\tilde{\ell}_{2i+2}$ of ℓ_{2i+2}. For the sake of brevity, we say that $(\tilde{\ell}_{2i+1}, \tilde{\ell}_{2i+2})$ is the inversion of $\rho^{\overline{n}}$ that *corresponds* to $(\ell_{2i+1}, \ell_{2i+2})$.

We can now define the key objects for our reasoning, that is, the factors of the output of a pumped run $\rho^{\overline{n}}$ that correspond in the original run ρ to the factors produced between the locations of the (co-)inversions of $\overline{\ell}$. Formally, for every $2k$-tuple $\overline{n}$ of natural numbers and every index $i = 0, \ldots, k-1$, we define

$$v^{\overline{n}}(i) \;=\; \mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_{2i+1}, \tilde{\ell}_{2i+2}]\big)$$

where $(\tilde{\ell}_{2i+1}, \tilde{\ell}_{2i+2})$ is the (co-)inversion of $\rho^{\overline{n}}$ that corresponds to $(\ell_{2i+1}, \ell_{2i+2})$. Below we highlight the relevant factors inside the output produced by S on $u^{\overline{n}}$:

$$S(u^{\overline{n}}) \;=\; \mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_0, \tilde{\ell}_1]\big) \cdot \boldsymbol{v^{\overline{n}}(0)} \cdot \mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_2, \tilde{\ell}_3]\big) \cdot \boldsymbol{v^{\overline{n}}(1)} \cdot \ldots \cdot \boldsymbol{v^{\overline{n}}(k-1)} \cdot \mathsf{out}\big(\rho^{\overline{n}}[\tilde{\ell}_{2k}, \tilde{\ell}_{2k+1}]\big) \quad (1)$$

where $\tilde{\ell}_0$ is the first location of $\rho^{\overline{n}}$, $\tilde{\ell}_{2k+1}$ is the last location of $\rho^{\overline{n}}$.

In a similar way, we can factorize the output produced by the k-pass L-sweeping transducer T when reading the input $u^{\overline{n}}$. However, the focus here is on the factors of the output produced along each pass. Formally, given $\overline{n} \in \mathbb{N}^{2k}$, we let $\sigma^{\overline{n}}$ be some successful run of T on $u^{\overline{n}}$. For every $j = 0, \ldots, k-1$, we let ℓ'_j be the first location of $\sigma^{\overline{n}}$ at level j. We further let ℓ'_k be the last location of $\sigma^{\overline{n}}$, which is at level $k-1$. We then define

$$w^{\overline{n}}(j) \;=\; \mathsf{out}\big(\sigma^{\overline{n}}[\ell'_j, \ell'_{j+1}]\big)$$

and factorize the output of T on $u^{\overline{n}}$ as follows:

$$T(u^{\overline{n}}) \;=\; \boldsymbol{w^{\overline{n}}(0)} \cdot \boldsymbol{w^{\overline{n}}(1)} \cdot \ldots \cdot \boldsymbol{w^{\overline{n}}(k-1)}. \quad (2)$$

The next step is to exploit the hypothesis that S and T are equivalent. This means that Equations (1) and (2) represent the same word. From this we derive that, for any given $\overline{n} \in \mathbb{N}^{2k}$, at least one of the words $v^{\overline{n}}(i)$ highlighted in Equation (1) is a factor of the word $w^{\overline{n}}(i)$ highlighted in Equation (2). However, what is the index i for which this coverability relation holds depends on the parameter $\overline{n}$. In order to enable a reasoning similar to that of Proposition 5, we need to find a single index i such that, for "sufficiently many" parameters $\overline{n}$, $v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$. The definition below, formalizes what we mean precisely by "sufficiently many" $\overline{n}$ – intuitively, we require that specific coordinates of $\overline{n}$ are unbounded, as well the differences between these coordinates.

▶ **Definition 15.** Let $\mathcal{P}(\overline{n})$ denote an arbitrary property of tuples $\overline{n} \in \mathbb{N}^{2k}$. Further let h, h' be two distinct coordinates in $\{1, \ldots, 2k\}$. We say that $\mathcal{P}(\overline{n})$ holds *unboundedly on the coordinates* h, h' *of* $\overline{n}$ if, for all numbers $n_0 \in \mathbb{N}$, there exist $\overline{n}_1, \overline{n}_2 \in \mathbb{N}^{2k}$ such that:

- $\mathcal{P}(\overline{n}_1)$ and $\mathcal{P}(\overline{n}_2)$ hold,
- $\overline{n}_1[h] \geq n_0$ and $\overline{n}_1[h'] - \overline{n}_1[h] \geq n_0$,
- $\overline{n}_2[h'] \geq n_0$ and $\overline{n}_2[h] - \overline{n}_2[h'] \geq n_0$.

We recall that each factor $v^{\overline{n}}(i)$ is associated with the (co-)inversion $(\ell_{2i+1}, \ell_{2i+2})$, and that the corresponding components $\overline{n}[2i+1]$ and $\overline{n}[2i+2]$ of the parameter $\overline{n}$ denote the number of times the witnessing loops L_{2i+1} and L_{2i+2} are pumped in $\rho^{\overline{n}}$. The specific properties we are interested in are the following ones, for $i = 0, \ldots, k-1$:

$\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ *is a factor of* $w^{\overline{n}}(i)$".

It is not difficult to see that for every tuple $\overline{n} \in \mathbb{N}^{2k}$, $\mathcal{P}_i(\overline{n})$ holds for some $i \in \{0, \ldots, k-1\}$. From this, using a suitable counting argument, we can prove the crucial lemma below.

▶ **Lemma 16.** *There exists an index* $i \in \{0, \ldots, k-1\}$ *such that the property* $\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$" *holds unboundedly on the coordinates* $2i+1$ *and* $2i+2$ *of* $\overline{n}$.

The last piece of the puzzle consists of generalizing the statement of Proposition 5. The idea is that we can replace the hypothesis that S is one-way definable by the weaker assumption of Lemma 16. That is, if $\mathcal{P}_i(\overline{n})$ holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$, we can still use the same arguments based on pumping and Fine-Wilf's Theorem as in Proposition 5, in order to deduce that the output $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ between the locations of the (co-)inversion is periodic:

▶ **Proposition 17.** *If the property* $\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$" *holds unboundedly on the coordinates* $2i+1$ *and* $2i+2$ *of* $\overline{n}$*, then the output* $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ *produced between the locations of the (co-)inversion* $(\ell_{2i+1}, \ell_{2i+2})$ *is periodic, with period* e_S*. In particular, the* k*-inversion* $\overline{\ell} = (\ell_1, \ell_2), \ldots, (\ell_{2k-1}, \ell_{2k})$ *is safe.*

To conclude, we assumed that the L-sweeping transducer S is equivalent to a k-pass L-sweeping transducer T. We considered a successful run ρ of S and an arbitrary k-inversion $\overline{\ell}$ of it. By Lemma 16, we know that there is an index $i \in \{0, \ldots, k-1\}$ for which the property $\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ *is a factor of* $w^{\overline{n}}(i)$" holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$. From this, by applying Proposition 17, we derive that the k-inversion $\overline{\ell}$ is safe. This proves the left-to-right direction of the first claim of Theorem 9. ◀

5 Sweeping transducers and MSO

We provide a logical characterization of sweeping transducers. For this we will consider restricted forms of transductions definable in monadic-second order logic (MSO) [8].

MSO transductions are described by specifying the output (seen as a relational structure) from a fixed number of copies of the input. Formally, an *MSO transduction with m copies* consists of an MSO sentence Φ_{dom}, some unary MSO formulas $\Phi^i_a(x)$, one for each $i \in \{1, \ldots, m\}$ and $a \in \Delta$, and some binary MSO formulas $\Phi^{i,j}_{\leq}(x, y)$, one for each $i, j \in \{1, \ldots, m\}$. Intuitively, the sentence Φ_{dom} tells whether the transduction is defined on some input u. The unary formula $\Phi^i_a(x)$ tells whether the element x of the i-th copy of the input belongs to the output and is labeled with the letter a. The formula $\Phi^{i,j}_{\leq}(x, y)$ tells whether, in the produced output, the element x of the i-th copy of the input precedes the element y of the j-th copy of the input. Note that the sentence Φ_{dom} can easily guarantee that, whenever the output is defined, it is well-formed, namely, it is a word. For the sake of simplicity, we assume that $\Phi^i_a(x)$ *entails* Φ_{dom}, namely, for all words u and all positions x, $u \models \Phi^i_a(x)$ implies $u \models \Phi_{\text{dom}}$. Similarly, we assume that $\Phi^{i,j}_{\leq}(x, y)$ entails $\Phi^i(x)$ and $\Phi^j(y)$.

▶ **Definition 18.** Let T be an MSO transduction with m copies. We say that T is

- *order-preserving* if each formula $\Phi^{i,j}_{\leq}(x,y)$ entails $x \leq y$;
- *order-inversing* if each formula $\Phi^{i,j}_{\leq}(x,y)$ entails $x \geq y$;
- *k-phase* if there is a partition $I_0, I_1, \ldots, I_{k-1}$ of the set of indexes $\{1,\ldots,m\}$ such that $I_0 < I_1 < \ldots < I_{k-1}$, namely, $i < j$ for all $0 \leq h < h' < k$, $i \in I_h$, and $j \in I_{h'}$, and each formula $\Phi^{i,j}_{\leq}(x,y)$ entails $x \leq y$ if h is even, or $x \geq y$ otherwise.

We know from [9] that order-preserving MSO transductions capture precisely the one-way definable transductions. We obtain:

▶ **Theorem 19.** *k-phase MSO transductions have the same expressive power as functional, k-pass L-sweeping transducers.*

6 Conclusions

We showed that sweeping transducers, bounded-reversal transducers, and concatenation-free streaming transducers define the same subclass of regular word transductions. Our main result is an effective characterization of transductions definable by sweeping transducers with a fixed number of passes. As a consequence we obtained a procedure that minimizes the number of registers in a concatenation-free sweeping transducer.

We believe that similar results can be proven for two-way (non-sweeping) transducers, using a refined version of the constructions presented here. In this respect, an interesting open problem is to characterize the two-way transducers that are equivalent to sweeping transducers, but with an arbitrary (unspecified) number of passes.

References

1. A. V. Aho and J. D. Ullman. A characterization of two-way deterministic classes of languages. *J. Comput. Syst. Sci.*, 4(6):523–538, 1970.
2. R. Alur and P. Cerný. Expressiveness of streaming string transducers. In *FSTTCS*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2010. doi:10.4230/LIPIcs.FSTTCS.2010.1.
3. R. Alur and L. D'Antoni. Streaming tree transducers. In *ICALP*, volume 7392 of *LNCS*, pages 42–53. Springer, 2012.
4. R. Alur and M. Raghothaman. Decision problems for additive regular functions. In *ICALP*, volume 7966 of *LNCS*, pages 37–48. Springer, 2013.
5. F. Baschenis, O. Gauwin, A. Muscholl, and G. Puppis. One-way definability of sweeping transducers. In *FSTTCS*, volume 45 of *LIPIcs*, pages 178–191. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.FSTTCS.2015.178.
6. O. Carton and L. Dartois. Aperiodic two-way transducers and FO-transductions. In *CSL*, LIPIcs, pages 160–174. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.CSL.2015.160.
7. L. Daviaud, P.-A. Reynier, and J.-M. Talbot. A generalised twinning property for minimisation of cost register automata. In *LICS*. IEEE Computer Society, 2016.
8. J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Logic*, 2:216–254, 2001.
9. E. Filiot. Logic-automata connections for transformations. In *ICLA*, pages 30–57, 2015.
10. E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais. From two-way to one-way finite state transducers. In *LICS*, pages 468–477. IEEE Computer Society, 2013.

11 E. Filiot, S. N. Krishna, and A. Trivedi. First-order definable string transformations. In *FSTTCS*, volume 29 of *LIPIcs*, pages 147–159. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.147.

12 J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.

13 M. Sipser. Lower bounds on the size of sweeping automata. In *STOC*, pages 360–364. ACM, 1979.

A Linear Acceleration Theorem for 2D Cellular Automata on All Complete Neighborhoods

Anaël Grandjean[1] and Victor Poupet[2]

1 **LIRMM, Université Montpellier, Montpellier, France**
`anael.grandjean@lirmm.fr`
2 **LIRMM, Université Montpellier, Montpellier, France**
`victor.poupet@lirmm.fr`

Abstract

Linear acceleration theorems are known for most computational models. Although such results have been proved for two-dimensional cellular automata working on specific neighborhoods, no general construction was known. We present here a technique of linear acceleration for all two-dimensional languages recognized by cellular automata working on complete neighborhoods.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases 2D Cellular automata, linear acceleration, language recognition

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.115

1 Introduction

Cellular automata (CA) were initially introduced by S. Ulam and J. von Neumann [14] in the 1960s to study self-reproduction in discrete dynamical systems. They are massively parallel systems consisting of an infinite array of cells. Cells evolve synchronously depending on the states of their neighbors according to a uniform deterministic rule. Although initially considered in two dimensions, the definition can be adapted to any dimensional cellular space and even more general uniform graphs [8].

Soon after their introduction, they were shown to be computationally universal [9, 1]. As a computation model, they have been extensively studied as one-dimensional language recognizers [11, 3] but are also very well suited to the study of two-dimensional "picture languages" [10, 12, 13].

The neighborhood of a cellular automaton defines the underlying communications graph of the cells. Although most of the existing work on two-dimensional cellular automata focuses on the von Neumann (4 closest neighbors) and Moore (8 closest neighbors) neighborhoods, understanding how the choice of the neighborhood affects the algorithmic capabilities of the model is a key to understanding parallel computation.

Linear acceleration theorems are well known for most of the commonly considered computation models. It was first proved for one-dimensional cellular automata working on the standard neighborhood and two-dimensional cellular automata on von Neumann's neighborhood by W. T. Beyer [2], inspired by similar constructions for sequential input cellular automata [3, 5, 6]. The one-dimensional case was later generalized by J. Mazoyer and N. Reimen for arbitrary neighborhoods [7]. As for two-dimensional neighborhoods, V. Terrier extended the construction to the Moore neighborhood [12] and then to the slightly more general class of neighborhoods whose convex hull has at most one vertex in the positive quarter plane [13].

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 115; pp. 115:1–115:12


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

In this paper, we prove a general linear acceleration result for all complete neighborhoods on two-dimensional cellular automata.

The main theorem is stated in Section 3 and proved in Sections 4 to 7. Sections 4 and 6 describe the two main elements of the construction (compression of the input and accelerated simulation of the original automaton respectively). Section 5 presents a technique to perform a sequence of tasks on a cellular automaton without the need for synchronization at the start of each new task, used in the proof of the theorem to combine sections 4 and 6. Although this technique is elementary and has been used in previous publications (a special case was used by W. T. Beyer in 1969 [2]), reviews of previous articles seem to indicate that it is not common knowledge. It is therefore presented here in a separate section and stated generally in the hopes that it can be easily reused in future publications.

The construction presented in this article is similar in several ways to previously published constructions, most notably those of V. Terrier in [13]. Significant improvements include compression of the input in *almost* optimal time (Section 4, specifically Subsection 4.2) and a more general simulation technique (Section 6).

2 Definitions

2.1 Cellular Automata

▶ **Definition 1** (Cellular Automaton)**.** A *cellular automaton* (CA) is a quadruple $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$ where

- $d \in \mathbb{N}$ is the dimension of the automaton ;
- $\mathcal{Q}$ is a finite set whose elements are called *states* ;
- $\mathcal{N}$ is a finite subset of $\mathbb{Z}^d$ called *neighborhood* of the automaton ;
- $\delta : \mathcal{Q}^{\mathcal{N}} \to \mathcal{Q}$ is the *local transition function* of the automaton.

▶ **Definition 2** (Configuration)**.** A *d-dimensional configuration* $\mathfrak{C}$ over the set of states $\mathcal{Q}$ is a mapping from $\mathbb{Z}^d$ to $\mathcal{Q}$. The elements of $\mathbb{Z}^d$ will be referred to as *cells*.

Given a CA $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, a configuration $\mathfrak{C} \in \mathcal{Q}^{\mathbb{Z}^d}$ and a cell $c \in \mathbb{Z}^d$, we denote by $\mathcal{N}_{\mathfrak{C}}(c)$ the neighborhood of c in $\mathfrak{C}$:

$$\mathcal{N}_{\mathfrak{C}}(c) : \left\{ \begin{array}{ccc} \mathcal{N} & \to & \mathcal{Q} \\ n & \mapsto & \mathfrak{C}(c+n) \end{array} \right.$$

From the local transition function δ of a CA $\mathcal{A} = (d, \mathcal{Q}, \mathcal{N}, \delta)$, we can define the *global transition function of the automaton* $\Delta : \mathcal{Q}^{\mathbb{Z}^d} \to \mathcal{Q}^{\mathbb{Z}^d}$ obtained by applying the local rule on all cells :

$$\Delta(\mathfrak{C}) = \left\{ \begin{array}{ccc} \mathbb{Z}^d & \to & \mathcal{Q} \\ c & \mapsto & \delta(\mathcal{N}_{\mathfrak{C}}(c)) \end{array} \right.$$

The action of the global transition rule makes $\mathcal{A}$ a dynamical system over the set $\mathcal{Q}^{\mathbb{Z}^d}$. Because of this dynamic, in the following we will identify the CA $\mathcal{A}$ with its global rule so that $\mathcal{A}(\mathfrak{C})$ is the image of a configuration $\mathfrak{C}$ by the action of the CA $\mathcal{A}$, and more generally $\mathcal{A}^t(\mathfrak{C})$ is the configuration resulting from applying t times the global rule of the automaton from the initial configuration $\mathfrak{C}$.

▶ **Definition 3** (Quiescent and Permanent States)**.** For a given CA $\mathcal{A}$, we say that a state q is *quiescent* if a cell in state q remains in this state if all its neighbors are also in q. We say that q is *permanent* if a cell in state q remains in that state regardless of the state of its neighbors.

In this article we will only consider 2-dimensional cellular automata (2DCA). From now on the set of cells will always be $\mathbb{Z}^2$.

2.2 Neighborhoods

Throughout the article, we use the additive notation for vector sums, the power notation for neighborhood composition and the product notation for scalar product:

▶ **Definition 4** (Vector Sum). Given two neighborhoods $\mathcal{N}_1$ and $\mathcal{N}_2$ and a cell $c \in \mathbb{Z}^2$, we define the vector sums $\mathcal{N}_1 + \mathcal{N}_2 = \{x + y \mid x \in \mathcal{N}_1, y \in \mathcal{N}_2\}$ and $c + \mathcal{N}_1 = \{c + x \mid x \in \mathcal{N}_1\}$.

▶ **Definition 5** (Neighborhood Powers). Given a neighborhood $\mathcal{N}$, we define

$$\mathcal{N}^0 = \{0\} \tag{1}$$

$$\forall k \in \mathbb{N}, \quad \mathcal{N}^{k+1} = \mathcal{N} + \mathcal{N}^k \tag{2}$$

▶ **Definition 6** (Scalar product). Given a neighborhood $\mathcal{N}$ and an integer $k \in \mathbb{Z}$, we define the scalar product $k\mathcal{N} = \{kx \mid x \in \mathcal{N}\}$.

▶ **Definition 7** (Complete Neighborhood). A neighborhood $\mathcal{N}$ is said to be *complete* if

$$\bigcup_{k \in \mathbb{N}} \mathcal{N}^k = \mathbb{Z}^2$$

▶ **Definition 8** (Convex Hull and Convex Neighborhood). The *convex hull* of a neighborhood $\mathcal{N}$ is the smallest convex polygon $\mathrm{CH}(\mathcal{N}) \subset \mathbb{R}^2$ such that $\mathcal{N} \subseteq \mathrm{CH}(\mathcal{N})$. Moreover a neighborhood $\mathcal{N}$ is said to be *convex* if it contains all points of integer coordinates in its convex hull: $\mathcal{N} = \mathrm{CH}(\mathcal{N}) \cap \mathbb{Z}^2$.

▶ Remark. If $\mathcal{N}$ is a convex neighborhood, $\mathcal{N}^p$ is also convex for any $p \in \mathbb{N}$.

2.3 Two-Dimensional Language Recognition

▶ **Definition 9** (Picture). For $n, m \in \mathbb{N}$ and Σ a finite alphabet, an (n, m)-*picture* (picture of width n and height m) over Σ is a mapping

$$p : [\![0, n-1]\!] \times [\![0, m-1]\!] \to \Sigma$$

$\Sigma^{n,m}$ denotes the set of all (n, m)-pictures over Σ and $\Sigma^{*,*} = \bigcup_{n,m \in \mathbb{N}} \Sigma^{n,m}$ the set of all pictures over Σ. A *picture language* over Σ is a set of pictures over Σ.

▶ **Definition 10** (Picture Configuration). Given an (n, m)-picture p over Σ, we define the *picture configuration* associated to p with quiescent state $q_0 \notin \Sigma$ as

$$\mathfrak{C}_{p,q_0} : \begin{cases} \mathbb{Z}^2 & \to & \Sigma \cup \{q_0\} \\ x, y & \mapsto & \begin{cases} p(x, y) & \text{if } (x, y) \in [\![0, n-1]\!] \times [\![0, m-1]\!] \\ q_0 & \text{otherwise} \end{cases} \end{cases}$$

▶ **Definition 11** (Picture Recognizer). Given a picture language L over an alphabet Σ, we say that a 2DCA $\mathcal{A} = (2, \mathcal{Q}, \mathcal{N}, \delta)$ such that $\Sigma \subseteq \mathcal{Q}$ recognizes L with quiescent state $q_0 \in \mathcal{Q} \setminus \Sigma$, accepting state $q_a \in \mathcal{Q}$ and rejecting state $q_r \in \mathcal{Q}$ in time $\tau : \mathbb{N}^2 \to \mathbb{N}$ if q_a and q_r are permanent states and for any picture p (of size $n \times m$), starting from the picture configuration $\mathfrak{C}_{p,q_0}$ at time 0, the origin cell of the automaton at time $\tau(n, m)$ is in state q_a if $p \in L$ and state q_r if $p \notin L$.

▶ **Definition 12** (Real Time). Given a complete neighborhood $\mathcal{N}$, the real time function $\mathrm{RT}_{\mathcal{N}} : \mathbb{N}^2 \to \mathbb{N}$ associated to $\mathcal{N}$ is defined as

$$\mathrm{RT}_{\mathcal{N}}(n, m) = \min\{t \mid [\![0, n-1]\!] \times [\![0, m-1]\!] \subseteq \mathcal{N}^t\}$$

3 The Main Theorem

Most of the article will be dedicated to the proof of the following theorem

▶ **Theorem 13** (Linear Acceleration). *For any complete neighborhood $\mathcal{N}$, any real number $\epsilon > 0$, any finite alphabet Σ and any language $L \subseteq \Sigma^{*,*}$, if L is recognized by a 2DCA working on $\mathcal{N}$ in time*

$$(n, m) \mapsto \mathrm{RT}_{\mathcal{N}}(n, m) + f(n, m)$$

for some function $f : \mathbb{N}^2 \to \mathbb{N}$ then L can be recognized in time

$$(n, m) \mapsto \lceil (1 + \epsilon) \mathrm{RT}_{\mathcal{N}}(n, m) + \epsilon f(n, m) \rceil$$

by a 2DCA with neighborhood $\mathcal{N}$.

▶ **Corollary 14.** *For any complete neighborhood $\mathcal{N}$, any language recognized in time $(n, m) \mapsto k \mathrm{RT}_{\mathcal{N}}(n, m)$ for some $k > 1$ can be recognized in time $(n, m) \mapsto (1 + \epsilon) \mathrm{RT}_{\mathcal{N}}(n, m)$ for any real number $\epsilon > 0$.*

To prove Theorem 13, we consider a 2DCA $\mathcal{A}$ working on a complete neighborhood $\mathcal{N}$ and describe the construction of a 2DCA $\mathcal{A}'$ working on the same neighborhood that simulates the behavior of $\mathcal{A}$ in a way that enables it to recognize the same language as $\mathcal{A}$ in a linearly shorter time.

3.1 Preliminary Remarks

The following observations will greatly simplify the proof of Theorem 13.

▶ **Claim 15.** *It is sufficient to prove Theorem 13 up to an additive constant, meaning that we only need to prove that L can be recognized in time*

$$(n, m) \mapsto (1 + \epsilon) \mathrm{RT}_{\mathcal{N}}(n, m) + \epsilon f(n, m) + O(1)$$

Proof. Consider that we have this weaker result. To get rid of the $O(1)$ term simply choose $\epsilon' < \epsilon$. For any $C > 0$, for (n, m) large enough we have

$$(1 + \epsilon) \mathrm{RT}_{\mathcal{N}}(n, m) + \epsilon f(n, m) > (1 + \epsilon') \mathrm{RT}_{\mathcal{N}}(n, m) + \epsilon' f(n, m) + C$$

The automaton can handle all the finitely many inputs of small size in real time. ◀

▶ **Claim 16.** *It is sufficient to prove Theorem 13 for all complete convex neighborhoods.*

Proof. Consider a complete neighborhood $\mathcal{N}$ and let $\mathcal{N}'$ be the convex neighborhood having same convex hull as $\mathcal{N}$. The real time functions $\mathrm{RT}_{\mathcal{N}}$ and $\mathrm{RT}_{\mathcal{N}'}$ differ by at most a constant. Moreover a CA on $\mathcal{N}$ can simulate the behavior of a CA on $\mathcal{N}'$ with a loss of at most a constant number of steps and conversely (see [4] for more details).

The property from the theorem therefore translates directly from one neighborhood to the other with at most a constant difference that can be ignored according to Claim 15. ◀

From now on we will consider that $\mathcal{N}$ is a convex neighborhood.

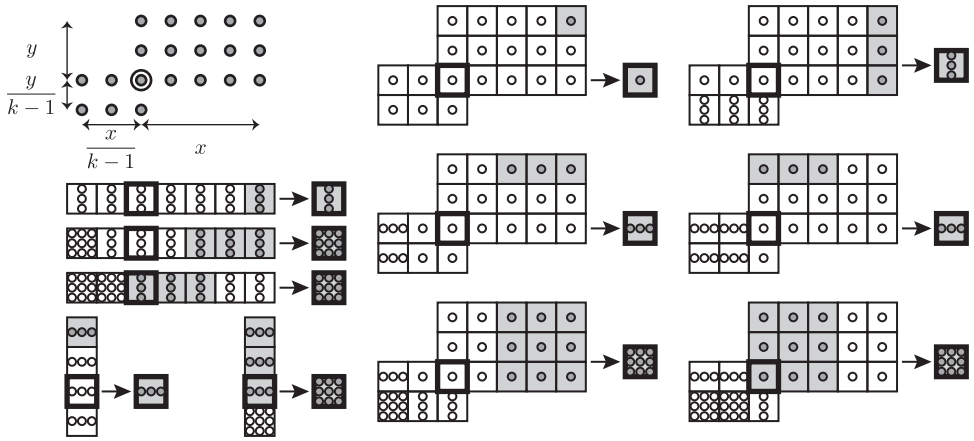


Figure 1 Rules for the compression by a factor 3 of inputs of ratio $\frac{n}{m} = 2$ with the neighborhood represented in the top left. The information that the cell takes as its new state is represented in grey. Information travels towards the bottom left. By looking down and left a cell determines if the information from the top right should simply pass through (first case) or if some of its neighbors are already full in which case it should start packing information. Left column shows simplified rules for which the cell has already packed information in one of the directions and therefore only the neighbors in the remaining direction are significant.

4 Compression of the Input

The first phase of the construction is to compress the input by a factor $k > \frac{1}{\epsilon}$. We want to move the states of the initial configuration towards the origin, packing them in groups of $k \times k$ as illustrated by Figure 2.

Although such compressions are relatively simple to perform on the von Neumann and Moore neighborhoods, on a more general neighborhood it is not possible to know in which direction the information should travel to move towards the origin at optimal speed. In general, the optimal travel direction depends on the proportion $\frac{n}{m}$ of the input.

We first show that if the proportion of the input is fixed, compression can be done in optimal time on any complete neighborhood. Then, by performing a finite number of compressions in parallel, each assuming a different proportion, we show that any input is close enough to one of these assumed proportions to be compressed in "nearly optimal" time, which will be sufficient for the proof of Theorem 13.

4.1 Compression of an Input of Constrained Proportion

If the size of the input is known to be of proportion $\frac{n}{m} = \alpha$ for some fixed rational α, we can perform a compression by a factor k with a neighborhood such as the one illustrated on the top left of Figure 1 with $\frac{x}{y} = \alpha$. On such a neighborhood, compressing the input is simply a matter of transferring the states from the top right to the bottom left, packing them in groups of $(1 \times k)$, $(k \times 1)$ or $(k \times k)$ when they cannot go any further in one or both directions (see Figures 1 and 2). The compression is completed in time $\lceil \frac{k-1}{k} \mathrm{RT} \rceil$.

Note that in order to compress by a factor k, the cell must be able to see $\frac{1}{k-1}$ times as far towards the left and bottom as it sees towards the right and top in order to properly determine when it should start packing states. Because the convex hull of a complete neighborhood $\mathcal{N}$ contains an open set around the origin (otherwise $(\mathcal{N}^p)_{p\in\mathbb{N}}$ would not expand in all directions), it contains its homothetic image of ratio $-\frac{1}{k_0-1}$ for some k_0 (see Figure 3). On such a neighborhood compression by any factor $k \geq k_0$ is possible.

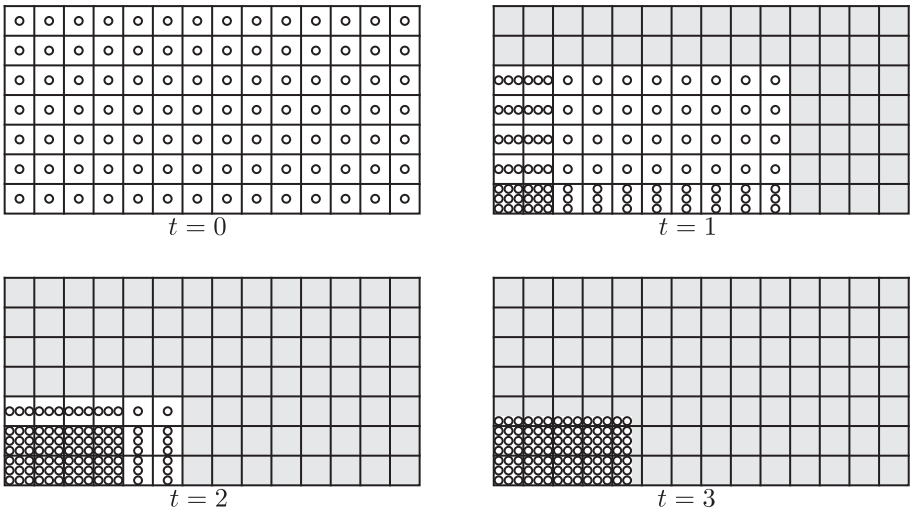


Figure 2 Compression of an input of size (14×7) with the neighborhood and rules from Figure 1.

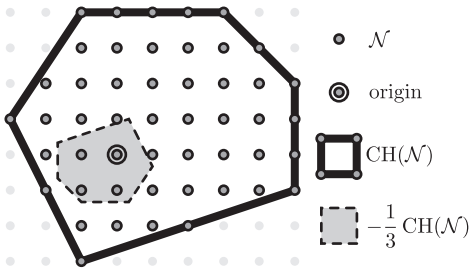


Figure 3 An example neighborhood on which a compression by a factor $k = 4$ is possible.

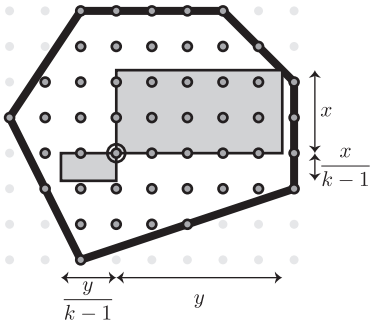


Figure 4 The area of the neighborhood that will be used for the compression of inputs of proportion $\frac{n}{m} = 2$ by a factor $k = 4$.

To compress inputs of proportion $\frac{n}{m} = \alpha$ on some complete neighborhood $\mathcal{N}$, we consider the largest rectangle $[0, x] \times [0, y]$ with $x, y \in \mathbb{Q}$ and $\frac{x}{y} = \alpha$ included in the convex hull of $\mathcal{N}$ (see Figure 4). For all $k \geq k_0$, the rectangle $[-\frac{x}{k-1}, 0] \times [-\frac{y}{k-1}, 0]$ is also in the convex hull of $\mathcal{N}$.

These rectangles have rational but not necessarily integer dimensions. If we consider the neighborhood $\mathcal{N}^p$ for some large p, all is scaled up by a factor p and the corresponding rectangles can be made of integer dimensions. The real time function on inputs of proportion α for $\mathcal{N}^p$ is equal to the real time function of the neighborhood containing only the two rectangles (of integer coordinates). The compression algorithm described by Figures 1 and 2 therefore finishes in time $\frac{k-1}{k}\,\mathrm{RT}_{\mathcal{N}^p} + O(1)$ on $\mathcal{N}^p$. An automaton working on $\mathcal{N}$ can simulate one step of an automaton working on $\mathcal{N}^p$ in p time steps, and since $\mathrm{RT}_{\mathcal{N}^p} = \lceil \frac{1}{p}\,\mathrm{RT}_{\mathcal{N}} \rceil$, the compression can be completed on $\mathcal{N}$ in time $\frac{k-1}{k}\,\mathrm{RT}_{\mathcal{N}} + O(1)$.

4.2 Compression of General Input

Let us now consider inputs of arbitrary proportions. As discussed in the previous subsection, for inputs of proportion α the optimal direction in which the information should travel for

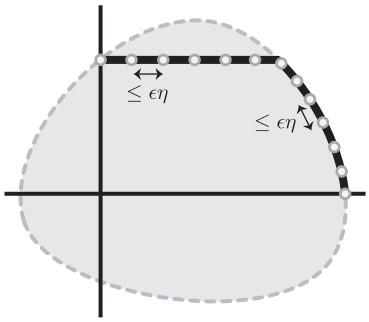


Figure 5 Choosing the set S of proportions that will be used by $\mathcal{A}'$ for compressions. The thick line is the set of corners of maximal rectangles for all possible proportions. Along this line, we pick a finite set of points at intervals of at most $\epsilon\eta$.

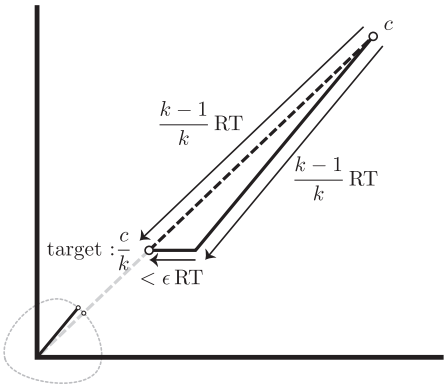


Figure 6 Compression of an input of arbitrary proportion. The optimal direction for the compression is represented by a dashed line. The closest chosen direction is represented by a solid line inside the neighborhood. It is at a distance at most $\epsilon\eta$ of the optimal vector. The travel path of the farthest cell of the input is represented as a solid black line, made of an initial segment along the almost-optimal direction and an extra segment to compensate for the deviation.

a compression is defined by the diagonal of the largest rectangle $[0, x] \times [0, y]$ with $\frac{x}{y} = \alpha$ included in the convex hull of $\mathcal{N}$. The first thing to note is that since $\mathcal{N}$ is complete, there exists $\eta > 0$ such that $[0, \eta] \times [0, \eta] \subseteq \mathrm{CH}(\mathcal{N})$ and hence all maximal rectangles in $\mathrm{CH}(\mathcal{N})$ have at least one dimension greater than η. The corners (x, y) of such maximal rectangles all lie on a line. Let us pick a finite set S of rational points on this line from one extremity to the other with distance at most $\epsilon\eta$ between two consecutive points (see Figure 5).

For each proportion $\alpha = \frac{x}{y}$ with $(x, y) \in S$, $\mathcal{A}'$ performs a compression of the input as described in the previous subsection. All compressions take place at the same time in parallel. Note that even if the proportion of the input is not exactly that for which the compression is optimized, the input is still compressed properly although not as quickly.

Let us prove that one of the compressions that are run by the automaton compresses the input in time at most $(\frac{k-1}{k} + \epsilon)\,\mathrm{RT}$. A compression along the vector corresponding exactly to the proportion of the input would take a time $\frac{k-1}{k}\,\mathrm{RT}$. A compression along one of the vectors in S that is closest to the optimal vector (at distance at most $\epsilon\eta$) puts all states from the input within a distance at most $\epsilon\eta\,\mathrm{RT}$ from their destination in time $\frac{k-1}{k}\,\mathrm{RT} + O(1)$. By choosing the closest vector properly amongst the two choices, the remaining distance can be travelled in time at most $\epsilon\,\mathrm{RT} + O(1)$ as illustrated by Figure 6 (information travels at speed at least η in one of the dimensions).

For any possible input, at least one of the compressions completes in time at most $(\frac{k-1}{k} + \epsilon)\,\mathrm{RT} + O(1)$.

5 Transition

After the input has been compressed, the automaton $\mathcal{A}'$ should immediately start simulating the behavior of $\mathcal{A}$, k steps at a time. However the cells of $\mathcal{A}'$ receive the compressed input at different times. If we wanted all the cells to start the next phase at the same time, we would

require some synchronization scheme such as a firing-squad synchronization algorithm but this would take a linear time. Instead, we show that synchronization is not required to start the accelerated simulation as each cell of the automaton proceeds with the next phase as soon as the relevant information is available.

This technique is very general and can be used in numerous situations where a cellular automaton performs a computation by executing a series of separate tasks one after the other without having to spend time synchronizing all cells. In its general form, it can be stated in the following way:

▶ **Proposition 17** (Passive Synchronization). *Given a CA $\mathcal{A}$ of any dimension working on a complete neighborhood $\mathcal{N}$, there exists a CA $\mathcal{A}'$ working on the same neighborhood $\mathcal{N}$ that can simulate the behavior of $\mathcal{A}$ on any input even if the configuration is given asynchronously in such a way that each cell of $\mathcal{A}'$ computes states of the simulated automaton at least as fast as if the computation had started synchronously when the last cell receives its input.*

Formally, if we denote by $\mathcal{Q}$ the states of $\mathcal{A}$, $\mathcal{A}'$ has states $\{\perp\} \cup (\mathcal{Q} \times \mathcal{Q}')$ where $\perp$ is a permanent state (cannot be changed by the transition rule of the automaton) and $\mathcal{Q}'$ is a set of extra working states containing a default state ν. The cells of $\mathcal{A}'$ are initially in state $\perp$ and considered inactive*. Before each transition of the automaton, any number of inactive cells of $\mathcal{A}'$ might be* activated *by some external action over which $\mathcal{A}'$ has no control. Activating a cell c changes its state to $(\mathfrak{C}(c), \nu)$ where $\mathfrak{C}$ is the input of the simulated automaton $\mathcal{A}$.*

If there exists a time t_0 at which all cells have been activated then for any cell c the projection on $\mathcal{Q}$ of the state of c in $\mathcal{A}'$ at time $(t_0 + t)$ is the state of c in the evolution of $\mathcal{A}$ from the configuration $\mathfrak{C}$ at time t' for some $t' \geq t$.

Proof. The idea is to make all cells of $\mathcal{A}'$ compute one step of $\mathcal{A}$ whenever they have enough information to do so, while remembering their past states that other cells might need at a later time.

When a cell c is activated, it receives the initial state $\mathfrak{C}(c)$ and we say that its *simulated time* is 0. From that point on, it looks at its neighbors and waits for all of them to be activated. When this happens, it sees all initial states in its neighborhood and can compute the next state in the evolution of $\mathcal{A}$, increasing its simulated time to 1. As time passes it keeps watching its neighbors until all of them are also at a simulated time at least equal to its own, which means that it has all the information necessary to compute the next step and increase its simulated time further.

Let us prove that this process can be carried out with finitely many states. First, notice that since $\mathcal{N}$ is complete, there exists $\tau \in \mathbb{N}$ such that $-\mathcal{N} \subseteq \mathcal{N}^\tau$. In order to compute its state for a simulated time $(\tau + t)$ a cell c needs to have had access to the state at the simulated time t of all cells in $(c + \mathcal{N}^\tau)$ which includes the set $(c - \mathcal{N})$ of cells that have c in their neighborhood. This means that a cell cannot be more than τ steps ahead in its simulation than the cells that have it in their neighborhood, which implies that the difference of simulated times between two neighbor cells is at most τ. If each cell stores the value of its simulated time modulo $(2\tau + 1)$, it is possible for a cell to know the relative difference in simulated time with all cells in its neighborhood. Furthermore, it is sufficient that a cell remembers its last τ simulated states to be sure that when a cell c at simulated time t looks at its neighbors that are more advanced in the simulation it can see their simulated state at time t.

Finally, we prove by induction that at time $(t_0 + t)$ all cells have a simulated time at least t. This is obviously true at time t_0 since all cells have been activated. By induction, at time

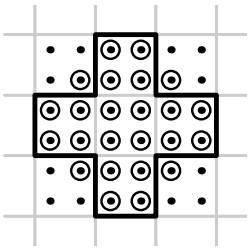


Figure 7 Compression of factor $k = 2$ with the von Neumann neighborhood. The thick line represents $\rho(c+\mathcal{N})$, the states that are visible to the central cell in one step. The circles represent $\rho(c) + \mathcal{N}^k$, the states that should be known to compute two steps of the original automaton on the states that the central cell holds.

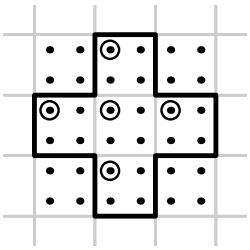


Figure 8 The thick line represents $\rho(c + \mathcal{N})$. The circles represent the cells in $(c' + k\mathcal{N})$ for some c' in $\rho(c)$ (the one in the top left).

$(t_0 + t)$ for any cell at simulated time t all its neighbors are at least at simulated time t so it can compute a new step of the simulation, which proves that at time $(t_0 + t + 1)$ all cells have a simulated time of at least $(t + 1)$.

Note that this process is such that the cells who are behind in their simulation can compute new states without delay, whereas the ones ahead wait for their neighbors to catch up. ◀

In the following section we describe how $\mathcal{A}'$ can simulate k steps of $\mathcal{A}$ at a time starting from a compressed input. We assume that all cells complete the compression and start the simulation synchronously when the last cell receives its compressed information. Using Proposition 17, we can connect the two constructions (cells are *activated* for the simulation when they receive their compressed input) and ensure that the origin is always at least as advanced in its computation as if the simulation had started synchronously.

6 Simulation on a Compressed Input

Let us denote by ρ the function that maps a cell of $\mathcal{A}'$ to the set of cells of $\mathcal{A}$ whose states it receives after the compression of a factor k :

$$\forall c \in \mathbb{Z}^2, \quad \rho(c) = \{kc + (x, y) \mid x, y \in [\![0, k-1]\!]\}$$

and extend the notation to sets of cells by $\rho(S) = \bigcup_{c \in S} \rho(c)$.

The states of $\mathcal{A}$ that are held in the neighborhood of a cell c in $\mathcal{A}'$ are the ones corresponding to the cells of $\mathcal{A}$ in $\rho(c + \mathcal{N})$. To be able to compute k steps of the original automaton, the cell c in $\mathcal{A}'$ needs to be able to see in its neighborhood the states corresponding to the cells $(\rho(c) + \mathcal{N}^k)$ of $\mathcal{A}$. Although this is the case for simple rectangular neighborhoods, it is not true for some neighborhoods (see Figure 7).

What is true however is that $\rho(c) + k\mathcal{N} \subseteq \rho(c + \mathcal{N})$ since for any $v \in \mathcal{N}$ if the state of a cell c' in $\mathcal{A}$ is held by a cell c in $\mathcal{A}'$ after compression of the input, the cell $(c + v)$ in $\mathcal{A}'$ holds the state of $(c' + kv)$ in $\mathcal{A}$ (see Figure 8).

▶ **Lemma 18.** $\forall k \in \mathbb{N}, \exists \alpha \in \mathbb{N}, \quad \mathcal{N}^\alpha + k\mathcal{N} = \mathcal{N}^\alpha + \mathcal{N}^k$

Proof. The inclusion $\mathcal{N}^\alpha + k\mathcal{N} \subseteq \mathcal{N}^\alpha + \mathcal{N}^k$ is obvious for any α. As for the converse, choose α such that $\alpha + k > |\mathcal{N}|(k-1)$. Any $x \in \mathcal{N}^\alpha + \mathcal{N}^k$ can be written as the sum of $(\alpha + k)$

elements of $\mathcal{N}$ and therefore at least one of these elements appears at least k times, which proves that $x \in \mathcal{N}^\alpha + k\mathcal{N}$. ◀

By Lemma 18, we can choose α such that $\mathcal{N}^\alpha + k\mathcal{N} = \mathcal{N}^\alpha + \mathcal{N}^k$. We now modify the behavior of $\mathcal{A}'$ so that during the first α steps of the computation, before starting the compression, all cells gather the initial states contained in their $\mathcal{N}^\alpha$ neighborhood. From here onwards, each cell performs all of the computation as described earlier on all the states it holds : a cell c holds at time $(\alpha + t)$ the states that would have been on all the cells in $(c + \mathcal{N}^\alpha)$ at time t on the automaton $\mathcal{A}'$ as described until now. At time $(\alpha + t)$ the cells in $(c + \mathcal{N})$ as a whole hold all the states that would have been on the cells in $(c + \mathcal{N}^{\alpha+1})$ at time t, which is exactly what is needed to compute the states of all cells in $(c + \mathcal{N}^\alpha)$ at time $(t + 1)$. This extra step adds a constant time α to the computation of the automaton[1].

After the initial gathering and compression of the input, the cell c in $\mathcal{A}'$ holds the initial states in $\mathcal{A}$ for the cells in $(\rho(c) + \mathcal{N}^\alpha)$. Let us show by induction that this is enough to simulate the behavior of $\mathcal{A}'$ with a linear speed-up of factor k. Assume that at time $(t_0 + t)$, any cell c of $\mathcal{A}'$ holds the states at time t for the cells of $\mathcal{A}$ in $(\rho(c) + \mathcal{N}^\alpha)$.

This means that the cells in the neighborhood $(c + \mathcal{N})$ of c in $\mathcal{A}'$ at time $(t_0 + t)$ hold the states at time t in $\mathcal{A}$ of the cells in $\rho(c + \mathcal{N}) + \mathcal{N}^\alpha$. By Lemma 18, we have

$$\rho(c + \mathcal{N}) + \mathcal{N}^\alpha \quad \supseteq \quad \rho(c) + k\mathcal{N} + \mathcal{N}^\alpha \quad \supseteq \quad \rho(c) + \mathcal{N}^{\alpha+k}$$

which shows that cell c in $\mathcal{A}'$ at time $(t_0 + t)$ sees enough information to compute the states in $\mathcal{A}$ for the cells in $(\rho(c) + \mathcal{N}^\alpha)$ at time $(t + k)$.

7 Total time

We have completed the description of the behavior of the automaton $\mathcal{A}'$. Let us now evaluate the total time taken to recognize the language L recognized by $\mathcal{A}$ in time $\mathrm{RT}_\mathcal{N} + f$.

The compression of the input takes a time $(\frac{k-1}{k} + \epsilon)\,\mathrm{RT}_\mathcal{N} + O(1)$. The simulation of $\mathcal{A}$ from a fully compressed input takes a time $\frac{1}{k}(\mathrm{RT}_\mathcal{N} + f) + O(1)$ for some $k \geq \frac{1}{\epsilon}$, and Proposition 17 shows that no time is lost by completing the compression asynchronously (the time of the compression is the time at which the last cell is correctly compressed).

The total time for the simulation of $\mathcal{A}$ is therefore

$$(\frac{k-1}{k} + \epsilon)\,\mathrm{RT}_\mathcal{N} + \frac{1}{k}(\mathrm{RT}_\mathcal{N} + f) + O(1) \leq (1 + \epsilon)\,\mathrm{RT}_\mathcal{N} + \epsilon f + O(1)$$

By Claim 15, the $O(1)$ term can be eliminated, which concludes the proof of Theorem 13.

8 Conclusion

The linear acceleration presented in this article is slightly weaker than the previously known results on a limited class of neighborhoods (which contains the von Neumann and Moore neighborhoods). On these neighborhoods, as well as all one-dimensional complete neighborhoods, any language that can be recognized in time $(\mathrm{RT} + f)$ can be recognized in time $(\mathrm{RT} + \epsilon f)$ for any $\epsilon > 0$.

[1] The constant time α is actually not lost since the origin holds the states that would be on the cells in $\mathcal{N}^\alpha$, which enables it to compute its own state α time steps ahead. However, for the purpose of proving Theorem 13, adding a constant time to the computation is irrelevant.

Although the difference is only significant if $f = o(\mathrm{RT})$, it would be interesting to know whether this stronger statement can be proved for general two-dimensional complete neighborhoods. This would either require an optimal-time compression of the input or a completely different construction skipping the compression altogether.

As we currently understand it, optimal-time compression seems unlikely on general neighborhoods. The problem is that states from the initial configuration should move towards the origin in the optimal direction permitted by the neighborhood. Before receiving any information from the axes, a cell has no way of knowing the precise direction to the origin. If the neighborhood's convex hull has more than one vertex in the positive quarter of the plane, moving along any of the directions permitted by the neighborhood might be sub-optimal, as oppposed to the case of the Moore neighborhood in which going diagonally at first is never sub-optimal and by the time it is necessary to change direction to go either horizontally or vertically information is received from the axes.

If only one cell needs to send its information towards the origin, the problem can be solved by spreading the information in all directions and spreading symmetric signals from the origin. It is however not possible to implement this for all cells at the same time with finitely many states.

Acknowledgments. The authors would like to thank Jacques Mazoyer for his helpful conversations and inspiring ideas at the start of the work that led to this article.

References

1 J. Albert and K. Čulik II. A simple universal cellular automaton and its one-way and totalistic version. *Complex Systems*, 1:1–16, 1987.

2 W.T. Beyer. *Recognition of topological invariants by iterative arrays*. Massachusetts Institute of Technology, Project MAC, 1969.

3 Stephen N. Cole. Real-time computation by n-dimensional iterative arrays of finite-state machines. *IEEE Transactions on Computers*, C-18(4):349–365, 1969.

4 Martin Delacourt and Victor Poupet. Real time language recognition on 2d cellular automata: Dealing with non-convex neighborhoods. In Ludek Kucera and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, volume 4708 of *Lecture Notes in Computer Science*, pages 298–309. Springer, 2007. doi:10.1007/978-3-540-74456-6_28.

5 P. C. Fischer. Generation of primes by one-dimensional real-time iterative array. *Journal of the Assoc. Comput. Mach.*, 12:388–394, 1965.

6 F.C. Hennie. *Iterative Arrays of Logical Circuits*. MIT Press Classics. MIT Press, 1961.

7 Jacques Mazoyer and Nicolas Reimen. A linear speed-up theorem for cellular automata. *Theor. Comput. Sci.*, 101(1):59–98, 1992. doi:10.1016/0304-3975(92)90150-E.

8 Zsuzsanna Róka. Simulations between cellular automata on Cayley graphs. *Theoretical Computer Science*, 225(1-2):81–111, 1999.

9 Alvy R. Smith III. Simple computation-universal cellular spaces. *J. ACM*, 18(3):339–353, 1971. doi:10.1145/321650.321652.

10 Alvy R. Smith III. Two-dimensional formal languages and pattern recognition by cellular automata. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory (Swat 1971)*, SWAT'71, pages 144–152, Washington, DC, USA, 1971. IEEE Computer Society. doi:10.1109/SWAT.1971.29.

11 Alvy R. Smith III. Real-time language recognition by one-dimensional cellular automata. *Journal of the Assoc. Comput. Mach.*, 6:233–253, 1972.

12 Véronique Terrier. Two-dimensional cellular automata recognizer. *Theor. Comput. Sci.*, 218(2):325–346, 1999. doi:10.1016/S0304-3975(98)00329-6.

13 Véronique Terrier. Two-dimensional cellular automata and their neighborhoods. *Theor. Comput. Sci.*, 312(2-3):203–222, 2004. doi:10.1016/j.tcs.2003.08.011.

14 John von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, IL, USA, 1966.

New Interpretation and Generalization of the Kameda-Weiner Method*

Hellis Tamm

Institute of Cybernetics, Tallinn University of Technology, Tallinn, Estonia
hellis@cs.ioc.ee

Abstract

We present a reinterpretation of the Kameda-Weiner method of finding a minimal nondeterministic finite automaton (NFA) of a language, in terms of atoms of the language. We introduce a method to generate NFAs from a set of languages, and show that the Kameda-Weiner method is a special case of it. Our method provides a unified view of the construction of several known NFAs, including the canonical residual finite state automaton and the atomaton of the language.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Nondeterministic finite automata, NFA minimization, Kameda-Weiner method, atoms of regular languages

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.116

1 Introduction

Nondeterministic finite automata (NFAs), introduced by Rabin and Scott [11] in 1959, have played a major role in the theory and applications of finite automata. In particular, the problem of finding NFAs with the minimal number of states has received much attention. Different approaches have been used over the years when trying to solve this problem, of which the work done by Kameda and Weiner [10] in 1970 seems to be among the most classical ones. Kameda and Weiner studied the problem of NFA minimization using a matrix based on the states of the minimal deterministic finite automata (DFAs) for a given language and its reverse. They suggested a method of finding a minimal NFA using grids of this matrix.

We present a reinterpretation of the Kameda-Weiner method, using the recently introduced atoms of regular languages [3, 5], and continuing the work started by Brzozowski and Tamm in [4], where the Kameda-Weiner method was formulated in terms of quotients and atoms of a language. We show that the matrix used by Kameda and Weiner can be viewed as the *quotient-atom* matrix of the language, and that any maximal grid of this matrix can be seen as the set of atoms that the grid involves. We also show that, instead of applying the rather complicated *intersection rule* of the Kameda-Weiner method, to construct an NFA corresponding to a cover of the matrix, consisting of maximal grids, one can use sets of atoms associated with grids, and form an NFA based on these sets. We note that essentially the same approach to the Kameda-Weiner method, which uses projections of grids (corresponding to sets of atoms), has been presented by Champarnaud and Coulon [6].

Furthermore, we generalize the idea of constructing an NFA using sets of atoms. Namely, we introduce a method to generate NFAs from a set of languages, and show that the

* This work was supported by the ERDF funded CoE project EXCS and ICT National Programme project "Coinduction", and the Estonian Ministry of Education and Research institutional research grant IUT33-13.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 116; pp. 116:1–116:12

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

Kameda-Weiner method of constructing a minimal NFA is a special case of this method. The introduced method provides a unified view of the construction of several known NFAs including, for example, the canonical residual finite state automaton and the átomaton of the language.

The structure of the rest of the paper is as follows. In Section 2, we provide definitions for automata, quotients, and atoms of a regular language, and recall some results related to atoms. Section 3 describes the Kameda-Weiner method of finding a minimal NFA of a language and shows how the Kameda-Weiner construction of an NFA can be expressed in terms of atoms. In Section 4, we introduce a method to generate NFAs from a set of languages and present a few examples of known NFAs that can be constructed using this method. In Section 5, we show that the NFA minimization method presented by Kameda and Weiner is a special case of generating an NFA by our method. Section 6 concludes the paper.

2 Automata, Quotients, and Atoms of Regular Languages

A *nondeterministic finite automaton (NFA)* is a quintuple $\mathcal{N} = (Q, \Sigma, \delta, I, F)$, where Q is a finite, non-empty set of *states*, Σ is a finite non-empty *alphabet*, $\delta : Q \times \Sigma \to 2^Q$ is the *transition function*, $I \subseteq Q$ is the set of *initial states*, and $F \subseteq Q$ is the set of *final states*. We extend the transition function to functions $\delta' : Q \times \Sigma^* \to 2^Q$ and $\delta'' : 2^Q \times \Sigma^* \to 2^Q$, using δ for all these functions. An NFA $\mathcal{N}' = (Q', \Sigma', \delta', I', F')$ is a *subautomaton* of $\mathcal{N}$ if $Q' \subseteq Q$, $\Sigma' \subseteq \Sigma$, $I' \subseteq I$, $F' \subseteq F$, and $q \in \delta'(p, a)$ implies $q \in \delta(p, a)$ for every $p, q \in Q'$ and $a \in \Sigma'$.

The *language accepted* by an NFA $\mathcal{N}$ is $L(\mathcal{N}) = \{w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\}$. The *right language* of a state q of $\mathcal{N}$ is $L_{q,F}(\mathcal{N}) = \{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\}$. A state is *empty* if its right language is empty. The *left language* of a state q of $\mathcal{N}$ is $L_{I,q} = \{w \in \Sigma^* \mid q \in \delta(I, w)\}$. A state is *unreachable* if its left language is empty. An NFA is *trim* if it has no empty or unreachable states. If $\mathcal{N}_1 = (Q_1, \Sigma, \delta_1, I_1, F_1)$ and $\mathcal{N}_2 = (Q_2, \Sigma, \delta_2, I_2, F_2)$ are NFAs, then a map φ from Q_1 into Q_2 is a *morphism* from $\mathcal{N}_1$ into $\mathcal{N}_2$ if and only if $\varphi(I_1) \subseteq I_2$, $\varphi(F_1) \subseteq F_2$, and $q \in \delta_1(p, a)$ implies $\varphi(q) \in \delta_2(\varphi(p), a)$.

A *deterministic finite automaton (DFA)* is a quintuple $\mathcal{D} = (Q, \Sigma, \delta, q_0, F)$, where Q, Σ, and F are as in an NFA, $\delta : Q \times \Sigma \to Q$ is the transition function, and q_0 is the initial state.

The following three operations on automata are commonly used: the *determinization* operation D applied to an NFA $\mathcal{N}$, yielding a DFA $\mathcal{N}^D$, obtained by the well-known subset construction, the *reversal* operation R which, when applied to an NFA $\mathcal{N}$, yields an NFA $\mathcal{N}^R$, where the sets of the initial and the final states of $\mathcal{N}$ are interchanged and all transitions are reversed, and the *trimming* operation T which, when applied to an NFA $\mathcal{N}$, results in an NFA $\mathcal{N}^T$ where all unreachable and empty states are removed.

The *left quotient*, or simply *quotient*, of a language L by a word $w \in \Sigma^*$ is the language $w^{-1}L = \{x \in \Sigma^* \mid wx \in L\}$. There is one *initial* quotient, $\varepsilon^{-1}L = L$. A quotient is *final* if it contains ε. It is well known that there is a one-to-one correspondence between the set of states $Q = \{q_0, \ldots, q_{n-1}\}$ of the minimal DFA $\mathcal{D} = (Q, \Sigma, \delta, q_0, F)$ accepting L and the set of quotients $\{K_0, \ldots, K_{n-1}\}$ of L, such that $L_{q_i,F}(\mathcal{D}) = K_i$ for $i = 0, \ldots, n-1$.

An *atom* of a regular language L with quotients $K_0, \ldots, K_{n-1}$ is any non-empty language of the form $\widetilde{K_0} \cap \cdots \cap \widetilde{K_{n-1}}$, where $\widetilde{K_i}$ is either K_i or $\overline{K_i}$, and $\overline{K_i}$ is the complement of K_i with respect to Σ^*. Thus atoms of L are regular languages uniquely determined by L and they define a partition of Σ^*. They are pairwise disjoint and every quotient of L (including L itself) is a union of atoms. Also, every quotient of an atom of L is a union of atoms. It has been noticed that atoms are exactly the classes of the *left congruence* of L [9] defined as

follows: for $x, y \in \Sigma^*$, x is equivalent to y if for every $u \in \Sigma^*$, $ux \in L$ if and only if $uy \in L$. This idea was used in [2], where this equivalence is called the *atom congruence*.

A regular language L with n quotients has at most 2^n atoms. An atom is *initial* if it has L (rather than $\overline{L}$) as a term; it is *final* if it contains ε. There is exactly one final atom, the atom $\widehat{K_0} \cap \cdots \cap \widehat{K_{n-1}}$, where $\widehat{K_i} = K_i$ if $\varepsilon \in K_i$, and $\widehat{K_i} = \overline{K_i}$ otherwise. Let $A = \{A_0, \ldots, A_{m-1}\}$ be the set of atoms of L, let I_A be the set of initial atoms, and let A_{m-1} be the final atom. If $\overline{K_0} \cap \cdots \cap \overline{K_{n-1}}$ is an atom, then it is called the *negative* atom, all the other atoms are *positive*.

We use a one-to-one correspondence $A_i \leftrightarrow \mathbf{A}_i$ between atoms A_i of a language L and the states $\mathbf{A}_i$ of the NFA $\mathcal{A}$ defined as follows [5]:

▶ **Definition 1.** The *átomaton* of L is the NFA $\mathcal{A} = (\mathbf{A}, \Sigma, \alpha, \mathbf{I}_A, \{\mathbf{A}_{m-1}\})$, where $\mathbf{A} = \{\mathbf{A}_i \mid A_i \in A\}$, $\mathbf{I}_A = \{\mathbf{A}_i \mid A_i \in I_A\}$, and $\mathbf{A}_j \in \alpha(\mathbf{A}_i, a)$ if and only if $A_j \subseteq a^{-1}A_i$, for all $\mathbf{A}_i, \mathbf{A}_j \in \mathbf{A}$ and $a \in \Sigma$.

The right language of any state $\mathbf{A}_i$ of the átomaton is the atom A_i [5].

The next theorem is a slightly modified version of the result by Brzozowski [1]:

▶ **Theorem 2.** *If an NFA $\mathcal{N}$ has no empty states and $\mathcal{N}^R$ is deterministic, then $\mathcal{N}^D$ is minimal.*

Since it was shown in [5] that $\mathcal{A}^R$ is a minimal DFA for the reverse language of L, we know by Theorem 2 that $\mathcal{A}^R$ is isomorphic to $\mathcal{D}^{RD}$, where $\mathcal{D}$ is the minimal DFA of L. Thus, $\mathcal{A}$ is isomorphic to $\mathcal{D}^{RDR}$.

A new class of NFA's was defined in [5] as follows: an NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ is *atomic* if for every $q \in Q$, the right language $L_{q,F}(\mathcal{N})$ of q is a union of atoms of $L(\mathcal{N})$. Also, it was shown that for any NFA $\mathcal{N}$, $\mathcal{N}^D$ is a minimal DFA if and only if $\mathcal{N}^R$ is atomic.

3 NFA Minimization by Kameda and Weiner

Kameda and Weiner [10] have developed a theory of NFA minimization. They used minimal DFAs for a language L and its reverse L^R to form a matrix, and based on the grids in this matrix, a minimal NFA was found. We note that the biclique edge cover technique presented by Gruber and Holzer [8] as a lower bound method for the size of a minimal NFA, uses another representation of the same matrix.

We present main principles of the Kameda-Weiner method, using mostly our terminology and notation. Kameda and Weiner [10] consider a trim minimal DFA $\mathcal{D} = (Q, \Sigma, \delta, q_0, F)$ with Q of cardinality p, and its reversed determinized and trim version $\mathcal{D}^{RDT}$; the set of states of $\mathcal{D}^{RDT}$ is a subset S of cardinality r of $2^Q \setminus \emptyset$. They then form an $p \times r$ matrix T where the rows correspond to non-empty states $q_i \in Q$ of $\mathcal{D}$, which is the trim minimal DFA of a language L, and columns, to states $S_j \in S$ of $\mathcal{D}^{RDT}$, which is the trim minimal DFA of the language L^R by Theorem 2. The entry $t_{i,j}$ of the matrix T is 1 if $q_i \in S_j$, and 0 otherwise.

We use $\mathcal{D}^{RDRT}$, the trim átomaton, instead of $\mathcal{D}^{RDT}$, since the state sets of these two automata are identical. Interpret the rows of the matrix as non-empty quotients of L and columns, as positive atoms of L. Then $t_{i,j} = 1$ if and only if quotient K_i contains atom A_j as a subset, and it is clear that every regular language defines a unique such matrix, which we will call the *quotient-atom matrix*.

The ordered pair (K_i, A_j) is a *point* of T if $t_{i,j} = 1$. A *grid* g of T is the direct product $g = P \times R$ of a set P of quotients with a set R of atoms, such that every atom in R is a

subset of every quotient in P. If $g = P \times R$ and $g' = P' \times R'$ are two grids of T, then $g \subseteq g'$ if and only if $P \subseteq P'$ and $R \subseteq R'$. Thus $\subseteq$ is a partial order on the set of all grids of T, and a grid is *maximal* if it is not contained in any other grid. We say that a grid $g = P \times R$ is *horizontally maximal* if for any grid $g' = P \times R'$, $R' \subseteq R$. Similarly, a grid $g = P \times R$ is *vertically maximal* if for any grid $g' = P' \times R$, $P' \subseteq P$. Clearly, any maximal grid is both horizontally and vertically maximal.

A *cover* G of T is a set $G = \{g_0, \ldots, g_{k-1}\}$ of grids, such that every point (K_i, A_j) belongs to some grid g_i in G. A *minimal cover* has the minimal number of grids.

Let f_G be the function that assigns to every non-empty quotient K_i the subset of a cover G, consisting of grids $g = P \times R$ such that $K_i \in P$. The NFA constructed by the Kameda-Weiner method is $\mathcal{N}_G = (G, \Sigma, \eta_G, I_G, F_G)$, where G is a cover consisting of maximal grids, $I_G = f_G(K_0)$ is the set of grids corresponding to the initial quotient K_0, and F_G is defined by $g \in F_G$ if and only if $g \in f_G(K_i)$ implies that K_i is a final quotient. For every grid $g = P \times R$ and $a \in \Sigma$, we can compute $\eta_G(g, a)$ by the formula $\eta_G(g, a) = \bigcap_{K_i \in P} f_G(a^{-1}K_i)$. It is said that the NFA $\mathcal{N}_G$ is obtained from $\mathcal{D}$ by the *intersection rule*, using the (grid) cover G.

It may be the case that $\mathcal{N}_G$ does not accept the language L. A cover G is called *legal* if $L(\mathcal{N}_G) = L$. To find a minimal NFA of a language L, the method in [10] tests the covers of the quotient-atom matrix of L in the order of increasing size to see if they are legal. The first legal NFA is a minimal one.

Next, we will interpret the Kameda-Weiner method in terms of atoms. For this, we first show the relationship between maximal grids and certain sets of atoms. Let us start with the following definition:

▶ **Definition 3.** Let R be a set of atoms and let $U(R) = \bigcup_{A_j \in R} A_j$ be the union of these atoms. We define the *maximized* version of $U(R)$ to be the language $max(U(R)) = \bigcap_{U(R) \subseteq K_i} K_i$. We say that the set R is *maximal* if $max(U(R)) = U(R)$.

The following proposition is an easy observation:

▶ **Proposition 4.** *Let R be a set of atoms. Then*
1. $U(R) \subseteq max(U(R))$,
2. $max(max(U(R))) = max(U(R))$.

▶ **Proposition 5.** *Let R_i and R_j be sets of atoms. The following properties hold:*
1. *If $U(R_i) \subseteq U(R_j)$, then $max(U(R_i)) \subseteq max(U(R_j))$.*
2. *For every $a \in \Sigma$, $max(a^{-1}U(R_i)) \subseteq a^{-1}max(U(R_i))$.*

Proof. To prove the first claim, let $U(R_i) \subseteq U(R_j)$. Then it is easy to see that the inclusion $\{K_h \mid U(R_i) \subseteq K_h\} \supseteq \{K_k \mid U(R_j) \subseteq K_k\}$ holds, implying that also the inclusion $\bigcap_{U(R_i) \subseteq K_h} K_h \subseteq \bigcap_{U(R_j) \subseteq K_k} K_k$ holds. Thus, $max(U(R_i)) \subseteq max(U(R_j))$.

To prove the second property, consider the set of quotients K_h such that $U(R_i) \subseteq K_h$. Since $U(R_i) \subseteq K_h$ implies that $a^{-1}U(R_i) \subseteq a^{-1}K_h$ holds, it is clear that the inclusion $\{a^{-1}K_h \mid U(R_i) \subseteq K_h\} \subseteq \{K_k \mid a^{-1}U(R_i) \subseteq K_k\}$ holds. This implies that also the inclusion $\bigcap_{a^{-1}U(R_i) \subseteq K_k} K_k \subseteq \bigcap_{U(R_i) \subseteq K_h} a^{-1}K_h$ holds. Since $\bigcap_{U(R_i) \subseteq K_h} a^{-1}K_h = a^{-1} \bigcap_{U(R_i) \subseteq K_h} K_h$, we get that $max(a^{-1}U(R_i)) \subseteq a^{-1}max(U(R_i))$. ◀

Next, we will see that any maximal grid can be considered as a maximal set of atoms it involves.

▶ **Proposition 6.** *For any grid $g = P \times R$, $U(R) \subseteq \bigcap_{K_i \in P} K_i$ holds.*

Proof. For any grid $g = P \times R$, it holds that for every $K_i \in P$ and $A_j \in R$, $A_j \subseteq K_i$, implying that $U(R) \subseteq \bigcap_{K_i \in P} K_i$. ◀

▶ **Proposition 7.** *A grid $g = P \times R$ is horizontally maximal if and only if the equality $\bigcap_{K_i \in P} K_i = U(R)$ holds.*

Proof. For any grid $g = P \times R$, the inclusion $U(R) \subseteq \bigcap_{K_i \in P} K_i$ holds by Proposition 6. Let g be a horizontally maximal grid. Then there is no grid $g' = P \times R'$, such that $R \subset R'$. That is, there is no $A_l \in \{A_0, \ldots, A_{m-1}\} \setminus R$, such that $U(R) \cup A_l \subseteq \bigcap_{K_i \in P} K_i$ would hold. Since every quotient is a disjoint union of atoms, every intersection of quotients is also a union of atoms. Therefore, the equality $\bigcap_{K_i \in P} K_i = U(R)$ holds.

Conversely, if $\bigcap_{K_i \in P} K_i = U(R)$, then for every grid $g' = P \times R'$, the inclusion $R' \subseteq R$ holds. Thus, the grid $g = P \times R$ is horizontally maximal. ◀

▶ **Corollary 8.** *A grid $g = P \times R$ is horizontally maximal if and only if the set R is maximal.*

Proof. Let $g = P \times R$ be a horizontally maximal grid. By Proposition 7, this means that the equality $\bigcap_{K_i \in P} K_i = U(R)$ holds. Since $max(U(R)) = \bigcap_{U(R) \subseteq K_i} K_i$, it is clear that $max(U(R)) = U(R)$. Thus, R is maximal. ◀

▶ **Corollary 9.** *A grid $g = P \times R$ is maximal if and only if P is a maximal set of quotients such that the equality $\bigcap_{K_i \in P} K_i = U(R)$ holds.*

According to Corollaries 8 and 9, any maximal grid involves a maximal set of atoms and the set of quotients, such that the intersection of these quotients is the union of the atoms involved.

As the main result of this section, we will prove the following theorem which shows how the construction of the NFA $\mathcal{N}_G$ can be expressed in terms of atoms:

▶ **Theorem 10.** *Let $G = \{g_0, \ldots, g_{k-1}\}$ be a cover consisting of maximal grids $g_i = P_i \times R_i$, $i = 0, \ldots, k-1$, and let $\mathcal{N}_G = (G, \Sigma, \eta_G, I_G, F_G)$ be the corresponding NFA, obtained by the intersection method. It holds that $g_i \in I_G$ if and only if $U(R_i) \subseteq L$, and $g_i \in F_G$ if and only if $\varepsilon \in U(R_i)$. For any $g_i, g_j \in G$ and $a \in \Sigma$, $g_j \in \eta_G(g_i, a)$ if and only if the inclusion $U(R_j) \subseteq a^{-1}U(R_i)$ holds.*

Proof. The set I_G of initial states of $\mathcal{N}_G$ consists of those grids that intersect the initial quotient $K_0 = L$. That is, for every grid $g_i \in G$, $g_i \in I_G$ if and only if $U(R_i) \subseteq L$ holds.

The set F_G of final states of $\mathcal{N}_G$ is the set of grids that intersect only the final quotients. Equivalently, for any $g_i \in G$, it holds that $g_i \in F_G$ if and only if $U(R_i)$ includes the final atom. The latter is equivalent to having $\varepsilon \in U(R_i)$.

Next, let $g_j \in \eta_G(g_i, a)$ for some $g_i, g_j \in G$ and $a \in \Sigma$. By the intersection rule, it holds that $\eta_G(g_i, a) = \bigcap_{K_h \in P_i} f(a^{-1}K_h)$. That is, $g_j \in \eta_G(g_i, a)$ if and only if $g_j \in f(a^{-1}K_h)$ for every $K_h \in P_i$. This implies that $g_j \in \eta_G(g_i, a)$ if and only if $a^{-1}K_h \in P_j$ holds for every $K_h \in P_i$.

It is clear that if $a^{-1}K_h \in P_j$ holds for every $K_h \in P_i$, then the inclusion $\bigcap_{K_k \in P_j} K_k \subseteq \bigcap_{K_h \in P_i} a^{-1}K_h$ holds. And conversely, if $\bigcap_{K_k \in P_j} K_k \subseteq \bigcap_{K_h \in P_i} a^{-1}K_h$, then since by Corollary 9, P_j is a maximal set of quotients such that the equality $\bigcap_{K_k \in P_j} K_k = U(R_j)$ holds, it must be that $a^{-1}K_h \in P_j$ for $K_h \in P_i$. Thus, $a^{-1}K_h \in P_j$ for $K_h \in P_i$ if and only if the inclusion $\bigcap_{K_k \in P_j} K_k \subseteq \bigcap_{K_h \in P_i} a^{-1}K_h$ holds. Because of the equality $\bigcap_{K_h \in P_i} a^{-1}K_h = a^{-1}\bigcap_{K_h \in P_i} K_h$, we get the equivalent condition $\bigcap_{K_k \in P_j} K_k \subseteq a^{-1}\bigcap_{K_h \in P_i} K_h$. Using Corollary 9, we get the inclusion $U(R_j) \subseteq a^{-1}U(R_i)$. ◀

Theorem 10 provides another way of constructing the NFA $\mathcal{N}_G$ from a given set of maximal grids covering the quotient-atom matrix: instead of applying the intersection rule to get transitions of $\mathcal{N}_G$, one can use the sets of atoms corresponding to the grids, and apply quotients of unions of the atoms involved. This can be done using the transition function of the átomaton.

We mention that basically the same approach to the Kameda-Weiner method has been presented by Champarnaud and Coulon [6]. They used projections of grids, consisting of subsets of the state set of the DFA $\mathcal{D}^{RDT}$, to construct an NFA similarly as in Theorem 10.

In the next section, we will generalize this idea of using sets of atoms (or unions of atoms) of a language L, to construct NFAs for L.

4 Generating Automata by a Set of Languages

In this section, we introduce a method to generate NFAs from a set of languages.

Let L be a regular language, and let $K = \{K_0, \ldots, K_{n-1}\}$ be the set of quotients of L. A set $\{L_0, \ldots, L_{k-1}\}$ of languages is a *cover* of the quotients of L, or simply, a cover for L, if every quotient K_j of L is a union of some L_i's. We note that since L is the quotient of itself by the empty word ε, L is a union of some L_i's.

We define the NFA based on a cover $\{L_0, \ldots, L_{k-1}\}$ as follows:

▶ **Definition 11.** The NFA generated by a cover $\{L_0, \ldots, L_{k-1}\}$ for L is defined by $\mathcal{G} = (Q, \Sigma, \delta, I, F)$, where $Q = \{q_0, \ldots, q_{k-1}\}$, $I = \{q_i \mid L_i \subseteq L\}$, $F = \{q_i \mid \varepsilon \in L_i\}$, and $q_j \in \delta(q_i, a)$ if and only if $L_j \subseteq a^{-1}L_i$ for all $q_i, q_j \in Q$ and $a \in \Sigma$.

▶ **Lemma 12.** *For all states q_i, q_j of NFA $\mathcal{G}$ and for any word $w \in \Sigma^+$, $q_j \in \delta(q_i, w)$ if and only if $L_j \subseteq w^{-1}L_i$.*

Proof. We prove the statement by induction on the length of w. If $w = a$ for some $a \in \Sigma$, then the lemma holds by Definition 11.

Now, let $w = ua$, where $u \in \Sigma^+$ and $a \in \Sigma$, and assume that the lemma holds for u, that is, for all states q_i, q_j of $\mathcal{G}$, $q_j \in \delta(q_i, u)$ if and only if $L_j \subseteq u^{-1}L_i$. Consider a state q_i, and let $q_k \in \delta(q_i, ua)$. Then there is a state q_j, such that $q_j \in \delta(q_i, u)$ and $q_k \in \delta(q_j, a)$. Equivalently, by the induction assumption and Definition 11, respectively, the inclusions $L_j \subseteq u^{-1}L_i$ and $L_k \subseteq a^{-1}L_j$ hold. Hence $L_k \subseteq a^{-1}L_j \subseteq a^{-1}(u^{-1}L_i) = (ua)^{-1}L_i$. Thus, $q_k \in \delta(q_i, ua)$ if and only if $L_k \subseteq (ua)^{-1}L_i$. ◀

▶ **Proposition 13.** *The following properties hold for NFA $\mathcal{G}$:*

1. *$L_{q_i,F}(\mathcal{G}) \subseteq L_i$ for every $q_i \in Q$.*
2. *$L(\mathcal{G}) \subseteq L$.*

Proof. 1. Consider a state q_i of $\mathcal{G}$. Let $w \in L_{q_i,F}(\mathcal{G})$. If $w = \varepsilon$, then $q_i \in F$, and $\varepsilon \in L_i$ by Definition 11. If $w \in \Sigma^+$, then there is some q_j such that $q_j \in F$ and $q_j \in \delta(q_i, w)$. By Lemma 12, $L_j \subseteq w^{-1}L_i$ and $\varepsilon \in L_j$ implying that $w \in L_i$.

2. Since $L(\mathcal{G})$ is the union of right languages of the initial states of $\mathcal{G}$, the claim follows from Definition 11 and Part 1. ◀

▶ **Lemma 14.** *If $a^{-1}L_i$ is a union of L_j's for every L_i and $a \in \Sigma$, then $w^{-1}L_i$ is a union of L_j's for every L_i and $w \in \Sigma^+$.*

Proof. Let $a^{-1}L_i$ be a union of L_j's for every L_i and $a \in \Sigma$, that is, $a^{-1}L_i = \bigcup_{j \in J_{i,a}} L_j$ for some $J_{i,a} \subseteq \{0, \ldots, k-1\}$. We prove the statement by induction on the length of w. If $w = a$ for some $a \in \Sigma$, then the lemma trivially holds.

Now, let $w = ua$, where $u \in \Sigma^+$ and $a \in \Sigma$, and assume that the lemma holds for u, that is, $u^{-1}L_i = \bigcup_{j \in J_{i,u}} L_j$ for some $J_{i,u} \subseteq \{0, \ldots, k-1\}$. Then $(ua)^{-1}L_i = a^{-1}(u^{-1}L_i) = a^{-1}(\bigcup_{j \in J_{i,u}} L_j) = \bigcup_{j \in J_{i,u}} a^{-1}L_j = \bigcup_{j \in J_{i,u}} \bigcup_{h \in J_{j,a}} L_h$. Thus, $(ua)^{-1}L_i$ is a union of L_h's. ◀

▶ **Proposition 15.** *Let $\mathcal{G} = (Q, \Sigma, \delta, I, F)$ be the NFA generated by a cover $\{L_0, \ldots, L_{k-1}\}$ for L. The equality $L_{q_i,F}(\mathcal{G}) = L_i$ holds for every $q_i \in Q$ if and only if $a^{-1}L_i$ is a union of L_j's for every L_i and $a \in \Sigma$.*

Proof. First, let the equality $L_{q_i,F}(\mathcal{G}) = L_i$ hold for every $q_i \in Q$. Let us consider any L_i and $a \in \Sigma$. Then it holds that $a^{-1}L_i = a^{-1}L_{q_i,F}(\mathcal{G}) = \bigcup_{q_j \in \delta(q_i,a)} L_{q_j,F}(\mathcal{G}) = \bigcup_{L_j \subseteq a^{-1}L_i} L_j$.

Conversely, assume that $a^{-1}L_i$ is a union of L_j's for every L_i and $a \in \Sigma$. Let us consider any state q_i of $\mathcal{G}$. By Proposition 13, the inclusion $L_{q_i,F}(\mathcal{G}) \subseteq L_i$ holds, so we only have to show that $L_i \subseteq L_{q_i,F}(\mathcal{G})$. Let w be any word in L_i. If $w = \varepsilon$, then $q_i \in F$, and so $w \in L_{q_i,F}(\mathcal{G})$. If $w \in \Sigma^+$, then by Lemma 14, $w^{-1}L_i$ is a union of L_j's. Since $w \in L_i$, there must be some L_j such that $L_j \subseteq w^{-1}L_i$ and $\varepsilon \in L_j$. By Lemma 12, there is some $q_j \in F$ such that $q_j \in \delta(q_i, w)$. Therefore $w \in L_{q_i,F}(\mathcal{G})$, and we conclude that $L_{q_i,F}(\mathcal{G}) = L_i$. ◀

▶ **Proposition 16.** *Let $\mathcal{G} = (Q, \Sigma, \delta, I, F)$ be the NFA generated by a cover $\{L_0, \ldots, L_{k-1}\}$ for L. If $a^{-1}L_i$ is a union of L_j's for every L_i and $a \in \Sigma$, then $\mathcal{G}$ accepts L.*

Proof. If $a^{-1}L_i$ is a union of L_j's for every L_i and $a \in \Sigma$, then by Proposition 15, $L_{q_i,F}(\mathcal{G}) = L_i$ holds for every $q_i \in Q$. Since $L(\mathcal{G}) = \bigcup_{q_i \in I} L_{q_i,F}(\mathcal{G}) = \bigcup_{L_i \subseteq L} L_i = L$, the equality $L(\mathcal{G}) = L$ holds. ◀

We present four examples of covers for the language L and the corresponding NFAs generated by these covers, where the condition of Proposition 16 holds, ensuring that the generated NFA accepts L:

▶ **Example 17.** Consider the set $K = \{K_0, \ldots, K_{n-1}\}$ of quotients of L as a cover for L. Let $\mathcal{G}_K$ be the NFA generated by the set K. Since for every quotient K_i and $a \in \Sigma$ there exists some quotient K_j such that $a^{-1}K_i = K_j$, we know by Proposition 16 that $\mathcal{G}_K$ accepts L. It is well known that the states of the minimal DFA correspond to the quotients of L. However, the NFA $\mathcal{G}_K$ is isomorphic to the *saturated* version [7] of the minimal DFA of L.

▶ **Example 18.** Consider the set $K' \subseteq K$ of *prime* quotients of L, that is, those non-empty quotients of L which are not unions of other quotients, as a cover for L. Let $\mathcal{G}_{K'}$ be the NFA generated by the set K'. Since every quotient of L is a union of some prime quotients of L, it is clear that for every prime quotient K'_i and $a \in \Sigma$, $a^{-1}K'_i$ is a union of prime quotients. Thus, $\mathcal{G}_{K'}$ accepts L by Proposition 16. The NFA $\mathcal{G}_{K'}$ is known as the *canonical residual finite state automaton (canonical RFSA)* [7] of L.

▶ **Example 19.** Consider the set $A = \{A_0, \ldots, A_{m-1}\}$ of atoms of L. The set of atoms is a cover for L, because every quotient of L is a union of atoms [5]. The NFA $\mathcal{G}_A$, generated by the set A, is the átomaton of L (cf. Definition 1). It is known that for every atom A_i and $a \in \Sigma$, $a^{-1}A_i$ is a union of atoms [5]. Thus, the condition of Proposition 16 holds, and $\mathcal{G}_A$ accepts L.

▶ **Example 20.** Let $A_0, \ldots, A_{m-1}$ be the atoms of L. Consider the set $M = \{M_0, \ldots, M_{m-1}\}$ of the maximized versions of atoms, that is, $M_i = max(A_i)$ for $i = 0, \ldots, m-1$. Clearly, if $A_i \subseteq K_j$ for some atom A_i and quotient K_j, then the inclusion $M_i \subseteq K_j$ holds by Definiton 3. Thus, the set M is a cover for L. The NFA $\mathcal{G}_M$, generated by the set M, is the *maximized átomaton* [13] of L. Since for any $M_i \in M$ and $a \in \Sigma$, $a^{-1}M_i = \bigcup_{A_j \subseteq M_i} a^{-1}A_j$, and because $a^{-1}A_j$ is a union of atoms [5], we get that $a^{-1}M_i$ is a union of atoms. By [13, Proposition 2, Part 4], the inclusion $A_j \subseteq a^{-1}M_i$ holds if and only if $M_j \subseteq a^{-1}M_i$ holds. We conclude that $a^{-1}M_i$ is a union of M_j's, and by Proposition 16, $\mathcal{G}_M$ accepts L.

However, we note that the condition of Proposition 16 is not necessary for the generated NFA to accept L.

▶ **Proposition 21.** *If $\mathcal{N}$ is a trim NFA accepting L, with the set $\{L_0, \ldots, L_{k-1}\}$ of the right languages of its states, then this set is a cover for L.*

Proof. By determinizing $\mathcal{N}$, the quotients of L are formed as unions of some L_i's. ◀

▶ **Proposition 22.** *Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be a trim NFA of L, with the set $\{L_0, \ldots, L_{k-1}\}$ of the right languages of its states, and let $\mathcal{G} = (Q', \Sigma, \delta', I', F')$ be the NFA generated by the set $\{L_0, \ldots, L_{k-1}\}$. Let $\varphi : Q \to Q'$ be the mapping assigning to every state q of $\mathcal{N}$, the state q'_i of $\mathcal{G}$, such that $L_i = L_{q,F}(\mathcal{N})$. Then φ is a morphism from $\mathcal{N}$ into $\mathcal{G}$.*

Proof. Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be a trim NFA accepting L, with the set $\{L_0, \ldots, L_{k-1}\}$ of the right languages of its states. By Proposition 21, the set $\{L_0, \ldots, L_{k-1}\}$ is a cover for L. Let $\mathcal{G} = (Q', \Sigma, \delta', I', F')$ be the NFA generated by the set $\{L_0, \ldots, L_{k-1}\}$, with $Q' = \{q'_0, \ldots, q'_{k-1}\}$. Let $\varphi : Q \to Q'$ be the mapping assigning to every state q of $\mathcal{N}$, the state q'_i of $\mathcal{G}$, such that $L_i = L_{q,F}(\mathcal{N})$. We note that there may be some states p and q of $\mathcal{N}$, such that $p \neq q$ and $L_{p,F}(\mathcal{N}) = L_{q,F}(\mathcal{N})$, so φ is a many-to-one correspondence. We show that φ is a morphism from $\mathcal{N}$ into $\mathcal{G}$.

First, if $q \in I$ is an initial state of $\mathcal{N}$, then there is some L_i, such that $L_i = L_{q,F}(\mathcal{N})$ and $L_i \subseteq L$, which implies that the corresponding state q'_i of $\mathcal{G}$ is also initial, that is, $\varphi(q) \in I'$.

Similarly, if $q \in F$, then there is some L_i, such that $L_i = L_{q,F}(\mathcal{N})$ and $\varepsilon \in L_i$, implying that $q'_i \in F'$, that is, $\varphi(q) \in F'$.

If $q \in \delta(p, a)$ holds for some states $p, q \in Q$ and $a \in \Sigma$, then there are some L_i and L_j, such that $L_i = L_{p,F}(\mathcal{N})$, $L_j = L_{q,F}(\mathcal{N})$, and $L_j \subseteq a^{-1}L_i$. It is implied that $q'_j \in \delta'(q'_i, a)$, that is, $\varphi(q) \in \delta'(\varphi(p), a)$. We conclude that φ is a morphism from $\mathcal{N}$ into $\mathcal{G}$. ◀

▶ **Theorem 23.** *If there is a trim NFA accepting L, with the set $\{L_0, \ldots, L_{k-1}\}$ of the right languages of its states, then the NFA generated by the cover $\{L_0, \ldots, L_{k-1}\}$ for L is such an NFA.*

Proof. Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be a trim NFA accepting L, with the set $\{L_0, \ldots, L_{k-1}\}$ of the right languages of its states. By Proposition 21, the set $\{L_0, \ldots, L_{k-1}\}$ is a cover for L. Let $\mathcal{G} = (Q', \Sigma, \delta', I', F')$ be the NFA generated by the set $\{L_0, \ldots, L_{k-1}\}$, with $Q' = \{q'_0, \ldots, q'_{k-1}\}$. By Proposition 22, there is a morphism $\varphi : Q \to Q'$ from $\mathcal{N}$ into $\mathcal{G}$, such that $\varphi(q) = q'_i$ for some $q \in Q$ and $q'_i \in Q'$ if and only if $L_{q,F}(\mathcal{N}) = L_i$.

The morphism φ implies that for every state $q \in Q$, with its right language $L_{q,F}(\mathcal{N}) = L_i$ for some L_i, the inclusion $L_{q,F}(\mathcal{N}) \subseteq L_{q'_i,F'}(\mathcal{G})$, that is, $L_i \subseteq L_{q'_i,F'}(\mathcal{G})$ holds. Since by Proposition 13, Part 1, the inclusion $L_{q'_i,F'}(\mathcal{G}) \subseteq L_i$ holds, the equality $L_{q'_i,F'}(\mathcal{G}) = L_i$ must hold. Also, the morphism φ implies that the inclusion $L(\mathcal{N}) \subseteq L(\mathcal{G})$ holds. Since by Proposition 13, Part 2, $L(\mathcal{G}) \subseteq L$, and we assumed that $L(\mathcal{N}) = L$, we conclude that $\mathcal{G}$ accepts L. ◀

Theorem 23 shows that our method to generate NFAs from a set of languages is indeed general. That is, if one is interested in finding an NFA for a given language, such that the states of that NFA correspond to certain languages, this method can be used to generate such an NFA if it exists. If the generated NFA is not such an NFA, then it does not exist.

To conclude this section, we point out three cases which can occur if a cover $\{L_0, \ldots, L_{k-1}\}$ for L is used to generate an NFA $\mathcal{G}$:

First, the NFA $\mathcal{G}$ accepts L, and the right language of every state q_i of $\mathcal{G}$ is L_i. This case is described by Propositions 15 and 16.

In the second case, the NFA $\mathcal{G}$ accepts L, but the right language of some state q_i of $\mathcal{G}$ is not L_i. The third case is when $\mathcal{G}$ does not accept L. Characterization of the last two cases is an interesting problem for further study.

5 Generating Automata by Atomic Languages

Let L be a regular language, with its quotients $K_0, \ldots, K_{n-1}$ and atoms $A_0, \ldots, A_{m-1}$.

▶ **Definition 24.** A language L_i is *atomic* with regard to L if L_i is a union of atoms of L.

Let $\mathcal{N} = (Q, \Sigma, \delta, I, F)$ be a trim NFA accepting L, with $Q = \{q_0, \ldots, q_{k-1}\}$. For every state q_i of $\mathcal{N}$, we define an atomic language $B_i = \bigcup_{L_{q_i,F}(\mathcal{N}) \cap A_h \neq \emptyset} A_h$ as the union of all atoms of L which intersect with the right language of q_i. In other words, B_i is the smallest atomic language that contains the right language of state q_i. Clearly, if $L_{q_i,F}(\mathcal{N}) \subseteq K_j$ holds for some quotient K_j of L, then, because every quotient is a union of atoms, $B_i \subseteq K_j$ holds as well. Since by Proposition 21, the set of right languages of the states of $\mathcal{N}$ forms a cover for L, the set of B_i's has the same property. We note that there may be some states q_i and q_j of $\mathcal{N}$, such that $q_i \neq q_j$, but $B_i = B_j$. Let the set of distinct B_i's be B.

Let $\mathcal{G}_B = (Q_B, \Sigma, \delta_B, I_B, F_B)$ be the NFA generated by the cover B for the language L. We note that $|Q_B| \leqslant |Q|$. Let $\varphi_{atom} : Q \to Q_B$ be the mapping assigning to state q_i of $\mathcal{N}$, the state q_{B_i} of $\mathcal{G}_B$, such that $B_i = \bigcup_{L_{q_i,F}(\mathcal{N}) \cap A_h \neq \emptyset} A_h$.

▶ **Proposition 25.** *The mapping φ_{atom} is a morphism from $\mathcal{N}$ into $\mathcal{G}_B$.*

Proof. First, if $q_i \in I$ is initial, then $L_{q_i,F}(\mathcal{N}) \subseteq L$, and since L is a union of (initial) atoms, the inclusion $B_i \subseteq L$ holds, implying that q_{B_i} is also initial, that is, $\varphi_{atom}(q_i) \in I_B$.

Similarly, if $q_i \in F$, then $\varepsilon \in L_{q_i,F}(\mathcal{N})$, implying that $\varepsilon \in B_i$, and thus $q_{B_i} \in F_B$, that is, $\varphi_{atom}(q_i) \in F_B$.

It remains to be shown that for all states $q_i, q_j \in Q$ and $a \in \Sigma$, if $q_j \in \delta(q_i, a)$ holds, then $\varphi_{atom}(q_j) \in \delta_B(\varphi_{atom}(q_i), a)$ holds as well. Let $q_j \in \delta(q_i, a)$ for some $q_i, q_j \in Q$ and $a \in \Sigma$. Then the inclusion $L_{q_j,F}(\mathcal{N}) \subseteq a^{-1}L_{q_i,F}(\mathcal{N})$ holds. Because of $L_{q_i,F}(\mathcal{N}) \subseteq B_i$, the inclusion $L_{q_j,F}(\mathcal{N}) \subseteq a^{-1}B_i$ holds. Since it is known that any quotient of a union of atoms is some union of atoms, $a^{-1}B_i$ is a union of atoms. Consequently, $L_{q_j,F}(\mathcal{N}) \subseteq B_j \subseteq a^{-1}B_i$ holds, implying that $q_{B_j} \in \delta_B(q_{B_i}, a)$, that is, $\varphi_{atom}(q_j) \in \delta_B(\varphi_{atom}(q_i), a)$.

We conclude that φ_{atom} is a morphism from $\mathcal{N}$ into $\mathcal{G}_B$. ◀

▶ **Corollary 26.** *For every state q_i of $\mathcal{N}$, the inclusion $L_{q_i,F}(\mathcal{N}) \subseteq L_{q_{B_i},F_B}(\mathcal{G}_B)$ holds. Also, $L(\mathcal{G}_B) = L$.*

Proof. The morphism $\varphi_{atom} : Q \to Q_B$ implies that for every $q_i \in Q$, the inclusion $L_{q_i,F}(\mathcal{N}) \subseteq L_{q_{B_i},F_B}(\mathcal{G}_B)$ holds, and also that $L(\mathcal{N}) \subseteq L(\mathcal{G}_B)$ holds.

Since $L(\mathcal{N}) = L$, and we know by Proposition 13 that $L(\mathcal{G}_B) \subseteq L$, we conclude that $L(\mathcal{G}_B) = L$. ◀

▶ **Corollary 27.** *If there is a one-to-one correspondence between the sets Q and B, then the NFA $\mathcal{N}$ is isomorphic to a subautomaton of $\mathcal{G}_B$.*

Next, for every atomic language B_i we consider its maximized version, the language $C_i = max(B_i) = \bigcap_{B_i \subseteq K_j} K_j$. Clearly, C_i is also atomic, and $B_i \subseteq C_i$. If the inclusion $B_i \subseteq K_j$ holds for some quotient K_j, then by the definition of C_i, $C_i \subseteq K_j$ holds as well. Since the set of B_i's forms a cover for L, so does the set of corresponding C_i's. We note that there may be some B_i and B_j, such that $B_i \neq B_j$, but $C_i = C_j$. Let the set of distinct C_i's be C.

Let $\mathcal{G}_C = (Q_C, \Sigma, \delta_C, I_C, F_C)$ be the NFA generated by the cover C for the language L. We note that $|Q_C| \leqslant |Q_B|$. Let $\varphi_{max} : Q_B \to Q_C$ be the mapping assigning to state q_{B_i} of $\mathcal{G}_B$, the state q_{C_i} of $\mathcal{G}_C$.

▶ **Proposition 28.** *The mapping φ_{max} is a morphism from $\mathcal{G}_B$ into $\mathcal{G}_C$.*

Proof. First, if $q_{B_i} \in I_B$, then $B_i \subseteq L$. Since C_i is a subset of the same quotients as B_i, $C_i \subseteq L$, implying that $q_{C_i} \in I_C$. If $q_{B_i} \in F_B$, then $\varepsilon \in B_i$, and since $B_i \subseteq C_i$, it holds that $\varepsilon \in C_i$, so we get $q_{C_i} \in F_C$.

We also have to show that if $q_{B_j} \in \delta_B(q_{B_i}, a)$ holds for some states q_{B_i} and q_{B_j} of $\mathcal{G}_B$ and $a \in \Sigma$, then $q_{C_j} \in \delta_C(q_{C_i}, a)$ for the corresponding states q_{C_i} and q_{C_j} of $\mathcal{G}_C$. Indeed, if $q_{B_j} \in \delta_B(q_{B_i}, a)$, then $B_j \subseteq a^{-1}B_i$. By Proposition 5, Part 1, we know that $max(B_j) \subseteq max(a^{-1}B_i)$, and by Part 2, the inclusion $max(a^{-1}B_i) \subseteq a^{-1}max(B_i)$ holds. Since $C_i = max(B_i)$ and $C_j = max(B_j)$, we get that $C_j \subseteq a^{-1}C_i$ holds. Thus, $q_{C_j} \in \delta_C(q_{C_i}, a)$.

We conclude that φ_{max} is a morphism from $\mathcal{G}_B$ into $\mathcal{G}_C$. ◀

▶ **Corollary 29.** *For every state q_{B_i} of $\mathcal{G}_B$, the inclusion $L_{q_{B_i},F_B}(\mathcal{G}_B) \subseteq L_{q_{C_i},F_C}(\mathcal{G}_C)$ holds. Also, $L(\mathcal{G}_C) = L$.*

Proof. The morphism $\varphi_{max} : Q_B \to Q_C$ implies that for every $q_{B_i} \in Q_B$, the inclusion $L_{q_{B_i},F_B}(\mathcal{G}_B) \subseteq L_{q_{C_i},F_C}(\mathcal{G}_C)$ holds, and also that $L(\mathcal{G}_B) \subseteq L(\mathcal{G}_C)$ holds.

Since $L(\mathcal{G}_B) = L$ by Corollary 26, and $L(\mathcal{G}_C) \subseteq L$ by Proposition 13, we conclude that $L(\mathcal{G}_C) = L$. ◀

▶ **Corollary 30.** *If there is a one-to-one correspondence between the sets B and C, then the NFA $\mathcal{G}_B$ is isomorphic to a subautomaton of $\mathcal{G}_C$.*

Based on the results above, we can state the following theorem:

▶ **Theorem 31.** *There is a morphism $\varphi_{max} \circ \varphi_{atom}$ from a trim NFA $\mathcal{N}$ into the NFA $\mathcal{G}_C$, generated by the set C of languages $C_i = max(\bigcup_{L_{q_i,F}(\mathcal{N}) \cap A_h \neq \emptyset} A_h)$, where q_i is a state of $\mathcal{N}$, with $L(\mathcal{G}_C) = L(\mathcal{N})$. Moreover, if there is a one-to-one correspondence between the states of $\mathcal{N}$ and $\mathcal{G}_C$, then $\mathcal{N}$ is isomorphic to a subautomaton of $\mathcal{G}_C$.*

The following theorem shows that the NFA minimization method presented by Kameda and Weiner is a special case of generating an NFA:

▶ **Theorem 32.** *Let $G = \{g_0, \ldots, g_{k-1}\}$ be a set of maximal grids, with $g_i = P_i \times R_i$, forming a cover of the quotient-atom matrix of L. The NFA $\mathcal{N}_G$, obtained by the Kameda-Weiner method using G, is isomorphic to the NFA $\mathcal{G}_C$, generated by the set $C = \{C_0, \ldots, C_{k-1}\}$ of languages $C_i = U(R_i)$, $i = 0, \ldots, k-1$.*

Proof. Let $G = \{g_0, \ldots, g_{k-1}\}$ be a set of maximal grids $g_i = P_i \times R_i$, forming a cover of the quotient-atom matrix of L. Let $\mathcal{N}_G = (G, \Sigma, \eta_G, I_G, F_G)$ be the NFA obtained by the intersection method using G, and let $\mathcal{G}_C = (Q_C, \Sigma, \delta_C, I_C, F_C)$ be the NFA generated by the set $C = \{C_0, \ldots, C_{k-1}\}$, where $C_i = U(R_i)$ for $i = 0, \ldots, k-1$. We show that $\mathcal{N}_G$ is isomorphic to $\mathcal{G}_C$ by applying Theorem 10.

First, by Theorem 10, for every grid $g_i \in G$, it holds that $g_i \in I_G$ if and only if the inclusion $U(R_i) \subseteq L$ holds, that is, $C_i \subseteq L$. Since this is equivalent to the condition $q_{C_i} \in I_C$, there is a one-to-one correspondence between the sets I_G and I_C.

Also, it holds that $g_i \in F_G$ if and only if $\varepsilon \in U(R_i)$, that is, $\varepsilon \in C_i$. This is equivalent to the condition $q_{C_i} \in F_C$. Thus, there is a one-to-one correspondence between the sets F_G and F_C.

It remains to show that $g_j \in \eta_G(g_i, a)$ if and only if $q_{C_j} \in \delta_C(q_{C_i}, a)$ for all $g_i, g_j \in G$ and $a \in \Sigma$. Indeed, by Theorem 10, $g_j \in \eta_G(g_i, a)$ holds if and only if the inclusion $U(R_j) \subseteq a^{-1}U(R_i)$ holds, that is, $C_j \subseteq a^{-1}C_i$. On the other hand, by Definition 11, $q_{C_j} \in \delta_C(q_{C_i}, a)$ if and only if $C_j \subseteq a^{-1}C_i$, where $q_{C_i}, q_{C_j} \in Q_C$ and $a \in \Sigma$. Therefore, $g_j \in \eta_G(g_i, a)$ holds if and only if $q_{C_j} \in \delta_C(q_{C_i}, a)$ holds for all $g_i, g_j \in G$ and $a \in \Sigma$. ◀

▶ **Corollary 33.** *There exists an atomic NFA with the right languages $C_0, \ldots, C_{k-1}$, such that the set of atoms contained in every C_i is maximal, if and only if the Kameda-Weiner method finds it.*

Proof. Follows from Theorem 23 and Theorem 32. ◀

▶ **Corollary 34.** *There is a morphism from any trim NFA $\mathcal{N}$ into the NFA $\mathcal{N}_G$ obtained by the Kameda-Weiner method using the set G of maximal grids, corresponding to the maximal sets of atoms associated to the right languages of $\mathcal{N}$.*

Proof. Follows from Theorem 31 and Theorem 32. ◀

As a special case, if $\mathcal{N}$ is a minimal NFA, then by Theorem 31, $\mathcal{N}$ is isomorphic to a subautomaton of the NFA $\mathcal{G}_C$ generated by the set C of the maximized atomic languages of the right languages of $\mathcal{N}$, or equivalently, as by Theorem 32, of the NFA $\mathcal{N}_G$ obtained by the Kameda-Weiner method, using the corresponding maximal grids.

This indeed ensures that if one considers covers of the quotient-atom matrix, starting from the smallest cover, and produces NFAs according to the Kameda-Weiner method, or equivalently, generates NFAs, using unions of atoms corresponding to the grids in the cover, the first obtained NFA which accepts the given language, is a minimal NFA.

As we mentioned earlier, Champarnaud and Coulon [6] have presented an approach to the Kameda-Weiner method which, similarly to our method, finds NFAs corresponding to grid covers, using projections of grids (corresponding to sets of atoms). We note that they also used grid extensions and automaton morphisms, similarly to our theory. However, we point out that our theory explicitly shows that atoms of regular languages have an important role in the Kameda-Weiner method.

We also mention that Sengoku's method [12] of constructing NFAs is related to atoms; it yields atomic NFAs. However, we note that by a result proved in [5], not every language has an atomic minimal NFA.

6 Conclusions

We presented a reinterpretation of the Kameda-Weiner method for NFA minimization, and generalized it by introducing a method to generate NFAs by certain sets of languages. We hope that our contributions provide a useful insight into the difficult problem of NFA minimization, to obtain a better understanding of this problem.

We also think that the introduced method of generating NFAs is of interest on its own as exemplified in Section 4. This method provides a unified view of the construction of several known NFAs, including the canonical RFSA and the átomaton of the language.

Acknowledgements. The author is grateful to Janusz Brzozowski for discussions and collaboration during the early stages of this work. The author also thanks Wolfgang Jeltsch for discussions.

References

1 J. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In *Proceedings of the Symposium on Mathematical Theory of Automata*, volume 12 of *MRI Symposia Series*, pages 529–561. Polytechnic Press, Polytechnic Institute of Brooklyn, N.Y., 1963.

2 J. Brzozowski and S. Davies. Quotient complexities of atoms in regular ideal languages. *Acta Cybernetica*, 22(2):293–311, 2015.

3 J. Brzozowski and H. Tamm. Theory of átomata. In G. Mauri and A. Leporati, editors, *Proceedings of the 15th International Conference on Developments in Language Theory (DLT)*, volume 6795 of *Lecture Notes in Computer Science*, pages 105–116. Springer, 2011.

4 J. Brzozowski and H. Tamm. Minimal nondeterministic finite automata and atoms of regular languages, 2013. URL: `http://arxiv.org/abs/1301.5585`.

5 J. Brzozowski and H. Tamm. Theory of átomata. *Theor. Comput. Sci.*, 539:13–27, 2014.

6 J.-M. Champarnaud and F. Coulon. Enumerating nondeterministic automata for a given language without constructing the canonical automaton. *Int. J. Found. Comput. Sci.*, 16:1253–1266, 2005.

7 F. Denis, A. Lemay, and A. Terlutte. Residual finite state automata. *Fund. Inform.*, 51:339–368, 2002.

8 H. Gruber and M. Holzer. Finding lower bounds for nondeterministic state complexity is hard. In *Proc. of DLT 2006*, volume 4036 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2006.

9 S. Iván. Complexity of atoms, combinatorially. *Information Processing Letters*, 116:356–360, 2016.

10 T. Kameda and P. Weiner. On the state minimization of nondeterministic finite automata. *IEEE Trans. Comput.*, C-19(7):617–627, 1970.

11 M. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. and Dev.*, 3:114–129, 1959.

12 H. Sengoku. Minimization of nondeterministic finite automata. Master's thesis, Kyoto University, Department of Information Science, Kyoto University, Kyoto, Japan, 1992.

13 H. Tamm. Generalization of the double-reversal method of finding a canonical residual finite state automaton. In *Proceedings of DCFS 2015*, volume 9118 of *LNCS*, pages 268–279. Springer, 2015.

Nesting Depth of Operators in Graph Database Queries: Expressiveness vs. Evaluation Complexity*†

M. Praveen[1] and B. Srivathsan[2]

1 **Chennai Mathematical Institute, Chennai, India**
2 **Chennai Mathematical Institute, Chennai, India**

Abstract

Designing query languages for graph structured data is an active field of research, where expressiveness and efficient algorithms for query evaluation are conflicting goals. To better handle dynamically changing data, recent work has been done on designing query languages that can compare values stored in the graph database, without hard coding the values in the query. The main idea is to allow variables in the query and bind the variables to values when evaluating the query. For query languages that bind variables only once, query evaluation is usually NP-complete. There are query languages that allow binding inside the scope of Kleene star operators, which can themselves be in the scope of bindings and so on. Uncontrolled nesting of binding and iteration within one another results in query evaluation being PSPACE-complete.

We define a way to syntactically control the nesting depth of iterated bindings, and study how this affects expressiveness and efficiency of query evaluation. The result is an infinite, syntactically defined hierarchy of expressions. We prove that the corresponding language hierarchy is strict. Given an expression in the hierarchy, we prove that it is undecidable to check if there is a language equivalent expression at lower levels. We prove that evaluating a query based on an expression at level i can be done in level i of the polynomial time hierarchy. Satisfiability of quantified Boolean formulas can be reduced to query evaluation; we study the relationship between alternations in Boolean quantifiers and the depth of nesting of iterated bindings.

1998 ACM Subject Classification F.1.3 Complexity measures and classes: Complexity hierarchies, F.4.3 Formal languages: Classes defined by grammars or automata, H.2.3 Languages: Query languages

Keywords and phrases graphs with data, regular data path queries, expressiveness, query evaluation, complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.117

1 Introduction

Graph structures representing data have found many applications like semantic web [11], social networks [19] and biological networks [13]. Theoretical models of such data typically have a graph with nodes representing entities and edges representing relations among them. One reason for the popularity of these models is their flexibility in handling semi-structured data. While traditional relational databases impose rigid structures on the relations between

* A full version of this paper is available at `http://arxiv.org/abs/1603.00658`.
† Both the authors are partially funded by a grant from Infosys Foundation.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 117; pp. 117:1–117:14

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

data elements, graph databases are better equipped to handle data in which relations are not precisely known and/or developing dynamically.

A fundamental query language for such models is Regular Path Queries (RPQs), which is now part of the W3C recommendation [18]. An RPQ consists of a regular expression. Suppose a communication network is modeled by a graph, where nodes represent servers and edges labeled ℓ represent links between them. Evaluating the RPQ ℓ^* on this graph results in the set of pairs of nodes between which there exists a route. Suppose each link has a priority and we need pairs of connected nodes where all intermediate links have the same priority. We can hard code the set of priorities in the query. If the set of priorities is not static, a querying mechanism which avoids hard coding is better. Every edge can be labeled by a supplementary data value (priority of the link, in this example) and we want query languages that can compare data values without hard coding them in the syntax. Nodes can also carry data values. In generic frameworks, there is no a priori bound on the number of possible data values and they are considered to be elements of an infinite domain. Graph databases with data values are often called data graphs in theory and property graphs in practice.

One way to design querying languages for data graphs is to extend RPQs using frameworks that handle words on infinite alphabets [16, 15, 12, 23]. Expressiveness and efficient algorithms for query evaluation are conflicting goals for designing such languages. We study a feature common to many of these languages, and quantify how it affects the trade-off between expressiveness and complexity of query evaluation. Variable finite automata [10] and parameterized regular expressions [2] are conservative extensions of classical automata and regular expressions. They have variables, which can be bound to letters of the alphabet at the beginning of query evaluation. The query evaluation problem is NP-complete for these languages. Regular expressions with binding (REWBs) [15] is an extended formalism where binding of variables to values can happen inside a Kleene star, which can itself be in the scope of another binding operator and so on. Allowing binding and iteration to occur inside each other's scope freely results in the query evaluation problem being PSPACE-complete. Here we study how the expressiveness and complexity of query evaluation vary when we syntactically control the depth of nesting of iterated bindings.

Contributions

1. We syntactically classify REWBs according to the depth of nesting of iterated bindings.
2. The resulting hierarchy of data languages is strict, and so is the expressiveness of queries.
3. It is undecidable to check if a given REWB has a language equivalent one at lower levels.
4. An REWB query in level i can be evaluated in Σ_i in the polynomial time hierarchy.
5. For lower bounds, we consider quantified Boolean formulas with some restrictions on quantifications and reduce their satisfiability to query evaluation, with some restrictions on the queries.

For proving strictness of the language hierarchy, we build upon ideas from the classic star height hierarchy [9]. Universality of REWBs is known to be undecidable [17, 12]. We combine techniques from this proof with tools developed for the language hierarchy to prove the third result above. The Σ_i upper bound for query evaluation involves complexity theoretic arguments based on the same tools. In the reductions from satisfiability of quantified Boolean formulas to the query evaluation problem, the relation between the number of alternations (in the Boolean quantifiers) and the depth of nesting (of iterated bindings in REWBs) is not straight forward. We examine this relation closely in the framework of parameterized complexity theory, which is suitable for studying the effect of varying the structure of input instances on the complexity.

Related work: The quest for efficient evaluation algorithms and expressive languages to query graph databases, including those with data values, is an active area of research; [1] is a recent comprehensive survey. Numerous formalisms based on logics and automata exist for handling languages over infinite alphabets [20]. In [16], the suitability of these formalisms as query languages has been studied, zeroing in on register automata mainly for reasons of efficient evaluation. The same paper introduced regular expressions with memory and proved that they are equivalent to register automata. REWBs [15] have slightly less expressive power but have better scoping structure for the binding operator. Properties of these expressions have been further studied in [12]. In [14], XPath has been adapted to query data graphs. Pebble automata have been adapted to work with infinite alphabets in [17]. A strict language hierarchy based on the number of pebbles allowed in pebble automata has been developed in [22]. Many questions about comparative expressiveness of register and pebble automata are open [17]. Fixed-point logics can be used to define languages over infinite alphabets [4]. These logics can use the class successor relation, which relates two positions with the same data value if no intermediate position carries the same value. Expressiveness of these logics increase [6, 5], when the number of alternations between standard successor relation and class successor relation increase.

2 Preliminaries

2.1 Data Languages and Querying Data Graphs

We follow the notation of [15]. Let Σ be a finite alphabet and $\mathcal{D}$ a countably infinite set. The elements of $\mathcal{D}$ are called *data values*. A *data word* is a finite string over the alphabet $\Sigma \times \mathcal{D}$. We will write a data word as $\binom{a_1}{d_1}\binom{a_2}{d_2}\ldots\binom{a_n}{d_n}$, where each $a_i \in \Sigma$ and $d_i \in \mathcal{D}$. A set of data words is called a *data language*.

An extension of standard regular expressions, called *regular expressions with binding (REWB)*, has been defined in [15]. Here, data values are compared using variables. For a set $\{x_1, x_2, \ldots, x_k\}$ of variables, the set of conditions $\mathcal{C}_k$ is the set of Boolean combinations of $x_i^{=}$ and $x_i^{\neq}$ for $i \in \{1, \ldots, k\}$. A data value $d \in \mathcal{D}$ and a partial valuation $\nu : \{x_1, \ldots, x_k\} \to \mathcal{D}$ satisfies the condition $x_i^{=}$ (written as $d, \nu \models x_i^{=}$) if $\nu(x_i) = d$. The satisfaction for other Boolean operators are standard.

▶ **Definition 2.1** (Regular expressions with binding (REWB) [15])**.** Let Σ be a finite alphabet and $\{x_1, \ldots, x_k\}$ a set of variables. Regular expressions with binding over $\Sigma[x_1, \ldots, x_k]$ are defined inductively as: $r := \varepsilon \mid a \mid a[c] \mid r + r \mid r \cdot r \mid r^* \mid a \downarrow_x (r)$ where $a \in \Sigma$ is a letter in the alphabet, $c \in \mathcal{C}_k$ is a condition on the variables and $x \in \{x_1, \ldots, x_k\}$ is a variable.

We call $\downarrow_x$ the *binding operator*. In the expression $a \downarrow_x (r)$, the expression r is said to be the *scope* of the binding $\downarrow_x$. A variable x in an expression is *bound* if it occurs in the scope of a binding $\downarrow_x$. Otherwise it is *free*. We write $fv(r)$ to denote the set of free variables in r and $r(\bar{x})$ to denote that $\bar{x}$ is the sequence of all free variables. The semantics of an REWB $r(\bar{x})$ over the variables $\{x_1, \ldots, x_k\}$ is defined with respect to a partial valuation $\nu : \{x_1, \ldots, x_k\} \to \mathcal{D}$ of the variables. A valuation ν is *compatible* with $r(\bar{x})$ if $\nu(\bar{x})$ is defined.

▶ **Definition 2.2** (Semantics of REWB)**.** Let $r(\bar{x})$ be an REWB over $\Sigma[x_1, \ldots, x_k]$ and let $\nu : \{x_1, \ldots, x_k\} \to \mathcal{D}$ be a valuation of variables compatible with $r(\bar{x})$. The language of data words $L(r, \nu)$ defined by $r(\bar{x})$ with respect to ν is given as follows:

r	$L(r,\nu)$	r	$L(r,\nu)$	r	$L(r.\nu)$
ε	$\{\varepsilon\}$	a	$\{\binom{a}{d} \mid d \in \mathcal{D}\}$	$a[c]$	$\{\binom{a}{d} \mid d, \nu \models c\}$
$r_1 + r_2$	$L(r_1,\nu) \cup L(r_2,\nu)$	$r_1 \cdot r_2$	$L(r_1,\nu) \cdot L(r_2,\nu)$	r_1^*	$(L(r_1,\nu))^*$
$a \downarrow_{x_i} (r_1)$	$\bigcup_{d\in\mathcal{D}}\{\binom{a}{d}\} \cdot L(r_1, \nu[x_i \to d])$				

where $\nu[x_i \to d]$ denotes the valuation which is the same as ν except for x_i which is mapped to d. An REWB r defines the data language $L(r) = \bigcup_{\nu \text{ compatible with } r} L(r,\nu)$.

For example, the REWB $a \downarrow_x (b[x^=]^*)$ defines the set of data words of the form ab^* with all positions having the same data value. The REWB $(a \downarrow_x (b[x^=]))^*$ defines the set of data words of the form $\binom{a}{d_1}\binom{b}{d_1}\binom{a}{d_2}\binom{b}{d_2}\cdots\binom{a}{d_n}\binom{b}{d_n}$.

▶ **Definition 2.3** (Data graphs). A *data graph* G over a finite alphabet Σ and an infinite set of data values $\mathcal{D}$ is a pair (V, E) where V is a finite set of vertices, and $E \subseteq V \times \Sigma \times \mathcal{D} \times V$ is a set of edges which carry labels from $\Sigma \times \mathcal{D}$.

We do not have data values on vertices, but they can be introduced without affecting the results. A *regular data path query* is of the form $Q = x \xrightarrow{r} y$ where r is an REWB. Evaluating Q on a data graph G results in the set $Q(G)$ of pairs of nodes $\langle u, v\rangle$ such that there exists a data path from u to v and the sequence of labels along the data path forms a data word in $L(r)$. Evaluating a regular data path query on a data graph is known to be PSPACE-complete in general and NLOGSPACE-complete when the query is of constant size [15]. We sometimes identify the query Q with the expression r and write $r(G)$ for $Q(G)$. A query r_1 is said to be contained in another query r_2 if for every data graph G, $r_1(G) \subseteq r_2(G)$. It is known from [12, Proposition 3.5] that a query r_1 is contained in the query r_2 iff $L(r_1) \subseteq L(r_2)$. Hence, if a class E_2 of REWBs is more expressive than the class E_1 in terms of defining data languages, E_2 can also express more queries than E_1.

2.2 Parameterized Complexity

The size of queries are typically small compared to the size of databases. To analyze the efficiency of query evaluation algorithms, the size of the input can be naturally split into the size of the query and the size of the database. Parameterized complexity theory is a formal framework for dealing with such problems. An instance of a parameterized problem is a pair (x, k), where x is an encoding of the input structure on which the problem has to be solved (e.g., a data graph and a query), and k is a parameter associated with the input (e.g., the size of the query). A parameterized problem is said to be in the parameterized complexity class Fixed Parameter Tractable (FPT) if there is a computable function $f : \mathbb{N} \to \mathbb{N}$, a constant $c \in \mathbb{N}$ and an algorithm to solve the problem in time $f(k)|x|^c$.

We will see later that the query evaluation problem is unlikely to be in FPT, when parameterized by the size of the regular data path query. There are many parameterized complexity classes that are unlikely to be in FPT, like W[SAT], W[P], AW[SAT] and AW[P]. To place parameterized problems in these classes, we use FPT-reductions.

▶ **Definition 2.4** (FPT reductions). A FPT reduction from a parameterized problem Q to another parameterized problem Q' is a mapping R such that:

1. For all instances (x, k) of parameterized problems, $(x, k) \in Q$ iff $R(x, k) \in Q'$.
2. There exists a computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all (x, k), say with $R(x, k) = (x', k')$, we have $k' \leqslant g(k)$.
3. There exist a computable function $f : \mathbb{N} \to \mathbb{N}$ and a constant $c \in \mathbb{N}$ such that R is computable in time $f(k)|x|^c$.

3 Nesting Depth of Iterated Bindings and Expressive Power

A binding $\downarrow_x$ along with a condition $[x^{=}]$ or $[x^{\neq}]$ is used to constrain the possible data values that can occur at certain positions in a data word. A binding inside a star — an *iterated binding* — imposes the constraint arbitrarily many times. For instance, the expression $r_1 := (a_1 \downarrow_{x_1} (b_1[x_1^{=}]))^*$ defines data words in $(a_1 b_1)^*$ where every a_1 has the same data value as the next b_1. We now define a syntactic mechanism for controlling the nesting depth of iterated bindings. The restrictions result in an infinite hierarchy of expressions. The expressions at level i are generated by F_i in the grammar below, defined by induction on i.

$$F_0 ::= \varepsilon \mid a \mid a[c] \mid F_0 + F_0 \mid F_0 \cdot F_0 \mid F_0^*$$
$$E_i ::= F_{i-1} \mid E_i + E_i \mid E_i \cdot E_i \mid a \downarrow_{x_j} (E_i)$$
$$F_i ::= E_i \mid F_i + F_i \mid F_i \cdot F_i \mid F_i^*$$

where $i \geqslant 1$, $a \in \Sigma$, c is a condition in $\mathcal{C}_k$ and $x_j \in \{x_1, \ldots, x_k\}$. Intuitively, E_i can add bindings over iterations (occurring in F_{i-1}) and F_i can add iterations over bindings (occurring in E_i). The nesting depth of iterated bindings in an expression in F_i is therefore i. The union of all expressions at all levels equals the set of REWBs. In this paper, we use subscripts to denote the levels of expressions and superscripts to denote different expressions in a level: so e_5^1 is some expression in E_5, f_3^2 is some expression in F_3.

We now give a sequence of expressions $\{r_i\}_{i \geqslant 1}$ such that each r_i is in F_i but no language equivalent expression exists in F_{i-1}. For technical convenience, we use an unbounded number of letters from the finite alphabet and an unbounded set of variables. The results can be obtained with a constant number of letters and variables.

▶ **Definition 3.1.** Let $\{a_1, b_1, a_2, b_2, \ldots\}$ be an alphabet and $\{x_1, x_2, \ldots\}$ a set of variables. We define r_1 to be $(a_1 \downarrow_{x_1} (b_1[x_1^{=}]))^*$. For $i \geqslant 2$, define $r_i := (a_i \downarrow_{x_i} (r_{i-1} b_i[x_i^{=}]))^*$.

From the syntax, it can be seen that each r_i is in F_i. To show that $L(r_i)$ cannot be defined by any expression in F_{i-1}, we will use an "automaton view" of the expression, as this makes pigeon-hole arguments simpler. No automata characterizations are known for REWBs in general; the restrictions on the binding and star operators in the expressions of a given level help us build specific automata in stages.

Standard finite state automata can be converted to regular expressions by considering generalized non-deterministic finite automata, where transitions are labeled with regular expressions instead of a single letter (see e.g., [21, Lemma 1.32]). The language of an expression f_i^1 can be accepted by such an automaton, where transitions are labeled with expressions in E_i. We will denote this automaton by $\mathcal{A}(f_i^1)$. Similarly, the language of an expression e_i^1 can be accepted by an automaton whose transitions are labeled with expressions in F_{i-1} or with $a \downarrow_x$. We will denote this automaton by $\mathcal{A}(e_i^1)$. There are no cycles in $\mathcal{A}(e_i^1)$, since e_i^1 can not use the Kleene $*$ operator except inside expressions in F_{i-1}. The runs of $\mathcal{A}(e_i^1)$ are sequences of pairs of a state and a valuation for variables. The valuations are updated after every transition with a label of the form $a \downarrow_x$. Formal semantics are given in Appendix A of the full version of this paper, which also contains all the proofs in detail.

We will prove that $L(r_i)$ cannot be defined by any expression in E_i (and hence not by any expression in F_{i-1}). We first define the following sequence of words, which will be used in the proof. Let $\{d[j_1, j_2] \in \mathcal{D} \mid j_1, j_2 \in \mathbb{N}\}$ be a set of data values such that $d[j_1, j_2] \neq d[j_1', j_2']$

if $\langle j_1, j_2\rangle \neq \langle j_1', j_2'\rangle$. For every $n \geqslant 1$, define the words:

$$u_{1,n} := \binom{a_1}{d[1,1]}\binom{b_1}{d[1,1]}\binom{a_1}{d[1,2]}\binom{b_1}{d[1,2]} \cdots \binom{a_1}{d[1,n^2]}\binom{b_1}{d[1,n^2]}$$

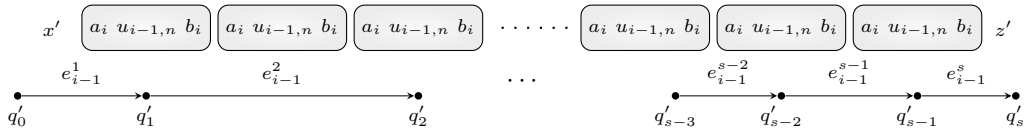
$$u_{i,n} := \binom{a_i}{d[i,1]}u_{i-1,n}\binom{b_i}{d[i,1]}\binom{a_i}{d[i,2]}u_{i-1,n}\binom{b_i}{d[i,2]}\cdots\binom{a_i}{d[i,n^2]}u_{i-1,n}\binom{b_i}{d[i,n^2]}$$

for all $i \geqslant 2$

In order to prove that $L(r_i)$ cannot be defined by any expression in E_i, we will show the following property: if $u_{i,n}$ occurs as a sub-word of a word w in the language of a "sufficiently small" expression e_i^1, then the same expression accepts a word where some a_j and a matching b_j have different data values. Let $Mismatch_{i,n}$ be the set of all data words obtained from $u_{i,n}$ by modifying the data values so that there exist two positions p, p' with $p < p'$ and a $j \leqslant i$ such that: p contains $\binom{a_j}{d}$ and p' contains $\binom{b_j}{d'}$ with $d \neq d'$; moreover between positions p and p', b_j does not occur in the word. We consider expressions in which no two occurrences of the binding operator use the same variable. For an expression e, let $|\mathcal{A}(e)|$ denote the number of states in the automaton $\mathcal{A}(e)$ and $|var(e)|$ denote the number of variables in e.

▶ **Lemma 3.2.** *Let e_i^1 be an expression and let $n \in \mathbb{N}$ be greater than $(|\mathcal{A}(e)| + 1)$ and $(|var(e)| + 1)$ for every sub-expression e of e_i^1. Let ν be a valuation of $fv(e_i^1)$ and let x, z be data words. Then: $xu_{i,n}z \in L(e_i^1, \nu) \implies x\bar{u}_{i,n}z \in L(e_i^1, \nu)$ for some $\bar{u}_{i,n} \in Mismatch_{i,n}$.*

Proof idea. By induction on i. Suppose $xu_{i,n}z \in L(e_i^1, \nu)$. The run of $\mathcal{A}(e_i^1)$ on $xu_{i,n}z$ consists of at most n transitions, since the automaton is acyclic and has at most n states. Each of the (at most) n transitions reads some sub-word in the language of some sub-expression f_{i-1}^1, while the whole word consists of n^2 occurrences of $a_i u_{i-1,n} b_i$. Hence, at least one sub word consists of n occurrences of $a_i u_{i-1,n} b_i$. A run of $\mathcal{A}(f_{i-1}^1)$ on such a sub-word is shown below.



Every transition of this run reads sub-words in the language of some sub-expression e_{i-1}^j. If some transition of this run reads an entire sub-word $u_{i-1,n}$ (as in transition $q_1' \longrightarrow q_2'$), then we can create a mismatch inside this $u_{i-1,n}$ by induction hypothesis. Otherwise, none of the transitions read an a_i and the corresponding b_i together (as in $q_{s-2}' \longrightarrow q_{s-1}'$ in the figure). None of the b_is is compared with the corresponding a_i, so the data value of one of the b_is can be changed to create a mismatch. The resulting data word will be accepted provided the change does not result in a violation of some condition. Since the range of the valuation has at most $(n-1)$ distinct values, one of the n b_is is safe for changing the data value. ◀

▶ **Theorem 3.3.** *For any i, the language $L(r_i)$ cannot be defined by any expression in E_i.*

Proof. Suppose r_i is equivalent to an expression e_i^1. Pick an n bigger than $|\mathcal{A}(e)|$ and $|fv(e)|$ for every sub-expression e of e_i^1. The word $u_{i,n}$ belongs to $L(r_i)$ and hence $L(e_i^1)$. By Lemma 3.2, we know that if this is the case, then $\bar{u}_{i,n} \in L(r_i)$ for some word $\bar{u}_{i,n} \in Mismatch_{i,n}$. But $L(r_i)$ cannot contain words with a mismatch. A contradiction. ◀

Given an expression at some level, it is possible that its language is defined by an expression at lower levels. Next we show that it is undecidable to check this.

▶ **Theorem 3.4.** *Given an expression in F_{i+1}, checking if there exists a language equivalent expression in F_i is undecidable.*

Proof idea. By reduction from Post's Correspondence Problem (PCP). The basic idea is from the proof of undecidability of universality of REWBs and related formalisms [17, 15]. For an instance $\{(u_1, v_1), \ldots, (u_n, v_n)\}$ of PCP, a solution (if it exists) can be encoded by a data word of the form $w_1 \# r_i \# w_2$, where w_1 is made up of u_i's, w_2 is made up of v_i's and r_i is from Definition 3.1. To ensure that such a data word indeed represents a solution, we need to match up the u_i's in w_1 with the v_i's in w_2, which can be done through matching data values. Consider the language of data words of the form $w_1' \# r_i \# w_2'$ that are *not* solutions of the given PCP instance. This language can be defined by an expression Δ in E_{i+1}, which compares data values in the left of $\# r_i \#$ with those on the right side, to catch mismatches. We can prove that no equivalent expression exists in lower levels, using techniques used in Lemma 3.2. On the other hand, if the given PCP instance does not have a solution, no data word encodes a solution, so the given language is defined by $\Sigma^* r_i \Sigma^*$, which is in F_i. ◀

4 Complexity of Query Evaluation

In this section, we will study how the depth of nesting of iterated bindings affects the complexity of evaluating queries. An instance of the query evaluation problem consists of a data graph G, an REWB e, a valuation ν for $fv(e)$ and a pair $\langle u, v \rangle$ of nodes in G. The goal is to check if u is connected to v by a data path in $L(e, \nu)$.

4.1 Upper Bounds

An expression in F_i can be thought of as a standard regular expression (without data values) over the alphabet of its sub-expressions. This is the main idea behind our upper bound results. The main result proves that evaluating queries in E_i can be done in Σ_i in the polynomial time hierarchy.

▶ **Lemma 4.1.** *With an oracle for evaluating E_i queries, F_i queries can be evaluated in polynomial time.*

Proof idea. Suppose the query f_i^1 is to be evaluated on the data graph G and f_i^1 consists of the sub-expressions $e_i^1, \ldots, e_i^m$ in E_i. For every j, add an edge labeled e_i^j between those pairs $\langle v_1, v_2 \rangle$ of nodes of G for which $\langle v_1, v_2 \rangle$ is in the evaluation of e_i^j on G. Evaluating the sub-expressions can be done with the oracle. Now f_i^1 can be treated as a standard regular expression over the finite alphabet $\{e_i^1, \ldots, e_i^m\}$, and can be evaluated in polynomial time using standard automata theoretic techniques. ◀

▶ **Theorem 4.2.** *For queries in E_i, the evaluation problem belongs to Σ_i.*

Proof idea. Since bindings in E_i are not iterated, each binding is performed at most once. The data value for each variable is guessed non-deterministically. The expression can be treated as a standard regular expression over its sub-expressions and the guessed data values. The sub-expressions are in F_{i-1}, which can be evaluated in polynomial time (Lemma 4.1) with an oracle for evaluating queries in E_{i-1}. This argument will not work in general for arbitrary REWBs — bindings that are nested deeply inside iterations and other bindings may occur more than polynomially many times in a single path. ◀

Next we consider the query evaluation problem with the size of the query as the parameter. An instance of the *parameterized weighted circuit satisfiability* problem consists of a Boolean circuit and the parameter $k \in \mathbb{N}$. The goal is to check if the circuit can be satisfied by a truth assignment of weight k (i.e., one that sets exactly k propositional atoms to true). The class W[P] is the set of all parameterized problems which are FPT-reducible to the weighted circuit satisfiability problem.

▶ **Theorem 4.3.** *Evaluating REWB queries in E_1, parameterized by the size of the query is in* W[P].

Proof idea. It is proved in [3, Lemma 7, Theorem 8] that a parameterized problem is in W[P] iff there is a non-deterministic Turing machine that takes an instance (x, k) and decides the answer within $f(k)|x|^c$ steps, of which at most $f(k)\log|x|$ are non-deterministic (for some computable function f and a constant c). Such a Turing machine exists for evaluating REWB queries in E_1, using the steps outlined in the proof idea of Theorem 4.2. ◀

Thus, the number of non-deterministic steps needed to evaluate an E_1 query depends only logarithmically on the size of the data graph. It is also known that W[P] is contained in the class PARA-NP — the class of parameterized problems for which there are deterministic algorithms taking instances (x, k) and computing an equivalent instance of the Boolean satisfiability problem in time $f(k)|x|^c$. Hence, we can get an efficient reduction to the satisfiability problem, on which state of the art SAT solvers can be run. Many hard problems in planning fall into this category [7].

We next consider the parameterized complexity of evaluating queries at higher levels. The parameterized class *uniform*-XNL is the class of parameterized problems Q for which there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ and a non-deterministic algorithm that, given a pair (x, k), decides if $(x, k) \in Q$ in space at most $f(k)\log|x|$ [3, Proposition 18].

▶ **Theorem 4.4.** *Evaluating REWB queries, with size of the query as parameter, is in uniform-*XNL.

Proof idea. Let k be the size of the query e_i^1 to be evaluated, on a data graph with n nodes. Suppose a pair of nodes is connected by a data path w in $L(e_i^1)$. Iterations in e_i^1 can only occur inside its F_{i-1} sub-expressions. Hence w consists of at most k sub-paths, each sub-path w_j in the language of some sub-expression f_{i-1}^j. When f_{i-1}^j is considered as a standard regular expression over its sub-expressions (in E_{i-1}), there are no bindings. By a standard pigeon hole principle argument, we can infer that w_j consists of at most kn sub-paths, each one in the language of some sub-expression e_{i-1}^1. This argument can be continued to prove that w is of length at most $(k^2n)^i$. The existence of such a path can be guessed and verified by a non-deterministic Turing machine in space $\mathcal{O}(ik^2 \log n)$. ◀

4.2 Lower Bounds

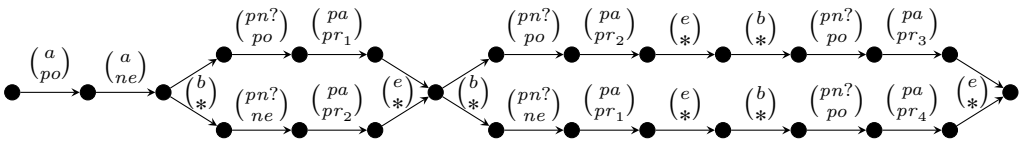
We obtain our lower bounds by reducing various versions of the Boolean formula satisfiability problem to query evaluation. We begin by describing a schema for reducing the problem of evaluating a Boolean formula on a given truth assignment to the problem of evaluating a query on a data graph. The basic ideas for the gadgets we construct below are from [15, proofs of Proposition 2, Theorem 5]. We will need to build on these ideas to address finer questions about the complexity of query evaluation.

Suppose the propositional atoms used in the Boolean formula are among $\{pr_1, \ldots, pr_n\}$. We use $pr_1, \ldots, pr_n$ also as data values. An edge labeled $\binom{pa}{pr_j}$ indicates the propositional

atom pr_j occurring in a sub-formula. The data values po and ne appear on edges labeled with the letter $pn?$, to indicate if a propositional atom appears positively or negatively. The symbol $*$ denotes an arbitrary data value different from all others. We will assume that the Boolean formula is in negation normal form, i.e., negation only appears in front of propositional atoms. This restriction does not result in loss of generality, since any Boolean formula can be converted into an equi-satisfiable one in negation normal form with at most linear blowup in the size. The data graph is a series parallel digraph with a source and a sink, defined as follows by induction on the structure of the Boolean formula.

- Positively occurring propositional atom pr_j: $\cdot \xrightarrow{\binom{b}{*}} \cdot \xrightarrow{\binom{pn?}{po}} \cdot \xrightarrow{\binom{pa}{pr_j}} \cdot \xrightarrow{\binom{e}{*}} \cdot$.
- Negatively occurring propositional atom pr_j: $\cdot \xrightarrow{\binom{b}{*}} \cdot \xrightarrow{\binom{pn?}{ne}} \cdot \xrightarrow{\binom{pa}{pr_j}} \cdot \xrightarrow{\binom{e}{*}} \cdot$.
- $\phi_1 \wedge \cdots \wedge \phi_r$: inductively construct the data graphs for the conjuncts, then do a standard serial composition, by fusing the sink of one graph with the source of the next one.
- $\phi_1 \vee \cdots \vee \phi_r$: inductively construct the data graphs for the disjuncts, then do a standard parallel composition, by fusing all the sources into one node and all the sinks into another node.
- After the whole formula is handled, the source of the resulting graph is fused with the sink of the following graph: $\cdot \xrightarrow{\binom{a}{po}} \cdot \xrightarrow{\binom{a}{ne}} \cdot$.

Let G_ϕ denote the data graph constructed above for formula ϕ. The data graph G_ϕ is shown below for $\phi = (pr_1 \vee \neg pr_2) \wedge ((pr_2 \wedge pr_3) \vee (\neg pr_1 \wedge pr_4))$.



The query uses $x_1, \ldots, x_k$ to remember the propositional atoms that are set to true.

$$e_{eval}[k] := a \downarrow_{x_{po}} (a \downarrow_{x_{ne}} (\quad (1)$$
$$(b(pn?[x^{=}_{po}] \cdot pa[x^{=}_1 \vee \cdots \vee x^{=}_k] + pn?[x^{=}_{ne}] \cdot pa[x^{\neq}_1 \wedge \cdots \wedge x^{\neq}_k])e)^* \ )) \ .$$

▶ **Lemma 4.5.** *Let ϕ be a Boolean formula over the propositional atoms $pr_1, \ldots, pr_n$ and $\nu : \{x_1, \ldots, x_k\} \to \{pr_1, \ldots, pr_n, *\}$ be a valuation. The source of G_ϕ is connected to its sink by a data path in $L(e_{eval}[k], \nu)$ iff ϕ is satisfied by the truth assignment that sets exactly the propositions in $\{pr_1, \ldots, pr_n\} \cap Range(\nu)$ to true.*

Proof idea. The two bindings in the beginning of $e_{eval}[k]$ forces x_{po}, x_{ne} to contain po, ne respectively. A positively occurring propositional atom generates a data path of the form $\cdot \xrightarrow{\binom{b}{*}} \cdot \xrightarrow{\binom{pn?}{po}} \cdot \xrightarrow{\binom{pa}{pr_j}} \cdot \xrightarrow{\binom{e}{*}} \cdot$, which can only be in the language of the expression $b \cdot pn?[x^{=}_{po}] \cdot pa[x^{=}_1 \vee \cdots \vee x^{=}_k]e$. This forces pr_j to be contained in one of $x_1, \ldots, x_k$. Similar arguments works for negatively occurring atoms. Rest of the proof is by induction on the structure of the formula. ◀

▶ **Theorem 4.6.** *For queries in E_1, the evaluation problem is* NP*-hard.*

Proof idea. To check if a Boolean formula ϕ is satisfiable, evaluate the query $a \downarrow_{x_1} a \downarrow_{x_2} \cdots a \downarrow_{x_n} e_{eval}[n]$ on the data graph $\cdot \xrightarrow{\binom{a}{pr_1/*}} \cdot \xrightarrow{\binom{a}{pr_2/*}} \cdots \xrightarrow{\binom{a}{pr_n/*}} \cdot - G_\phi \to \cdot$. Here, $\xrightarrow{\binom{a}{pr_j/*}}$ denotes two edges in parallel, one labeled with $\binom{a}{pr_j}$ and another with $\binom{a}{*}$. ◀

Evaluating queries in E_1 is NP-complete, evaluating REWB queries in general is PSPACE-complete and evaluating queries in E_i is in Σ_i. To prove a corresponding Σ_i lower bound, one would need to simulate Σ_i computations using queries with bounded depth of nesting of iterated bindings. However, this does not seem to be possible. We take a closer look at this in the rest of the paper. Finding the exact complexity of evaluating queries in E_i remains open.

We now extend our satisfiability-to-query evaluation schema to handle Boolean quantifiers. Let $PR = \{pr_1, \ldots, pr_n\}$ be a set of propositional atoms. To handle existential Boolean quantifiers, we build a new graph and a query. These gadgets build on earlier ideas to bring out the difference in the role played by the data graph and the query while reducing satisfiability to query evaluation. The new graph $G[\exists k/PR] \circ G$, is as follows: $\cdot \xrightarrow{\binom{a_1}{pr_1}} \cdot \xrightarrow{\binom{a_1}{pr_2}} \cdots \xrightarrow{\binom{a_1}{pr_n}} \cdot - G \rightarrow \cdot$. We assume that the letter a_1 is not used inside G, which is equal to G_ϕ for some Boolean formula ϕ. The new query $e[\exists k] \circ e$ is defined as follows:

$$e[\exists k] \circ e := a_1^* a_1 \downarrow_{x_1} a_1^* a_1 \downarrow_{x_2} a_1^* \cdots a_1^* a_1 \downarrow_{x_k} a_1^* e \tag{2}$$

where $e = e_{eval}[k]$ for some $k \in \mathbb{N}$.

We now give a parameterized lower bound for evaluating E_1 queries. An instance of the *weighted satisfiability* problem consists of a Boolean formula (not necessarily in Conjunctive Normal Form) and a parameter $k \in \mathbb{N}$. The goal is to check if the formula is satisfied by a truth assignment of weight k. The class W[SAT] is the set of all parameterized problems that are FPT-reducible to the weighted satisfiability problem (see [8, Chapter 25]).

▶ **Lemma 4.7.** *Let ϕ be a Boolean formula over the set PR of propositions and $k \in \mathbb{N}$. We can construct in polynomial time a data graph G and an REWB e_1^1 satisfying the following conditions.*

1. *The source of G is connected to its sink by a data path in $L(e_1^1)$ iff ϕ has a satisfying assignment of weight k.*
2. *The size of e_1^1 depends only on k.*

Proof idea. The required data graph is $G[\exists k/PR] \circ G_\phi$ and e_1^1 is $e[\exists k] \circ e_{eval}[k]$. The data path $\cdot \xrightarrow{\binom{a_1}{pr_1}} \cdot \xrightarrow{\binom{a_1}{pr_2}} \cdots \xrightarrow{\binom{a_1}{pr_n}} \cdot$ in the graph $G[\exists k/PR] \circ G_\phi$ has to be in the language of $a_1^* a_1 \downarrow_{x_1} a_1^* a_1 \downarrow_{x_2} a_1^* \cdots a_1^* a_1 \downarrow_{x_k} a_1^*$. This induces a valuation ν' which maps $\{x_1, \ldots, x_k\}$ injectively into PR, denoting the k propositions that are set to true. With this the data path continues from the source of G_ϕ to its sink. Rest of the proof follows from Lemma 4.5. ◀

▶ **Theorem 4.8.** *Evaluating REWB queries in E_1, parameterized by the size of the query is hard for* W[SAT] *under* FPT *reductions.*

Proof. The reduction given in Lemma 4.7 is a FPT reduction from the weighted satisfiability problem to the problem of evaluating E_1 queries , parameterized by the size of the query. ◀

Finally we extend our gadgets to handle universal Boolean quantifiers. These gadgets build upon the previous ideas and bring out the role of nested iterated bindings when satisfiability is reduced to query evaluation. We would first like to check if the source of some graph G is connected to its sink by a data path in the language of some REWB e, for every possible injective valuation $\nu : \{x_1, \ldots, x_k\} \rightarrow PR$. We will now design some data graphs and expressions to achieve this. Let *skip* be a letter not used in G. The data graphs $G_0, \ldots, G_k$ are as shown in Figure 1.

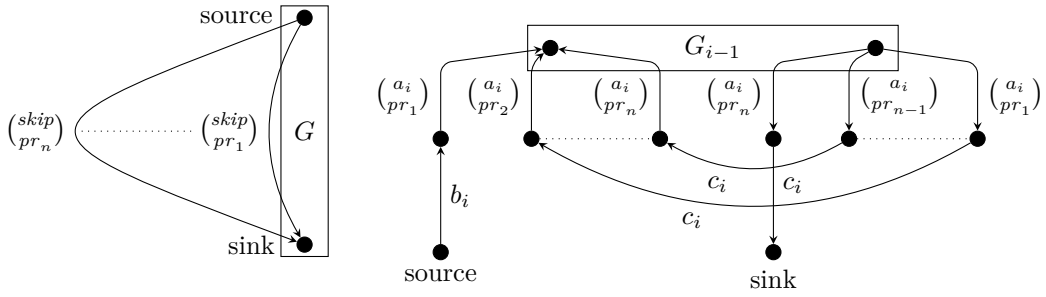


Figure 1 Data graphs G_0 (left) and G_i (right).

The expressions $e^0, \ldots, e^k$ are as follows.

$$e^0 := e + \bigdoublebarwedge_{1 \leqslant i < j \leqslant k} skip[x_i^{=} \wedge x_j^{=}] \qquad e^i := b_i(a_i \downarrow_{x_i} (e^{i-1} a_i[x_i^{=}])c_i)^* \tag{3}$$

The graph G_0 and the expression e^0 are designed to ensure that the source of G is connected to its sink by a path in $L(e, \nu)$, unless ν is not injective, in which case G can be bypassed by one of the edges labeled $\binom{skip}{pr_j}$ introduced in G_0. The graph G_i and the expression e_i are designed to ensure that any path from the source of G_i to its sink has to go through G_{i-1} multiple times, once for each pr_j stored in the variable x_i. The nesting depth of iterated bindings in the expression e^i is one more than that of e^{i-1}.

Suppose ν is a partial valuation of some variables, whose domain does not intersect with $\{x_1, \ldots, x_k\}$. We denote by $\nu[\{x_1, \ldots, x_k\} \to PR]$ the set of valuations ν' that extend ν such that $\text{domain}(\nu') = \text{domain}(\nu) \cup \{x_1, \ldots, x_k\}$ and $\{\nu'(x_1), \ldots, \nu'(x_k)\} \subseteq PR$. We additionally require that ν' is injective on $\{x_1, \ldots, x_k\}$ when we write $\nu[\{x_1, \ldots, x_k\} \xrightarrow{1:1} PR]$.

▶ **Lemma 4.9.** *Let $i \in \{1, \ldots, k\}$ and ν_i be a valuation for $fv(e^i) \backslash \{x_1, \ldots, x_i\}$. The source of G_i is connected to its sink by a data path in $L(e^i, \nu_i)$ iff for every $\nu \in \nu_i[\{x_1, \ldots, x_i\} \to PR]$, there is a data path in $L(e^0, \nu)$ connecting the source of G_0 to its sink.*

Proof idea. The data path has to begin with $b_i\binom{a_i}{pr_1}$ in the language of $b_i a_i \downarrow_{x_i}$, forcing x_i to store pr_1. Then the path has to traverse G_{i-1} using e^{i-1}. At the sink of G_{i-1}, the path is forced to take $\binom{a_i}{pr_1}c_i$ to satisfy the condition in $a_i[x_i^{=}]c_i$. This forces the path to start again in $\binom{a_i}{pr_2}$ and so on. ◀

We write $G[\forall k/PR] \circ G$ and $e[\forall k] \circ e$ to denote the graph G_k and REWB e^k constructed above. We implicitly assume that the variables $x_1, \ldots, x_k$ are not bound inside e. We can always rename variables to ensure this. If e is in E_i, then $e[\forall k] \circ e$ is in F_{i+k-1}.

▶ **Lemma 4.10.** *Let ν be a valuation for $fv(e) \backslash \{x_1, \ldots, x_k\}$ for some REWB e. The source of $G[\forall k/PR] \circ G$ is connected to its sink by a data path in $L(e[\forall k] \circ e, \nu)$ iff for all $\nu' \in \nu[\{x_1, \ldots, x_k\} \xrightarrow{1:1} PR]$, the source of G is connected to its sink by a data path in $L(e, \nu')$.*

Proof idea. Lemma 4.9 ensures that there is a path $w_{\nu'}$ in $L(e^0, \nu')$ connecting the source of G_0 to its sink for every valuation $\nu' \in \nu[\{x_1, \ldots, x_k\} \to PR]$. From Figure 1, $w_{\nu'}$ can either be a *skip* edge, or a path through G. By definition, e^0 allows a skip edge to be taken only when two variables among $x_1, \ldots, x_k$ have the same data value. Hence for valuations ν' that are injective on $\{x_1, \ldots, x_k\}$, $w_{\nu'}$ is in $L(e, \nu')$. ◀

If ϕ is a partially quantified Boolean formula with the propositional atoms in PR occurring freely, we write $\exists^k PR\ \phi$ to denote that atoms in PR are existentially quantified with the constraint that exactly k of them should be set to true. We write $\forall^k PR\ \phi$ to denote that atoms in PR are universally quantified and that only those assignments that set exactly k of the atoms to true are to be considered. An instance of the *weighted quantified satisfiability* problem consists of a Boolean formula ϕ over the set PR of propositional atoms, a partition $PR_1, \ldots, PR_\ell$ of PR and numbers $k_1, \ldots, k_\ell$. The goal is to check if $(\exists^{k_1} PR_1 \forall^{k_2} PR_2 \cdots \phi)$ is true.

▶ **Lemma 4.11.** *Given an instance of the weighted quantified satisfiability problem, We can construct in polynomial time a data graph G and an REWB $e^1_{1+k_2+k_4+\cdots}$ satisfying the following conditions.*

1. *The source of G is connected to its sink by a data path in $L(e^1_{1+k_2+k_4+\cdots})$ iff the given instance of the weighted quantified satisfiability problem is a yes instance.*
2. *The size of $e^1_{1+k_2+k_4+\cdots}$ depends only on $k_1, \ldots, k_\ell$.*

Proof idea. The required data graph G is $G[\exists k_1/PR_1] \circ G[\forall k_2/PR_2] \circ \cdots \circ G_\phi$ and the required REWB $e^1_{1+k_2+k_4+\cdots}$ is $e[\exists k_1] \circ e[\forall k_2] \circ \cdots \circ e_{eval}[k_1 + \cdots + k_\ell]$. We assume that $\circ$ associates to the right, so $G_1 \circ G_2 \circ G_3$ is $G_1 \circ (G_2 \circ G_3)$ and $e^1 \circ e^2 \circ e^3$ is $e^1 \circ (e^2 \circ e^3)$. Correctness follows from Lemma 4.10 and Lemma 4.5. ◀

The weighted quantified satisfiability problem is parameterized by $\ell + k_1 + \cdots + k_\ell$. The class AW[SAT] is the set of parameterized problems that are FPT-reducible to the weighted quantified satisfiability problem (see [8, Chapter 26]).

▶ **Theorem 4.12.** *Evaluating REWB queries, parameterized by the size of the query is hard for* AW[SAT] *under* FPT *reductions.*

Proof. The reduction given in Lemma 4.11 is a FPT reduction from the weighted quantified satisfiability problem to the problem of evaluating REWB queries, with query size as the parameter. ◀

5 Summary and Open Problems

We have proved that increasing the depth of nesting of iterated bindings in REWBs increase expressiveness. Given an REWB, it is undecidable to check if its language can be defined with another REWB with smaller depth of nesting of iterated bindings. The complexity of query evaluation problems are summarized in the following table, followed by a list of technical challenges to be overcome for closing the gaps.

Query level	Evaluation	Parameterized complexity, query size is parameter
E_1	NP-complete	(?2)W[SAT] lower bound, W[P] upper bound
E_i, $i > 1$	(?1), Σ_i upper bound	(?3)
Unbounded	PSPACE-complete [15]	(?4)AW[SAT] lower bound, uniform-XNL upper bound

1. Suppose we want to check the satisfiability of a Σ_2 Boolean formula over $(n_e + n_u)$ propositional atoms of which the first n_e atoms are existentially quantified and the last n_u are universally quantified. With currently known techniques, reducing this to query evaluation results in an REWB in $E_{(n_u+1)}$. Hence, with bounded nesting depth, we cannot even prove a Σ_2 lower bound.

2. Weighted formula satisfiability, complete for W[SAT], can be simulated with series-parallel graphs. Queries in E_1 do not seem to be powerful enough for weighted circuits.
3. Without parameterization, the Σ_i upper bound is obtained by an oracle hierarchy of NP machines. With parameterization, an oracle hierarchy of W[P] machines does not correspond to any parameterized complexity class. See [3, Section 4] for discussions on subtle points which make classical complexity results fail in parameterized complexity.
4. As in point 2, here one might hope for a AW[P] lower bound, which is quantified weighted circuit satisfiability (stronger than AW[SAT], which is quantified weighted formula satisfiability). Even if this improvement can be made, there is another classical complexity result not having analogous result in parameterized complexity: not much is known about the relation between parameterized alternating time bounded class (AW[P]) and parameterized space bounded class (uniform-XNL).

Acknowledgements. The authors thank Partha Mukhopadhyay and Geevarghese Philip for helpful discussions about polynomial time hierarchy and parameterized complexity theory.

References

1 P. Barceló. Querying graph databases. In *Proceedings of PODS*, pages 175–188, New York, NY, USA, 2013. ACM.
2 P. Barceló, J. Reutter, and L. Libkin. Parameterized regular expressions and their languages. *Theoretical Computer Science*, 474:21–45, 2013.
3 Yijia Chen, J. Flum, and M. Grohe. Bounded nondeterminism and alternation in parameterized complexity theory. In *Computational Complexity, 2003*, pages 13–29, 2003.
4 T. Colcombet and A. Manuel. Generalized data automata and fixpoint logic. In *FSTTCS*, volume 29 of *LIPIcs*, pages 267–278, 2014.
5 T. Colcombet and A. Manuel. Combinatorial expressions and lower bounds. In *STACS*, volume 30 of *LIPIcs*, pages 249–261, 2015.
6 T. Colcombet and A. Manuel. Fragments of fixpoint logic on data words. In *FSTTCS*, volume 45 of *LIPIcs*, pages 98–111, 2015.
7 R. De Haan, M. Kronegger, and A. Pfandler. Fixed-parameter tractable reductions to sat for planning. In *Proceedings of IJCAI*, pages 2897–2903, 2015.
8 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Thomson Brooks/Cole, 1997.
9 L. C. Eggan. Transition graphs and the star-height of regular events. *Michigan Math. J.*, 10(4):385–397, 1963.
10 O. Grumberg, O. Kupferman, and S. Sheinvald. Variable automata over infinite alphabets. In *LATA*, volume 6031 of *LNCS*, pages 561–572, 2010.
11 C. Gutierrez, C. Hurtado, and A. Mendelzon. Foundations of semantic web databases. *JCSS*, 77(3):520–541, 2011.
12 E.V. Kostylev, J.L. Reutter, and D. Vrgoč. Containment of data graph queries. In *ICDT*, pages 131–142, 2014.
13 U. Leser. A query language for biological networks. *Bioinformatics*, 21(suppl 2):ii33–ii39, 2005.
14 L. Libkin, W. Martens, and D. Vrgoč. Querying graph databases with xpath. In *ICDT*, pages 129–140, New York, NY, USA, 2013. ACM.
15 L. Libkin, T. Tan, and D. Vrgoč. Regular expressions with binding over data words for querying graph databases. In *DLT*, volume 7907 of *LNCS*, pages 325–337, 2013.
16 L. Libkin and D. Vrgoč. Regular path queries on graphs with data. In *ICDT*, pages 74–85, 2012.

17 F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, 2004.

18 W3C Recommendation. Sparql 1.1 query language. 21 March 2013.

19 R. Ronen and O. Shmueli. Soql: A language for querying and creating data in social networks. In *ICDE*, pages 1595–1602, 2009.

20 L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL*, volume 4207 of *LNCS*, pages 41–57, 2006.

21 M. Sipser. *Introduction to the Theory of Computation*. Springer, 2013.

22 T. Tan. Graph reachability and pebble automata over infinite alphabets. *ACM Trans. Comput. Logic*, 14(3):19:1–19:31, 2013.

23 D. Vrgoč. Using variable automata for querying data graphs. *Information Processing Letters*, 115(3):425–430, 2015.

A Hierarchy of Local Decision*

Laurent Feuilloley[1], Pierre Fraigniaud[2], and Juho Hirvonen[3]

1 **Institut de Recherche en Informatique Fondamentale (IRIF), CNRS and University Paris Diderot, Paris, France**
laurent.feuilloley@liafa.univ-paris-diderot.fr

2 **Institut de Recherche en Informatique Fondamentale (IRIF), CNRS and University Paris Diderot, Paris, France**
pierre.fraigniaud@liafa.univ-paris-diderot.fr

3 **Helsinki Institute for Information Technology (HIIT), Department of Computer Science, Aalto University, Aalto, Finland**
juho.hirvonen@aalto.fi

Abstract

We extend the notion of *distributed decision* in the framework of distributed network computing, inspired by recent results on so-called *distributed graph automata*. We show that, by using distributed decision mechanisms based on the interaction between a *prover* and a *disprover*, the size of the certificates distributed to the nodes for certifying a given network property can be drastically reduced. For instance, we prove that minimum spanning tree can be certified with $O(\log n)$-bit certificates in n-node graphs, with just one interaction between the prover and the disprover, while it is known that certifying MST requires $\Omega(\log^2 n)$-bit certificates if only the prover can act. The improvement can even be exponential for some simple graph properties. For instance, it is known that certifying the existence of a nontrivial automorphism requires $\Omega(n^2)$ bits if only the prover can act. We show that there is a protocol with two interactions between the prover and the disprover enabling to certify nontrivial automorphism with $O(\log n)$-bit certificates. These results are achieved by defining and analysing a *local hierarchy* of decision which generalizes the classical notions of *proof-labelling schemes* and *locally checkable proofs*.

1998 ACM Subject Classification D.1.3 Concurrent Programming (Distributed programming), F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Distributed Network Computing, Distributed Algorithm, Distributed Decision, Locality

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.118

1 Introduction

This paper is tackling the long-standing issue of characterizing the power of local computation in the framework of distributed network computing [27]. Our concern is the ability to design *local* algorithms, defined as distributed algorithms in which every node of a network (i.e., every computing entity in the system) can compute its output after having consulted only nodes in its vicinity. That is, communications proceed along the links of the network, and, in a local algorithm, every node must output after having exchanged information with nodes at constant distance only. A *construction* task consists, for the nodes of a network $G = (V, E)$ where each node u is given an input $x(u)$, to collectively and concurrently compute a collection

* The first and second authors received additional supports from ANR project DISPLEXITY, and from INRIA project GANG.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 118; pp. 118:1–118:15

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

$y(u)$, $u \in V$, of individual outputs, such that (G, x, y) satisfies some property characterizing the task to be solved. For instance, the minimum-weight spanning tree (MST) task consists, given the weights $x(u)$ of all the incident edges of every node u, in computing a subset $y(u)$ of edges incident to u such that the set $\{y(u), u \in V\}$ forms a MST in G. Similarly, the maximal independent set (MIS) task consists of computing $y(u) \in \{0, 1\}$, $u \in V$, such that the set $\{u \in V : y(u) = 1\}$ forms an MIS. It is an easy observation that the MST task cannot be solved locally as the weights of far-away edges may impact the output of a node. In a seminal result Linial showed that the same is true for MIS [24]: there is no local algorithm for constructing an MIS, even on an n-node ring. Nevertheless, there are many construction tasks that can be solved locally, such as approximate solutions of NP-hard graph problems (see, e.g., [8, 21, 22, 23]). In general it is Turing-undecidable whether or not a construction task can be solved locally [26].

Interestingly, the Turing-undecidability result of Naor and Stockmeyer [26] concerning the locality of construction tasks holds even if one restricts the question to properties that can be locally decided. A distributed *decision* task [1, 13] consists, given an input $x(u)$ to every node in a network G, in deciding whether (G, x) satisfies some given property. An instance is accepted by a distributed algorithm if and only if every node individually accepts (i.e., every node u outputs $y(u) = \text{true}$). For instance, proper colouring can easily be decided locally by having each node merely comparing its colour with the ones of its neighbours. On the contrary, deciding whether a collection of edges defined by $\{x(u), u \in V\}$ forms a MST is not possible locally (in fact, even separating paths from cycles is not possible locally). Similarly to the sequential computing setting, there are strong connections between the construction variant of a task and the ability to locally decide the legality of a given candidate solution for the same task, as illustrated by, e.g., the derandomization results in [6, 26], and the approximation algorithms in [29]. These connections have motivated work focusing on the basic question: what can be decided locally? This paper is aiming at pushing further our current knowledge on this question.

Two specific lines of work have motivated our approach of local decision in this paper. The first line of research is related to the notion of *proof-labelling schemes* introduced by Korman et al. [20], who showed that while not all graph properties can be decided locally, they can all be *verified* locally, with the help of local *certificates* provided by a prover. Unfortunately, there are natural graph properties (e.g., the existence of a non-trivial automorphism) which require $\Omega(n^2)$-bit certificates to be verified by any local distributed algorithm [16]. Göös and Suomela introduced the more practical class LogLCP of all graph properties that can be verified using certificates of size $O(\log n)$ bits [16], i.e., merely the size required to store the identities of the nodes. The class LogLCP contains non locally decidable properties such as hamiltonicity and non-bipartiteness. LogLCP even contains graph properties that are not in NP. Also, all existential-MSO graph properties are shown to be in LogLCP.

The second line of research which motivated our approach is the study of *distributed graph automata*. In particular, [28] recently proved that an analogue of the polynomial hierarchy, where sequential polynomial-time computation is replaced by distributed local computation, turns out to coincide with MSO. However, while this result is important for our understanding of the computational power of finite automata, the model does not quite fit with the standard model of distributed computing aiming at capturing the power of large-scale computer networks (see, e.g., [27]). Indeed, on the one hand, the model in [28] is somewhat weaker than desired, by assuming a finite-state automaton at each node instead of a Turing machine, and by assuming anonymous computation instead of the presence of unique node identities. On the other hand, the very same model is also stronger than the standard

model, by assuming a decision-making mechanism based on an arbitrary mapping from the collection of all node states to $\{\text{true}, \text{false}\}$. Instead, the classical distributed decision mechanism is based on the logical conjunction of the individual decisions. This is crucial as this latter decision mechanism provides the ability for every node rejecting the current instance to raise an alarm, and/or to launch a recovery procedure, without having to collect all of the individual decisions.

In this paper, our objective is to push further the study initiated in [16] on the LogLCP class, by adopting the approach of [28]. Indeed, LogLCP can be seen as the first level Σ_1 of a *local hierarchy* $(\Sigma_k, \Pi_k)_{k\geq 0}$, where $\Sigma_0 = \Pi_0 = \mathsf{LD}$, the class of properties that can be locally decided [13], and, for $k \geq 1$, Σ_k is the class of graph properties for which there exists a local algorithm A such that, for every instance (G, x),

$$(G, x) \text{ is legal } \iff \exists \ell_1 \forall \ell_2 \exists \ell_3 \ldots Q\ell_k : A(G, x, \ell_1, \ell_2, \ldots, \ell_k) \text{ accepts}$$

with k alternations of quantifiers, and where Q is the universal quantifier if k is even, and the existential quantifier otherwise. (Π_k is defined similarly as Σ_k, but starting with a universal quantifier). The ℓ_i's are called *labelling functions*, assigning a label $\ell_i(v) \in \{0,1\}^*$ to every node v, such that, for every node v, $|\ell_i(v)| = O(\log n)$ in n-node networks. Our aim is to analyze the local hierarchy in the general context of distributed network computing [27], where each node has an identity which is unique in the network, every node has the computational power of a Turing machine, and where the acceptance of an instance by an algorithm is defined as the logical conjunction of the individual decisions of the nodes.

1.1 Our Results

We study a hierarchy $(\Sigma_k, \Pi_k)_{k\geq 0}$ of local decision which represents a natural extension of proof-labelling scheme, as well as of locally checkable proof, with succinct certificates (i.e., of size $O(\log n)$ bits). In addition to its conceptual interest, this hierarchy might have some practical impact. Indeed, any level k of the hierarchy can be viewed as a game between a *prover* and a *disprover*, who play in turn by alternating k times. Roughly, on legal instances, the prover aims at assigning distributed certificates responding to any attempt of the disprover to demonstrate that the instance is illegal, and vice-versa on illegal instances. The referee judging the correctness of the collection of certificates produced by the players is a local distributed algorithm. For instance, the extensively studied class Σ_1 includes problems whose solutions are such that their legality can be certified by a prover using distributed certificates. Instead, the class Π_2 includes problems whose solutions are such that their legality can be certified by a prover against any other candidate solution provided by a disprover, both using distributed certificates.

We show that many problems have succinct proofs in the hierarchy. Actually, climbing up the hierarchy enables to reduce drastically the size of the certificates. For instance, we show a quadratic improvement for MST, which requires locally checkable proofs of $\Omega(\log^2 n)$ bits, while MST stands at the second level of our hierarchy. That is, there is a Π_2-protocol for MST using distributed certificates of $O(\log n)$ bits. For graph properties such as nontrivial automorphism, the improvement can even be exponential in term of certificate size, by relaxing the verification from locally checkable proofs with $\Omega(n^2)$ bits proofs to Σ_3 (with $O(\log n)$ bits proofs). More generally, many natural optimization problems are on the second level of our hierarchy. On the other hand, we also show that there are simple (Turing-computable) languages outside the local hierarchy. This latter property illustrates the impact of insisting on compact $O(\log n)$-bits certificates: there are graph properties that cannot be

locally certified via a finite number of interactions between a prover and a disprover using succinct certificates.

In addition, we prove several results regarding the structure of the hierarchy. In particular, we show that if the hierarchy collapses partially at any level, then it collapses all the way down to that level. On the other hand, we prove that the hierarchy does not collapse to the first level (i.e., the first and second levels are distinct). Distributed decision is naturally asymmetric, that is, reversing the individual decision of the algorithm at each node does not correctly reverse the global decision of the algorithm. As a consequence, it is not necessarily the case that co-$\Sigma_k = \Pi_k$, and vice-versa. However, we show that one additional level of quantifiers is always sufficient to reverse a decision (i.e., to decide the complement of a language). Finally, we show that, for every graph property at the intersection of a level-k class and the complement of this class, there is a protocol deciding that property at level k with *unanimous* decision, for both legal and illegal instances.

All our positive results hold in the classical CONGEST model (in which every edge can transmit at most $O(\log n)$ bits at each round), while all our negative results hold in the more liberal LOCAL model (in which there are no constraints on the amount of bits that can be sent through an edge at each round).

All proofs missing from this extended abstract can be found in [7].

1.2 Related Work

Several forms of "local hierarchies" have been investigated in the literature, with the objective of understanding the power of local computation, or for the purpose of designing verification mechanisms for fault-tolerant computing. In particular, as we already mentioned, [28] has investigated the case of *distributed graph automata*, where nodes are anonymous finite automata, and where the decision function is a global interpretation of all the individual outputs of the nodes. In this context, it was proved that the local hierarchy is exactly captured by the MSO formulas on graphs.

The picture is radically different in the framework in which the computing entities are Turing machines with pairwise distinct identities, and where the decision function is the logical conjunction of all the individual boolean outputs. In [13], the authors investigated the local hierarchy in which the certificates must not depend on the identity-assignment to the nodes. Under such *identity-oblivious* certificates, there are distributed languages outside Σ_1. However, all languages are in the probabilistic version of Σ_1, that is, in Σ_1 where the correctness of the verification is only stochastically guaranteed with constant probability. In [11], it is proved that Σ_1 is exactly captured by the set of distributed languages that are closed under lift. (A configuration (G', x') is a t-lift of a configuration (G, x) if there is an input-preserving mapping from $V(G')$ to $V(G)$ which preserves the t-neighbourhood of the nodes in these graphs). Interestingly, in the same framework as [13] but where the decision function is a global interpretation of the all the individual outputs, instead of the logical conjunction of individual boolean outputs, [3, 4] proved that the local hierarchy collapses to Σ_1. Also, in the same framework as [13], but where the certificates may depend on the identity assignment, all distributed languages are in Σ_1 (see [20]).

In [16], the authors proved that, to be placed in the first level Σ_1 of the local hierarchy, there are distributed languages on graphs (e.g., the existence of a nontrivial automorphism) which require to exchange certificates of size $\Omega(n^2)$ bits among neighbours, which is enough to trivially decide any problem. Similarly, [18, 20] has proved that certifying Minimum-weight Spanning Tree (MST) requires to exchange certificates on $\Theta(\log^2 n)$ bits, which can be costly in networks with limited bandwidth, i.e., under the CONGEST model [27]. In [19], it is

proved that the size of the certificates for MST can be decreased to $O(\log n)$ bits, but to the expense of $O(\log n)$ rounds of communication. Recently, [25] has proved that the amount of communication between nodes (but not necessarily the size of the certificates) for verifying general network configurations can be exponentially decreased if using randomization, and [9] analyzed in depth the certificate size for s-t connectivity and acyclicity.

It is also worth mentioning the role of the node identities in distributed decision. For instance, after noticing that the identities are leaking information to the nodes about the size of the network (e.g., at least one node has an ID at least $n-1$ in n-node network), it was recently proved that restricting the algorithms to be identity-oblivious reduces the ability to decide languages locally in Σ_0 (see [10]), while this is not the case for Σ_1 (see [11]). Recently, [12] characterized the "power of the IDs" in local distributed decision. In [5], the authors discussed what can be computed in an anonymous networks, and showed that the answer to this question varies a lot depending on the commitment of the nodes to their first computed output value, i.e., whether it is revocable or not. In the context of local decision, the output is assumed to be irrevocable.

In general, we refer to [32] for a recent survey on local distributed computing, and we refer to [14, 15] for distributed decision in the context of asynchronous crash-prone systems with applications to runtime verification, and to [2] for distributed decision in contexts where nodes have the ability to share non classical resources (e.g., intricate quantum bits).

2 Local Decision

Let $G = (V, E)$ denote an undirected graph, where V is the set of nodes, and E is the set of edges. The subgraph induced by nodes at distance (i.e., number of hops) at most t from a node v is denoted by $B_G(v, t)$. All graphs considered in this paper are assumed to be connected (for non connected graphs, our results apply separately to each connected components). The number of nodes in the graph is denoted by n. In every graph $G = (V, E)$, each node $v \in V$ is assumed to have a name from the set $\{1, \ldots, N\}$, denoted by $\mathrm{id}(v)$, where N is polynomial in n. In other words, all identities are stored on $O(\log n)$ bits. In a fixed graph, all names are supposed to be pairwise distinct.

Distributed languages. A *distributed language* L is a set of pairs (G, x), where G is a graph and x is a function that assigns some local input $x(v)$ to each node v. We assume that all inputs $x(v)$ are polynomial in n, and thus can be stored locally on $O(\log n)$ bits. The following are typical examples of distributed languages:

- 3-COLOURING: (G, x) such that x encodes a proper 3-colouring of G;
- 3-COLOURABILITY: graphs that can be properly 3-coloured;
- NTA: graphs with a nontrivial automorphism;
- PLANARITY: planar graphs.

The *complement* $\bar{L}$ of a distributed language L is defined as the set $\bar{L} = \{(G, x) : (G, x) \notin L\}$. For instance, the complement of 3-COLOURING is NON-3-COLOURING, consisting of all pairs (G, x) such that x is not a proper 3-colouring of G.

Labellings. A *labelling* ℓ is a function $\ell \colon V(G) \to \{0,1\}^*$, assigning a bit string to each node. If, for every graph G and every node $v \in V(G)$, $\ell(v) \in \{0,1\}^k$, we say that the labelling ℓ is of size k. In this paper, we are mostly interested in labellings of logarithmic size in the number of nodes in the input graph.

Local algorithms. We use the standard LOCAL model of distributed computing [27, 24]. In this model each node $v \in V(G)$ is a computational entity that has direct communication links to other nodes, represented by the edges of G. Every node runs the same algorithm. In this paper, all algorithms are deterministic. Nodes communicate with each other in synchronous communication rounds. During each round, every node is allowed to (1) send a message to each of its neighbours, (2) receive a message from each of its neighbours, and (3) perform individual computation. At some point every node has to halt and produce a local output. The number of rounds until all nodes have halted is the *running time* of an algorithm.

A *local* algorithm is a distributed algorithm A for which there exists a constant t such that, for every instance (G, x), the running time of A in (G, x) is at most t. Since the most a node can do in t communication rounds is to gather all the information available in its local neighbourhood $B_G(v, t)$, a local algorithm A can be defined as a (computable) function from all possible labelled local neighbourhoods to some output set. Given an ordered set $\bar{\ell} = (\ell_1, \ell_2, \ldots, \ell_k)$ of labellings, for some $k \geq 0$, and given an instance (G, x), we denote by $A(G, v, x, \bar{\ell})$ the output of v in algorithm A running in G with input x and labelling $\bar{\ell}$.

Local decision. In *distributed decision*, the output of each node v corresponds to its own individual decision. That is, each node either *accepts* or *rejects*. Globally, the instance (G, x) is accepted if and only if every node accepts individually. In other words, the global acceptance is subject to the logical conjunction of all the individual acceptances. For the sake of simplifying the presentation, $A(G, v, x, \bar{\ell}) = 1$ (resp., $A(G, v, x, \bar{\ell}) = 0$) denotes the fact that v accepts (resp., rejects) in an execution of algorithm A on (G, x) labelled with $\bar{\ell}$. We say that A accepts if $A(G, v, x, \bar{\ell}) = 1$ for every node $v \in V(G)$, and rejects otherwise. We will use the shorthand $A(G, x, \bar{\ell}) = 1$ to denote that $\forall v \in V(G), A(G, v, x, \bar{\ell}) = 1$, and $A(G, x, \bar{\ell}) = 0$ to denote that $\exists v \in V(G), A(G, v, x, \bar{\ell}) = 0$.

The first class in the local hierarchy considered in this paper is *local decision*, denoted by LD. A language L is in LD if there exists a local algorithm A, such that for all graphs G, and all possible inputs x on G, we have that $(G, x) \in L \iff A(G, x)$ accepts.

As an example, deciding whether x is a 3-colouring of G is in LD, but deciding whether G is 3-colourable is not. Note that LD does not refer to any labellings. The algorithm A runs solely on graphs G with possible inputs to the nodes.

Example: certifying spanning trees. In a graph G, a *spanning tree* can be encoded as a distributed data-structure x such that, for every $v \in V(G)$, $x(v)$ encodes the identity of one of v's neighbours (its parent in the tree), but one node r for which $x(r) = \perp$ (this node is the root of the tree). Deciding whether x is a spanning tree of G is not in LD. However, a spanning tree can be *certified* locally as follows (see [1, 17]). Given a spanning tree x of G rooted at node r, a *prover* assigns label $\ell(v) = (\mathrm{id}(r), d(v))$ to each node v, where $d(v)$ is the distance of v to the root r in the spanning tree x. Such a label is on $O(\log n)$ bits. The verification algorithm A at node v checks that v agrees on $\mathrm{id}(r)$ with all its neighbours, and that $d(x(v)) = d(v) - 1$. If both tests are passed, then v accepts, otherwise it rejects. It follows that Algorithm A accepts if and only if x is a spanning tree of G. Now we can also accumulate counters, such as the number of nodes in the graph, toward the root. This ability to certify spanning trees and to use them to carry information is a simple but powerful tool that will be used throughout the paper.

3 The Local Hierarchy

We generalize the various classes of distributed decision from previous work into a hierarchy of distributed decision classes, in a way analogous to the polynomial hierarchy (in particular, our class Σ_1^{LD} is equal to[1] the class logLCP introduced by Göös and Suomela [16]).

3.1 Definition

We first define an infinite hierarchy $\{(\Sigma_k^{\mathsf{LD}})_{k\geq 0}, (\Pi_k^{\mathsf{LD}})_{k\geq 0}\}$ of classes. For the sake of simplifying the notation, each of these classes is now abbreviated in Σ_k or Π_k. Informally, each class can be defined by a game between two players, called the *prover* and the *disprover*, who can assign labels to the nodes. The nodes take these labels as additional inputs when running their local algorithm A. Both players are given the language L, the instance (G, x), and the algorithm A. The goal of the prover is to make the nodes accept the instance, whereas the disprover wants it to be rejected. In Σ_k (resp., Π_k), with $k > 0$, the prover (resp., disprover) goes first, and assigns an $O(\log n)$-bit label to each node. Then, the players alternate, assigning $O(\log n)$-bit labels to each node in turn, until k labels $\ell_1, \ell_2, \ldots, \ell_k$ are assigned. A language L is in the corresponding class if there exists a local algorithm A such that, for all instances (G, x), the prover has a winning strategy if and only if $(G, x) \in L$. In other words, the algorithm is such that if $(G, x) \in L$, no matter how the disprover assigns its own labels, the prover can make A accept. Conversely, if $(G, x) \notin L$, then the disprover has a winning strategy and thus it can force A to reject. Such a combination local algorithm A and prover-disprover pair is called a *decision protocol* for L in the corresponding class. Equivalently, we define $\mathsf{LD} = \Sigma_0 = \Pi_0$, and, for $k > 0$, Σ_k is defined as the set of languages L for which there exists $c \geq 0$, and a local algorithm A such that

$$(G, x) \in L \iff \exists \ell_1 \, \forall \ell_2 \ldots \mathrm{Q}\, \ell_k, A(G, x, \ell_1, \ell_2, \ldots, \ell_k) = 1,$$

where Q is the existential (resp., universal) quantifier if k is odd (resp., even), and every label ℓ_i is of size at most $c \log n$. The class Π_k is defined similarly, except that the acceptance condition is: $(G, x) \in L \iff \forall \ell_1 \, \exists \ell_2 \ldots \mathrm{Q}\, \ell_k, A(G, x, \ell_1, \ell_2, \ldots, \ell_k) = 1$.

▶ Remark. For both Σ_k and Π_k, the equivalence should hold *for every identity-assignment* to the nodes with identities in $[1, N]$, where N is a fixed function polynomial in n. Indeed, the membership of an instance (G, x) to a language is independent of the identities given to the nodes. On the other hand, the labels given by the prover and the disprover may well depend on the actual identities of the nodes in the graph where the decision algorithm A is run. This is for instance the case of the protocol for certifying spanning trees described in the previous section, establishing that SPANNING-TREE $\in \Sigma_1$.

3.2 The odd-even collapsing and the Λ_k-hierarchy

Interestingly, the ending universal quantifier in both Σ_{2k} and Π_{2k+1} does not help. The class Π_1 turns out to be just slightly stronger than LD. Specifically, we prove the following result.

▶ **Theorem 1.** *For every $k \geq 1$, $\Sigma_{2k} = \Sigma_{2k-1}$ and $\Pi_{2k+1} = \Pi_{2k}$. Moreover, $\mathsf{LD} \subseteq \Pi_1 \subseteq \mathsf{LD}^{\#\mathrm{node}}$, that is, local decision with access to an oracle providing each node with the number of nodes in the graph.*

[1] If fact, Σ_1^{LD} is equal to LogLCP as defined by Göös and Suomela [16] when one restricts computation to be performed by Turing Machines ([16] makes no assumption on the computational power of the nodes).

Proof. The result follows from the fact that an existential quantification on labels of size $O(\log n)$ bit is sufficient to provide the nodes with the exact size of the graph. This can be certified by accumulating subtree counters along a spanning tree (see Section 2 [16]).

▶ **Claim 1.** *Let $L = \{(G, x) : \text{for every } v \in V(G),\ x(v) = |V(G)|\}$. We have that $L \in \Sigma_1$.*

We show how to use this mechanism in the case of Σ_{2k}, for $k > 0$. Let $L \in \Sigma_{2k}$, and let A be a t-round local algorithm such that:

$$(G, x) \in L \iff \exists \ell_1\, \forall \ell_2\, \ldots \exists \ell_{2k-1}\, \forall \ell_{2k}, A(G, x, \ell_1, \ell_2, \ldots, \ell_{2k}) = 1.$$

Recall that all labellings ℓ_i, $i = 1, \ldots, 2k$, are of size at most $c \log n$ for some $c \geq 0$. We construct an algorithm A' that simulates A for a protocol that does not need the last universal quantifier on ℓ_{2k}. The first labelling ℓ'_1 consists of some correct ℓ_1 for A, with the aforementioned additional label that encodes a spanning tree x' (rooted at an arbitrary node) and the value of the number of nodes in G. Regarding the remaining labellings, for each ℓ_{2i-1} assigned by the disprover, the prover assigns ℓ_{2i} as in the protocol for A, ignoring the bits padded to ℓ_1 for creating ℓ'_1. After the labellings have been assigned, each node v gathers its radius-t neighbourhood. Then, it virtually assigns every possible combination of $(c \log n)$-bit labellings $\ell_{2k}(u)$ to each node $u \in B_G(v, t)$, and simulates A at v to check whether it accepts or rejects with this labelling. If every simulation accepts, then A' accepts at v, else it rejects. Since every node generates all possible ℓ_{2k} labellings in its neighbourhood, we get that

$$(G, x) \in L \iff \exists \ell'_1\, \forall \ell_2\, \ldots \exists \ell_{2k-1}, A'(G, \ell_1, \ell_2, \ldots, \ell_{2k-1}) \text{ accepts},$$

which places L in Σ_{2k-1}. The proof of $\Pi_{2k+1} = \Pi_{2k}$ is similar by using the first existential quantifier (which appears in second position) to certify the number of nodes in the graph. For the case of Π_1, the nodes use the number of nodes provided by the oracle. ◀

A consequence of Theorem 1 is that only of the classes Σ_k for odd k, and Π_k for even k, are worth investigating.

▶ **Definition 2.** We define the classes $(\Lambda_k)_{k \geq 0}$ as follows: $\Lambda_k = \begin{cases} \Sigma_k & \text{if } k \text{ is odd;} \\ \Pi_k & \text{otherwise.} \end{cases}$

In particular, $\Lambda_0 = \Pi_0 = \mathsf{LD}$. By definition, we get $\Lambda_k \subseteq \Lambda_{k+1}$ for every $k \geq 0$, as the distributed algorithm can simply ignore the first label.

▶ **Definition 3.** The local hierarchy is defined as $\mathsf{LH} = \cup_{k \geq 0} \Lambda_k$.

3.3 Complementary classes

We define the complement classes co-Λ_k, for $k \geq 0$, as co-$\Lambda_k = \{L : \bar{L} \in \Lambda_k\}$. Note that, due to the asymmetric nature of distributed decision (unanimous acceptance, but not rejection), simply reversing the individual decision of an algorithm deciding L is generally not appropriate to decide $\bar{L}$. Nevertheless, we show that an additional existential quantifier is sufficient to reverse any decision, implying the following theorem.

▶ **Theorem 4.** *For every $k \geq 0$,* co-$\Lambda_k \subseteq \Lambda_{k+1}$.

Proof. The proof uses a spanning tree certificate to reverse the decision, in a way similar to the proof that the complement of LD is contained in logLCP (i.e., according to our terminology, co-$\Lambda_0 \subseteq \Lambda_1$) due to Göös and Suomela [16]. Let $L \in \Lambda_k$, and let A be a t-round local algorithm deciding $L \in \Lambda_k$ using labels on at most $c \log n$ bits. We construct

an algorithm A' which simulates A, but uses an additional label ℓ_{k+1} to reverse the decisions made by A. Let us assume that k is even (as it will appear clear later, the proof is essentially the same for k odd). We have that

$$(G,x) \in L \iff \forall \ell_1 \, \exists \ell_2 \, \ldots \exists \ell_k, A(G,x,\ell_1,\ell_2,\ldots,\ell_k) = 1,$$

with all labels ℓ_i's of size at most $c \log n$ for some constant $c \geq 0$. In Algorithm A', the prover and the disprover essentially switch their roles. From the above, we have

$$(G,x) \notin L \iff \exists \ell_1 \, \forall \ell_2 \, \ldots \forall \ell_k \, \exists v \in G, A(G,v,x,\ell_1,\ell_2,\ldots,\ell_k) = 0.$$

The prover for A' always follows the disprover for A, and can always pick labellings $\ell_1, \ell_3, \ldots, \ell_{k-1}$ such that there is a rejecting node if and only if $(G,x) \notin L$. In the protocol for A', the prover sets ℓ_{k+1} to be a spanning tree rooted at one such rejecting node v. Every other node $u \neq v$ simply checks that ℓ_{k+1} constitutes a proper encoding of a spanning tree, and rejects if not. If all nodes $u \neq v$ accept, then ℓ_{k+1} is indeed a proper spanning tree, and it only remains to check that v rejects in A. To this end, the node v designated as the root of the spanning tree encoded by ℓ_{k+1} gathers all labellings in its radius-t neighbourhood, and computes $A(G,x,v,\ell_1,\ell_2,\ldots,\ell_k)$. If A rejects at v, we set A' to accept at v, and, otherwise, we set A' to reject at v.

As discussed in Section 2, the spanning tree can be encoded using $O(\log n)$ bits. All labellings $\ell_1, \ell_2, \ldots, \ell_k$ have size at most $c \log n$, therefore all labels of A' are of size at most $c' \log n$ for some $c' \geq c$. The protocol is correct, as a rejecting node exists in A if and only if $(G,x) \notin L$, and A' correctly accepts in this case. If $(G,x) \in L$, then we have that, for every choice the prover can make, the disprover can always choose its labellings so that A accepts. Thus, if the spanning tree ℓ_{k+1} is correct, the root of that tree will indeed detect that it is an accepting node in A, and so reject in A'. ◀

▶ **Corollary 5.** *For every $k \geq 0$, co-$\Lambda_k \subseteq$ co-Λ_{k+1}, and $\Lambda_k \subseteq$ co-Λ_{k+1}.*

Proof. If $L \in$ co-Λ_k, then, by definition, $\bar{L} \in \Lambda_k$, and thus also $\bar{L} \in \Lambda_{k+1}$, which implies that $L \in$ co-Λ_{k+1}. If $L \in \Lambda_k$, then $\bar{L} \in$ co-Λ_k, and thus, by Theorem 4, we get that $\bar{L} \in \Lambda_{k+1}$, which implies that $L \in$ co-Λ_{k+1}. ◀

The following theorem shows that, for every $k \geq 0$, and every language L in $\Lambda_k \cap$ co-Λ_k, there is an algorithm deciding L such that an instance $(G,x) \in L$ is accepted at all nodes, and an $(G,x) \notin L$ is rejected at all nodes.

▶ **Theorem 6.** *Let $k \geq 1$, and let $L \in \Lambda_k \cap$ co-Λ_k. Then there exists a local algorithm A such that, for every instance (G,x), and for every $v \in V(G)$,*

$$(G,x) \in L \iff \begin{cases} \forall \ell_1 \, \exists \ell_2 \, \forall \ell_3 \, \ldots \exists \ell_k, A(G,v,x,\ell_1,\ldots,\ell_k) = 1 & \text{if } k \text{ is even} \\ \exists \ell_1 \, \forall \ell_2 \, \ldots \exists \ell_k, A(G,v,x,\ell_1,\ldots,\ell_k) = 1 & \text{otherwise} \end{cases}$$

In Theorem 9 in the next section, we shall see several example of languages in $\Lambda_1 \cap$ co-Λ_1, in relation with classical optimization problems on graphs. By Theorem 6, all of these languages can be decided unanimously.

3.4 Separation results

From the previous results in this section, we get that the local hierarchy $\mathsf{LH} = \cup_{k \geq 0} \Lambda_k$ has a typical "crossing ladder" as depicted on Figure 1.

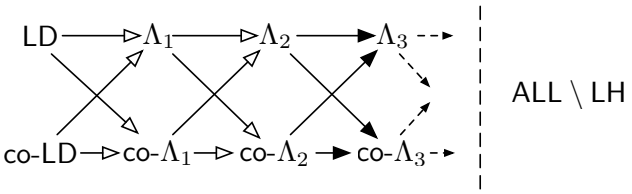


Figure 1 Structure of the local hierarchy. Arrows indicate inclusions, while hollow-headed arrows indicate strict inclusions.

In addition, we can show that some of the inclusions are strict. Indeed, it is known for long that LD is strictly included in Λ_1 (for instance, 2-COLOURABILITY $\in \Lambda_1 \setminus$ LD). Also, $\Lambda_0 \cup$ co-Λ_0 is strictly included in co-Λ_1. Indeed, for instance, NON-3-COLOURABILITY $\in$ co-$\Lambda_1 \setminus(\Lambda_0 \cup$ co-$\Lambda_0)$. Therefore, all inclusions between LD and co-LD and the classes at the first level are strict. Moreover, it is known [16] that NON-3-COLOURABILITY $\notin \Lambda_1$, implying that 3-COLOURABILITY $\notin$ co-Λ_1. On the other hand, both languages are in Λ_2, by application of Theorem 4. As a consequence, both are also in co-Λ_2. Therefore, all inclusions between the classes at the first and second levels are strict.

For $k \geq 2$, separating the classes at the kth level from the classes at the next level appears to be not straightforward. In particular, all classical counting arguments used to separate the three first levels (i.e., levels 0, 1, and 2) fail. On the other hand, we show that if $\Lambda_k = \Lambda_{k+1}$ for some k, then LH collapses to the kth level.

▶ **Theorem 7.** *If there exists $k \geq 0$ such that $\Lambda_k = \Lambda_{k+1}$, then $\Lambda_i = \Lambda_k$ for all $i > k$, that is,* LH *collapses at the kth level.*

Finally, we show that there are languages outside LH. In fact, this result holds, even if we restrict ourselves to languages with inputs 0 or 1 on oriented paths, i.e., with identity-assignment where nodes are given consecutive ID from 1 to n. The result follows from the fact that there are "only" $2^{2^{O(\log n)}}$ different local algorithms for such n-node instances at any fixed level of LH, while there are 2^{2^n} different languages on such instances.

▶ **Theorem 8.** *There exists a Turing-computable language on 0/1-labelled oriented paths that is outside* LH.

4 Positive results

In this section, we precisely identify the position of some relevant problems for distributed computing in the local hierarchy.

Optimization problems. Given an optimization problem π on graphs (e.g., finding a minimum dominating set), one defines two distinct distributed languages: the language OPT$_\pi$ (resp., ADM$_\pi$) is composed of all configurations (G, x) such that x encodes an optimal (resp., admissible) solution for π in graph G. Informally, in the context of optimization problems, the disprover aims at demonstrating that the current solution is not optimal or not admissible. Typically, the disprover does so by exhibiting a better solution or a proof of non-admissibility.

The minimum-weight spanning tree (MST) problem, which is one of the most studied problem in the context of network computing [18, 19, 20], is a typical example of optimization problems that we aim at considering in this section, but many other problems such as maximum independent set, max-cut, etc., are also of our interest. We show that, for any

optimization problem π, if deciding whether a candidate solution for π is admissible is "easy", and if the objective function for π has an additive form, then $\text{OPT}_\pi \in \text{co-}\Lambda_1$, and $\text{OPT}_\pi \in \Lambda_2$.

▶ **Theorem 9.** *Let π be an optimization problem on graphs. If the following two properties are satisfied:* (a) $\text{ADM}_\pi \in \Lambda_1 \cap \text{co-}\Lambda_1$, *and* (b) *the value to the objective function for π is the sum, over all nodes, of an individual value at each node which can be computed locally and encoded on $O(\log n)$ bits, then $\text{OPT}_\pi \in \text{co-}\Lambda_1$.*

Let us give concrete examples of problems satisfying hypotheses (a) and (b). In fact, most classical optimization problems are satisfying these hypotheses, and all the ones typically investigated in the framework of local computing (cf. the survey [32]) do satisfy (a) and (b).

▶ **Corollary 10.** *Let π be one of the following optimization problems: maximum independent set, minimum dominating set, maximum matching, max-cut, or min-cut. Then $\text{OPT}_\pi \in \text{co-}\Lambda_1$.*

The following corollary of Theorem 9 deals with two specific optimization problems, namely travelling salesman and MST. The former illustrates a significant difference between the local hierarchy defined from distributed graph automata in [28], and the one in this paper. Indeed, we show that travelling salesman is at the second level of our hierarchy, while it does not even belong to the graph automata hierarchy (as Hamiltonian cycle is not in MSO). Let TRAVELLING SALESMAN be the distributed language formed of all configurations (G, x) where G is a weighted graph, and x is an Hamiltonian cycle C in G of minimum weight (i.e., at node u, $x(u)$ is the pair of edges incident to u in C). Similarly, let MST be the distributed language formed of all configurations (G, x) where G is a weighted graph, and x is a MST T in G (i.e., at node u, $x(u)$ is the parent of u in T). Note that the case of MST is also particularly interesting. Indeed, MST is known to be in $\text{LCP}(\log^2(n))$ [18], but not in $\Lambda_1 = \text{LCP}(\log(n))$ [19]. Note also that, for MST, it is possible to trade locality for the size of the certificates, as it was established in [19] that one can use logarithmic certificates to certify MST in a logarithmic number of rounds. A consequence of Theorem 9 is the following.

▶ **Corollary 11.** $\text{MST} \in \text{co-}\Lambda_1$ *and* $\text{TRAVELLING SALESMAN} \in \text{co-}\Lambda_1$ *for weighted graphs with weights bounded by a polynomial in n.*

Non-trivial automorphism. The graph automorphism problem is the problem of testing whether a given graph has a nontrivial automorphism (i.e., an automorphism[2] different from the identity). Let NONTRIVIAL AUTOMORPHISM be the distributed language composed of the (connected) graphs that admit such an automorphism. It is known that this language is maximally hard for locally checkable proofs, in the sense that it requires proofs with size $\Omega(n^2)$ bits [16]. Nevertheless, we prove that this language is low in the local hierarchy.

▶ **Theorem 12.** $\text{NONTRIVIAL AUTOMORPHISM} \in \Lambda_3$.

Proof. The first label ℓ_1 at node u is an integer that is supposed to be the identity of the image of u by a nontrivial automorphism. Let us denote by $\phi : V(G) \to V(G)$ the mapping induced by ℓ_1. We are left with proving that deciding whether a given ϕ is a nontrivial automorphism of G is in Λ_2. Thanks to Theorem 4, it is sufficient to prove that this decision can be made in co-Λ_1. Thus let us prove that checking that (G, ϕ) is *not* a nontrivial automorphism is in Λ_1. If ϕ is the identity, then the certificate can just encode this

[2] Recall that $\phi : V(G) \to V(G)$ is an automorphism of G if and only if ϕ is a bijection, and, for every two nodes u and v, we have: $\{u, v\} \in E(G) \iff \{\phi(u), \phi(v)\} \in E(G)$.

information, and each node u checks that $\phi(u)$ is equal to its own ID. So assume now that ϕ is distinct from the identity, but is not an automorphism. To certify this, the prover assigns to each node a set of at most four spanning tree certificates, that "broadcast" to all nodes the identity of at most four nodes witnessing that ϕ is not an automorphism. Specifically, if $\phi(u) = \phi(v)$ with $u \neq v$, then the certificates are for three spanning trees, respectively rooted at u, v, and $\phi(u)$, and if $\{u, v\} \in E(G)$ is mapped to $\{\phi(u), \phi(v)\} \notin E(G)$, or $\{u, v\} \notin E(G)$ is mapped to $\{\phi(u), \phi(v)\} \in E(G)$, then the certificates are for four spanning trees, respectively rooted at $u, v, \phi(u)$, and $\phi(v)$. Checking such certificates can be done locally, and thus checking that (G, ϕ) is *not* a nontrivial automorphism is in Λ_1, and the claim follows. ◀

Problems from the polynomial hierarchy. As the local hierarchy LH is inspired by the polynomial hierarchy, it is natural to ask how their respective levels are connected. In this section, we show that some connections can indeed be established, for central problems in the polynomial hierarchy. For instance, let $k \geq 0$, and let us consider all (connected) graphs $G = (V, E)$ such that there exists $X \subseteq V$, $|X| \geq k$, such that, for every $S \subseteq X$, there is a cycle C in G containing all vertices in S, but none in $X \setminus S$. Such graphs have Cycle-VC-dimension, $\mathrm{VC}_{cycle}(G)$, at least k. Deciding whether, given G and k, we have $\mathrm{VC}_{cycle}(G) \geq k$ is Σ_3^P-complete [30, 31]. Let CYCLE-VC-DIMENSION be the distributed language composed of all configurations (G, k) such that all nodes of G have the same input k, and $\mathrm{VC}_{cycle}(G) \geq k$.

▶ **Theorem 13.** CYCLE-VC-DIMENSION $\in \Lambda_3$.

Proof. The existence of the set X can be certified setting a flag at each node in X, together with a tree T_X spanning X for proving that $|X| \geq k$. Given $S \subseteq X$, the cycle C can be certified in the same way as the Hamiltonian cycle in the proof of Corollary 11. ◀

We also show that the natural graph version QBF-SAT$_k$ of QBF-k-SAT is in Λ_k. Also, as a direct consequence of the fact that any language at level k of the distributed graph automata hierarchy [28] is at level at most $k + 1$ of LH, we get:

▶ **Theorem 14.** *All graph properties expressible in MSO are in* LH.

5 Conclusion

In this paper, we have defined and analyzed a local hierarchy LH of decision generalizing proof-labelling schemes and locally checkable proofs. Using this hierarchy, we have defined interactive local decision protocols enabling to decrease the size of the distributed certificates. We have defined the hierarchy for $O(\log n)$-bit size labels, mostly because this extends the class LogLCP in [16], and because this is consistent with the classical CONGEST model for distributed computation [27]. However, most of our results can be extended to labels on $O(B(n))$ bits, for $B(n)$ larger than $\log n$. In particular, it is worth noticing that the existence of a language L outside LH holds as long as $B = o(n)$.

The main open problem is whether LH has infinitely many levels, or whether it collapses at some level Λ_k. We know that the latter can only happen for $k \geq 2$, and thus it would be quite interesting to know whether $\Lambda_3 \neq \Lambda_2$. In particular, all the typical counting arguments used to separate Λ_2 from Λ_1, or, more generally, to give lower bounds on the label size in proof-labelling schemes or locally checkable proofs appear to be too weak for separating Λ_3 from Λ_2. A separation result for $\Lambda_3 \neq \Lambda_2$ would thus probably provide new tools and concepts for the design of space lower bounds in the framework of distributed computing.

Acknowledgements. We thank Jukka Suomela for pointing out that a counting argument can be used to find languages outside the local hierarchy, and Fabian Reiter for fruitful discussions about distributed graph automata.

References

1 Yehuda Afek, Shay Kutten, and Moti Yung. The local detection paradigm and its application to self-stabilization. *Theor. Comput. Sci.*, 186(1-2):199–229, 1997. doi:10.1016/S0304-3975(96)00286-1.

2 Heger Arfaoui and Pierre Fraigniaud. What can be computed without communications? *SIGACT News*, 45(3):82–104, 2014. doi:10.1145/2670418.2670440.

3 Heger Arfaoui, Pierre Fraigniaud, David Ilcinkas, and Fabien Mathieu. Distributedly testing cycle-freeness. In *Graph-Theoretic Concepts in Computer Science – 40th International Workshop, WG 2014, Nouan-le-Fuzelier, France, June 25-27, 2014. Revised Selected Papers*, pages 15–28, 2014. doi:10.1007/978-3-319-12340-0_2.

4 Heger Arfaoui, Pierre Fraigniaud, and Andrzej Pelc. Local decision and verification with bounded-size outputs. In *Stabilization, Safety, and Security of Distributed Systems – 15th International Symposium, SSS 2013, Osaka, Japan, November 13-16, 2013. Proceedings*, pages 133–147, 2013. doi:10.1007/978-3-319-03089-0_10.

5 Yuval Emek, Jochen Seidel, and Roger Wattenhofer. Computability in anonymous networks: Revocable vs. irrecovable outputs. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 183–195, 2014. doi:10.1007/978-3-662-43951-7_16.

6 Laurent Feuilloley and Pierre Fraigniaud. Randomized local network computing. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 340–349, 2015. doi:10.1145/2755573.2755596.

7 Laurent Feuilloley, Pierre Fraigniaud, and Juho Hirvonen. A hierarchy of local decision. *CoRR*, abs/1602.08925, 2016.

8 Patrik Floréen, Marja Hassinen, Joel Kaasinen, Petteri Kaski, Topi Musto, and Jukka Suomela. Local approximability of max-min and min-max linear programs. *Theory Comput. Syst.*, 49(4):672–697, 2011. doi:10.1007/s00224-010-9303-6.

9 Klaus-Tycho Förster, Thomas Luedi, Jochen Seidel, and Roger Wattenhofer. Local checkability, no strings attached. In *Proceedings of the 17th International Conference on Distributed Computing and Networking, Singapore, January 4-7, 2016*, page 21, 2016. doi:10.1145/2833312.2833315.

10 Pierre Fraigniaud, Mika Göös, Amos Korman, and Jukka Suomela. What can be decided locally without identifiers? In *ACM Symposium on Principles of Distributed Computing, PODC'13, Montreal, QC, Canada, July 22-24, 2013*, pages 157–165, 2013. doi:10.1145/2484239.2484264.

11 Pierre Fraigniaud, Magnús M. Halldórsson, and Amos Korman. On the impact of identifiers on local decision. In *Principles of Distributed Systems, 16th International Conference, OPODIS 2012, Rome, Italy, December 18-20, 2012. Proceedings*, pages 224–238, 2012. doi:10.1007/978-3-642-35476-2_16.

12 Pierre Fraigniaud, Juho Hirvonen, and Jukka Suomela. Node labels in local decision. In *Structural Information and Communication Complexity – 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings*, pages 31–45, 2015. doi:10.1007/978-3-319-25258-2_3.

13 Pierre Fraigniaud, Amos Korman, and David Peleg. Towards a complexity theory for local distributed computing. *J. ACM*, 60(5):35, 2013. doi:10.1145/2499228.

14 Pierre Fraigniaud, Sergio Rajsbaum, and Corentin Travers. Locality and checkability in wait-free computing. *Distributed Computing*, 26(4):223–242, 2013. doi:10.1007/s00446-013-0188-x.

15 Pierre Fraigniaud, Sergio Rajsbaum, and Corentin Travers. On the number of opinions needed for fault-tolerant run-time monitoring in distributed systems. In *Runtime Verification – 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings*, pages 92–107, 2014. doi:10.1007/978-3-319-11164-3_9.

16 Mika Göös and Jukka Suomela. Locally checkable proofs. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 159–168, 2011. doi:10.1145/1993806.1993829.

17 Gene Itkis and Leonid A. Levin. Fast and lean self-stabilizing asynchronous protocols. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 226–239, 1994. doi:10.1109/SFCS.1994.365691.

18 Amos Korman and Shay Kutten. Distributed verification of minimum spanning trees. *Distributed Computing*, 20(4):253–266, 2007.

19 Amos Korman, Shay Kutten, and Toshimitsu Masuzawa. Fast and compact self-stabilizing verification, computation, and fault detection of an MST. *Distributed Computing*, 28(4):253–295, 2015. doi:10.1007/s00446-015-0242-y.

20 Amos Korman, Shay Kutten, and David Peleg. Proof labeling schemes. *Distributed Computing*, 22(4):215–233, 2010. doi:10.1007/s00446-010-0095-3.

21 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 300–309, 2004. doi:10.1145/1011767.1011811.

22 Christoph Lenzen, Yvonne Anne Oswald, and Roger Wattenhofer. What can be approximated locally?: case study: dominating sets in planar graphs. In *SPAA 2008: Proceedings of the 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures, Munich, Germany, June 14-16, 2008*, pages 46–54, 2008. doi:10.1145/1378533.1378540.

23 Christoph Lenzen and Roger Wattenhofer. Leveraging Linial's locality limit. In *Distributed Computing, 22nd International Symposium, DISC 2008, Arcachon, France, September 22-24, 2008. Proceedings*, pages 394–407, 2008. doi:10.1007/978-3-540-87779-0_27.

24 Nathan Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992. doi:10.1137/0221015.

25 Baruch Mor, Pierre Fraigniaud, and Boaz Patt-Shamir. Randomized proof-labeling schemes. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21-23, 2015*, pages 315–324, 2015. doi:10.1145/2767386.2767421.

26 Moni Naor and Larry J. Stockmeyer. What can be computed locally? *SIAM J. Comput.*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

27 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*, volume 5. SIAM, 2000.

28 Fabian Reiter. Distributed graph automata. In *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 192–201, 2015. doi:10.1109/LICS.2015.27.

29 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012. doi:10.1137/11085178X.

30 Marcus Schaefer. Deciding the Vapnik-Cervonenkis dimension in Σ_3^p-complete. *J. Comput. Syst. Sci.*, 58(1):177–182, 1999. doi:10.1006/jcss.1998.1602.

31 Marcus Schaefer and Christopher Umans. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT news*, 33(3):32–49, 2002.

32 Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24, 2013. doi:10.1145/2431211.2431223.

Constraint Satisfaction Problems for Reducts of Homogeneous Graphs

Manuel Bodirsky[*1], Barnaby Martin[†2], Michael Pinsker[‡3], and András Pongrácz[§4]

1 **Institut für Algebra, TU Dresden, Dresden, Germany**
2 **School of Science and Technology, Middlesex University, London, United Kingdom**
3 **Department of Algebra, MFF UK, Praha, Czech Republic**
4 **Department of Algebra and Number Theory, University of Debrecen, Debrecen, Hungary**

Abstract

For $n \geq 3$, let (H_n, E) denote the n-th Henson graph, i.e., the unique countable homogeneous graph with exactly those finite graphs as induced subgraphs that do not embed the complete graph on n vertices. We show that for all structures Γ with domain H_n whose relations are first-order definable in (H_n, E) the constraint satisfaction problem for Γ is either in P or is NP-complete.

We moreover show a similar complexity dichotomy for all structures whose relations are first-order definable in a homogeneous graph whose reflexive closure is an equivalence relation.

Together with earlier results, in particular for the random graph, this completes the complexity classification of constraint satisfaction problems of structures first-order definable in countably infinite homogeneous graphs: all such problems are either in P or NP-complete.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Constraint Satisfaction, Homogeneous Graphs, Computational Complexity, Universal Algebra, Ramsey Theory

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.119

1 Introduction

1.1 Constraint satisfaction problems

A *constraint satisfaction problem* (CSP) is a computational problem in which the input consists of a finite set of variables and a finite set of *constraints*, and where the question is whether there exists a mapping from the variables to some fixed domain such that all the constraints are satisfied. We can thus see the possible constraints as relations on the domain,

* The first author has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 257039). Manuel Bodirsky has also been supported by the DFG-funded project 'Topo-Klon' (Project number 622397).
† The second author was supported by EPSRC grant EP/L005654/1.
‡ The third author has received funding from project P27600 of the Austrian Science Fund (FWF).
§ The fourth author has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 257039). The fourth author was supported by EPSRC grant EP/L005654/1. The fourth author was also supported by the Hungarian Scientific Research Fund (OTKA) grant no. K109185.

and in an instance of the CSP, we are asked to assign domain values to the variables such that certain specified tuples of variables become elements of certain specified relations.

When the domain is finite, and arbitrary constraints are permitted, then the CSP is NP-complete. However, when only constraints from a restricted set of relations on the domain are allowed in the input, there might be a polynomial-time algorithm for the CSP. The set of relations that is allowed to formulate the constraints in the input is often called the *constraint language*. The question which constraint languages give rise to polynomial-time solvable CSPs has been the topic of intensive research over the past years. It has been conjectured by Feder and Vardi [19] that CSPs for constraint languages over finite domains have a complexity dichotomy: they are either in P or NP-complete. This conjecture remains unsettled, although dichotomy is now known on substantial classes (for example when the domain has at most three elements [26, 17] or when the constraint language contains a single binary relation without sources and sinks [21, 1]). Various methods, combinatorial (graph-theoretic), logical, and universal-algebraic have been brought to bear on this classification project, with many remarkable consequences. A conjectured delineation for the dichotomy was given in the algebraic language in [18].

When the domain is infinite, the complexity of the CSP can be outside NP, and even undecidable [10]. But for natural classes of such CSPs there is often the potential for structured classifications, and this has proved to be the case for structures first-order definable over the order $(\mathbb{Q}, <)$ of the rationals [7] or over the integers with successor [8]. Another classification of this type has been obtained for CSPs where the constraint language is first-order definable over the random (Rado) graph [13], making use of structural Ramsey theory. This paper was titled 'Schaefer's theorem for graphs' and it can be seen as lifting the famous classification of Schaefer [26] from Boolean logic to logic over finite graphs, since the random graph is universal for the class of finite graphs.

1.2 Homogeneous graphs and their reducts

The notion of *homogeneity* from model theory plays an important role when applying techniques from finite-domain constraint satisfaction to constraint satisfaction over infinite domains. A relational structure is *homogeneous* if every isomorphism between finite induced substructures can be extended to an automorphism of the entire structure. Homogeneous structures are uniquely (up to isomorphism) given by the class of finite structures that embed into them. The structure $(\mathbb{Q}, <)$ and the random graph are among the most prominent examples of homogeneous structures. The class of structures that are definable over a homogeneous structure with finite relational signature is a very large generalisation of the class of all finite structures, and CSPs for those structures have been studied independently in many different areas of theoretical computer science, e.g. in temporal and spatial reasoning, phylogenetic analysis, computational linguistics, scheduling, graph homomorphisms, and many more; see [4] for references.

While homogeneous relational structures are abundant, there are remarkably few countably infinite homogeneous (undirected, irreflexive) *graphs*; they have been classified by Lachlan and Woodrow [23]. Besides the random graph mentioned earlier, an example of such a graph is the countable homogeneous *universal triangle-free* graph, one of the fundamental structures that appears in most textbooks in model theory. This graph is the up to isomorphism unique countable triangle-free graph (H_3, E) with the property that for every finite independent set $X \subseteq H_3$ and for every finite set $Y \subseteq H_3$ there exists a vertex $x \in H_3 \setminus (X \cup Y)$ such that x is adjacent to every vertex in X and to no vertex in Y.

Further examples of homogeneous graphs are the graphs (H_3, E), (H_4, E), (H_5, E), ..., called the *Henson graphs*, and their complements. Here, (H_n, E) for $n > 3$ is the generalisation

of the graph (H_3, E) above from triangles to cliques of size n. Finally, the list of Lachlan and Woodrow contains only one more family of infinite graphs, namely the graphs (C_n^s, E) whose reflexive closure Eq is an equivalence relation with n classes of equal size s, where $1 \leq n, s \leq \omega$ and either n or s equals ω, as well as their complements. We remark that (C_n^s, Eq) is itself homogeneous and first-order interdefinable with (C_n^s, E), and so we shall sometimes refer to the *homogeneous equivalence relations*.

All countable homogeneous graphs, and even all structures which are first-order definable over homogeneous graphs, are ω-*categorical*, that is, all countable models of their first-order theory are isomorphic. Moreover, all countably infinite homogeneous graphs Γ are *finitely bounded* in the sense that the *age* of Γ, i.e., the class of finite structures that embed into Γ, can be described by finitely many forbidden substructures. Finitely bounded homogeneous structures also share with finite structures the property of having a finite description: up to isomorphism, they are uniquely given by the finite list of forbidden structures that describes their age. Recent work indicates the importance of finite boundedness for complexity classification [2, 9], and it has been conjectured that all structures with a first-order definition in a finitely bounded homogeneous structure enjoy a complexity dichotomy, i.e., their CSP is either in P or NP-complete (cf. [15, 2]). The structures first-order definable in homogeneous graphs therefore provide the most natural class on which to test further the methods developed in [13] specifically for the random graph.

In this article we obtain a complete classification of the computational complexity of CSPs where all constraints have a first-order definition in one of the Henson graphs. We moreover obtain such a classification for CSPs where all constraints have a first-order definition in a countably infinite homogeneous graph whose reflexive closure is an equivalence relation, expanding earlier results for the special cases of one single equivalence class (so-called equality constraints [6]) and infinitely many infinite classes [16]. Together with the above-mentioned result on the random graph, this completes the classification of CSPs for constraints with a first-order definition in any countably infinite homogeneous graph, by Lachlan and Woodrow's classification.

Following an established convention [28, 11], we call a structure with a first-order definition in another structure Δ a *reduct* of Δ. That is, for us a reduct of Δ is as the classical definition of a reduct with the difference that we first allow a first-order expansion of Δ. With this terminology, the present article provides a complexity classification of the CSPs for all reducts of countably infinite homogeneous graphs. In other words, for every such reduct we determine the complexity of deciding its *primitive positive theory*, which consists of all sentences which are existentially quantified conjunctions of atomic formulas and which hold in the reduct. We remark that all reducts of such graphs can be defined by quantifier-free first-order formulas, by homogeneity and ω-categoricity.

For reducts of (H_n, E), the CSPs express computational problems where the task is to decide whether there exists a finite graph without any clique of size n that meets certain constraints. An example of a reduct whose CSP can be solved in polynomial time is $(H_n, \neq, \{(x, y, u, v) : E(x, y) \Rightarrow E(u, v)\})$, where $n \geq 3$ is arbitrary. As it turns out, for every CSP of a reduct of a Henson graph which is solvable in polynomial time, the corresponding reduct over the Rado graph, i.e., the reduct whose relations are defined by the same quantifier-free formulas, is also polynomial-time solvable. On the other hand, the CSP of the reduct $(H_n, \{(x, y, u, v) : E(x, y) \vee E(u, v)\})$ is NP-complete for all $n \geq 3$, but the corresponding reduct over the random graph can be decided in polynomial time.

Similarly, for reducts of the graph (C_n^s, E) whose reflexive closure is an equivalence relation with n classes of size s, where $1 \leq n, s \leq \omega$, the computational problem is to decide whether there exists an equivalence relation with n classes of size s that meets certain constraints.

1.3 Results

Our first result is the complexity classification of the CSPs of all reducts of Henson graphs, showing in particular that a uniform approach to infinitely many "base structures" (namely, the n-th Henson graph for each $n \geq 3$) is possible.

▶ **Theorem 1.** *Let $n \geq 3$, and let Γ be a finite signature reduct of the n-th Henson graph (H_n, E). Then* $\mathrm{CSP}(\Gamma)$ *is either in* P *or* NP*-complete.*

We then obtain a similar complexity dichotomy for reducts of homogeneous equivalence relations, expanding earlier results for special cases [16, 6].

▶ **Theorem 2.** *Let (C_n^s, E) be an infinite graph whose reflexive closure Eq is an equivalence relation with n classes of size s, where $1 \leq n, s \leq \omega$. Then for any finite signature reduct Γ of (C_n^s, E), the problem* $\mathrm{CSP}(\Gamma)$ *is either in* P *or* NP*-complete.*

Together with the classification of countable homogeneous graphs, and the fact that the complexity of the CSPs of the reducts of the Rado graph have been classified [13], this completes the CSP classification of reducts of all countably infinite homogeneous graphs, confirming further instances of the open conjecture that CSPs of reducts of finitely bounded homogeneous structures are either in P or NP-complete [15, 2].

▶ **Corollary 3.** *Let Γ be a finite signature reduct of a countably infinite homogeneous graph. Then* $\mathrm{CSP}(\Gamma)$ *is either in* P *or* NP*-complete.*

1.4 The strategy

The method we employ follows to a large extent the method invented in [13] for the corresponding classification problem where the 'base structure' is the random graph. The key component of this method is the usage of Ramsey theory (in our case, a result of Nešetřil and Rödl [24]) and the concept of *canonical functions* introduced in [12]. There are, however, some interesting differences and novelties that appear in the present proof, as we now shortly outline.

1.4.1 Henson graphs

When studying the proofs in [13], one might get the impression that the complexity of the method grows with the model-theoretic complexity of the base structure, and that for the random graph we have really reached the limits of bearableness for applying the Ramsey method.

However, quite surprisingly, when we step from the random graph to the graphs (H_n, E), which are in a sense more complicated structures from a model-theoretic point of view[1], the classification and its proof become easier again. It is one of the contributions of the present article to explain the reasons behind this effect. Essentially, certain *behaviours* of canonical functions existing on the random graph can not be realised in (H_n, E). For example the canonical polymorphisms of behaviour "max" (cf. preliminaries) play no role for the present classification, but account over the random graph for the tractability of, inter alia, the 4-ary relation defined by the formula $E(x, y) \vee E(u, v)$.

[1] For example, the random graph has a *simple* theory [27], whereas the Henson graphs are the most basic examples of structures whose theory is *not* simple.

Interestingly, we are able to reuse results about canonical functions over the random graph, since the calculus for composing behaviours of canonical functions is the same for any other structure with the same type space, and in particular the Henson graphs. Via this meta-argument we can, on numerous occasions, make statements about canonical functions over the Henson graphs which were proven earlier for the Rado graph, ignoring completely the actual underlying structure; even more comfortably, we can *a posteriori* rule out some possibilities in those statements because of the K_n-freeness of the Henson graphs. Examples of this phenomenon appear in Lemmas 14 and 15.

On the other hand, along with these simplifications, there are also new additional difficulties that appear when investigating reducts of (H_n, E) and that were not present in the classification of reducts of the random graph, which basically stem from the lower degree of symmetry of (H_n, E) compared to the Rado graph. For example, in expansions of Henson graphs by finitely many constants, not all orbits induce copies of Henson graphs; the fact that the analogous statement does hold for the Rado graph was used extensively in [13].

1.4.2 Equivalence relations

Similarly to the situation for the equivalence relation with infinitely many infinite classes studied in [16], there are two interesting sources of NP-hardness for the reducts Γ of other homogeneous equivalence relations: namely, if the equivalence relation is invariant under the polymorphisms of Γ, then the structure obtained from Γ by factoring by the equivalence relation might have a NP-hard CSP, implying NP-hardness for the CSP of Γ itself; or, roughly, for a fixed equivalence class the restriction of Γ to that class might have a NP-hard CSP, again implying NP-hardness of the CSP of Γ (assuming that Γ is a *model-complete core*, see Sections 3 and 6). But whereas for the equivalence relation with infinitely many infinite classes both the factor structure and the restriction to a class are again infinite structures, for the other homogeneous equivalence relations one of the two is a finite structure, obliging us to combine results about CSPs of finite structures with those of infinite structures. As it turns out, the two-element case is, not surprisingly, different from the other finite cases and, quite surprisingly, significantly more involved than the other cases.

2 Preliminaries

The following lemma has been first stated in [22] for finite domain structures Γ only, but the proof there also works for arbitrary infinite structures.

▶ **Lemma 4.** *Let $\Gamma = (D, R_1, \ldots, R_\ell)$ be a relational structure, and let R be a relation that has a primitive positive definition in Γ. Then* $\mathrm{CSP}(\Gamma)$ *and* $\mathrm{CSP}(D, R, R_1, \ldots, R_\ell)$ *are polynomial-time equivalent.*

▶ **Theorem 5** (from [10])**.** *Let Γ be a countable ω-categorical structure. Then the relations preserved by the polymorphisms of Γ are precisely those having a primitive positive definition in Γ.*

These facts make it possible to apply a universal algebraic approach, and classify the complexity of reducts of an ω-categorical structure through understanding the polymorphism clones of these reducts. In fact, we can state our results in terms of the polymorphism clones, see Theorems 22 and 38. Roughly speaking, we will conclude that if Γ is a reduct of a homogeneous graph with a finite relational language, then $\mathrm{CSP}(\Gamma)$ is NP-complete iff for some finite tuple c in Γ, the clone $\mathrm{Pol}(\Gamma, c)$ maps to the clone of projections via a continuous homomorphism.

If such a polymorphism does not exist, it indicates the existence of certain kind of functions in $\mathrm{Pol}(\Gamma)$, which satisfy an equation that prevents them from being mapped to projections with a homomorphism. The idea is to find patterns in the behaviour of these interesting functions, and show that they must generate one out of a given finite number of well-behaved functions; those appearing in Theorems 22 and 38, and whose incidence in $\mathrm{Pol}(\Gamma)$ automatically make $\mathrm{CSP}(\Gamma)$ be solvable in polynomial time. In order to find these well-behaved functions in $\mathrm{Pol}(\Gamma)$, we apply the method mentioned in Section 1.4 by using canonical functions.

▶ **Definition 6.** Let Δ be a structure. The *type* $\mathrm{tp}(a)$ of an n-tuple a in Δ is the set of first-order formulas with free variables $x_1, \ldots, x_n$ that hold for a in Δ. For structures $\Delta_1, \ldots, \Delta_k$ and tuples $a^1, \ldots, a^n \in \Delta_1 \times \cdots \times \Delta_k$, the type $\mathrm{tp}(a^1, \ldots, a^n)$ of $(a^1, \ldots, a^n)$ in $\Delta_1 \times \cdots \times \Delta_k$ is the k-tuple containing the types of $(a_i^1, \ldots, a_i^n)$ in Δ_i for each $1 \leq i \leq k$.

It is well-known that in homogeneous structures such as (H_n, E) and (C_n^k, E), two n-tuples have the same type if and only if they are in the same orbit of the automorphism group.

▶ **Definition 7.** Let $\Delta_1, \ldots, \Delta_k$ and Λ be structures. A *behaviour* B between $\Delta_1, \ldots, \Delta_k$ and Λ is a partial function from the types over $\Delta_1, \ldots, \Delta_k$ to the types over Λ. Pairs (s, t) with $B(s) = t$ are also called *type conditions*. We say that a function $f\colon \Delta_1 \times \cdots \times \Delta_k \to \Lambda$ *satisfies the behaviour* B if whenever $B(s) = t$ and $(a^1, \ldots, a^n)$ has type s in $\Delta_1, \ldots, \Delta_k$, then the n-tuple $(f(a_1^1, \ldots, a_k^1), \ldots, f(a_1^n, \ldots, a_k^n))$ has type t in Λ. A function $f\colon \Delta_1 \times \cdots \times \Delta_k \to \Lambda$ is *canonical* if it satisfies a behaviour which is a total function from the types over $\Delta_1, \ldots, \Delta_k$ to the types over Λ.

To provide immediate examples for these notions, we now define some behaviours that will appear in our proof as well as in the precise CSP classification. For m-ary relations $R_1, \ldots, R_k$ over a set D, we will in the following write $R_1 \cdots R_k$ for the m-ary relation on D^k that holds between k-tuples $x^1, \ldots, x^m \in D^k$ iff $R_i(x_i^1, \ldots, x_i^m)$ holds for all $1 \leq i \leq k$.

▶ **Definition 8.** Given a homogeneous graph G we say that a binary injective operation $f\colon G^2 \to G$ is

- *balanced in the first argument* if for all $u, v \in G^2$ we have that $\mathrm{E}{=}(u, v)$ implies $E(f(u), f(v))$ and $\mathrm{N}{=}(u, v)$ implies $N(f(u), f(v))$;
- *E-dominated (N-dominated) in the first argument* if for all $u, v \in G^2$ with $\neq{=}(u, v)$ we have that $E(f(u), f(v))$ ($N(f(u), f(v))$);
- *balanced/E-dominated/N-dominated in the second argument* if $(x, y) \mapsto f(y, x)$ is balanced/E-dominated/N-dominated in the first argument;
- *balanced/E-dominated/N-dominated* if f is balanced/E-dominated/N-dominated in both arguments, and *unbalanced* if f is not balanced;
- *of behaviour p_1* if for all $u, v \in G^2$ with $\neq{\neq}(u, v)$ we have $E(f(u), f(v))$ iff $E(u_1, v_1)$;
- *of behaviour p_2* if $(x, y) \mapsto f(y, x)$ is of behaviour p_1;
- *of behaviour projection* if it is of behaviour p_1 or p_2;
- *of behaviour* min if for all $u, v \in G^2$ with $\neq{\neq}(u, v)$ we have $E(f(u), f(v))$ iff $\mathrm{EE}(u, v)$.

A ternary canonical injection $f\colon G^3 \to G$ is

- *hyperplanely of behaviour projection* iff the functions $(u, v) \mapsto f(c, u, v)$, $(u, v) \mapsto f(u, c, v)$, and $(u, v) \mapsto f(u, v, c)$ are of behaviour projection for all $c \in G$. Similarly other hyperplane behaviours, such as hyperplanely E-dominated, are defined.
- of behaviour *minority* if for all $u, v \in G^3$ with $\neq{\neq}{\neq}(u, v)$ we have $E(f(u), f(v))$ if and only if $\mathrm{EEE}(u, v)$, $\mathrm{NNE}(u, v)$, $\mathrm{NEN}(u, v)$, or $\mathrm{ENN}(u, v)$.

2.1 Overview

This article is organised as follows. Basic notions and definitions, as well as the fundamental facts of the method we are going to use, are deferred for reasons of space to the appendix. Our notation and definitions may also be found in [13] unless they were already represented in the introduction.

Sections 3 to 5 deal with the Henson graphs: Section 3 is complexity-free and investigates the structure of reducts of Henson graphs via polymorphisms and Ramsey theory. In Section 4, we provide hardness results for different classes of reducts. In Section 5 we rephrase Theorem 1, and we discuss the complexity classification in more detail, formulating in particular a tractability criterion for CSPs of reducts of Henson graphs.

Section 6 investigates the structure of reducts of homogeneous equivalence relations via polymorphisms and Ramsey theory and describes the polynomial-time cases.

3 Polymorphisms over Henson graphs

We investigate polymorphisms of reducts of (H_n, E). We start with unary polymorphisms in Section 3.1, obtaining that we can assume that the relations E and N are pp-definable in our reducts. We then turn to binary polymorphisms in Section 3.2, obtaining Proposition 16 telling us that we may further assume the existence of a binary injective polymorphism. Building on the results of those sections, we show in Section 3.3 via an analysis of ternary polymorphisms that for any reduct which pp-defines the relations E and N, either the polymorphisms preserve a certain relation H, or there is a polymorphism of behaviour min (Proposition 18).

3.1 The unary case: model-complete cores

A countable ω-categorical structure Δ is called a *model-complete core* if $\mathrm{Aut}(\Delta)$ is dense in $\mathrm{End}(\Delta)$, or equivalently, every endomorphism of Δ is an elementary self-embedding, i.e., preserves all first-order formulas over Δ. Every countable ω-categorical structure Γ is *homomorphically equivalent* to an up to isomorphism unique ω-categorical model-complete core Δ, that is, there exists homomorphisms from Γ into Δ and vice-versa [3]. Since the CSPs of homomorphically equivalent structures are equal, it has proven fruitful in classification projects to always work with model-complete cores. The following proposition essentially calculates the model-complete cores of the reducts of Henson graphs.

▶ **Proposition 9.** *Let Γ be a reduct of (H_n, E). Then either* $\mathrm{End}(\Gamma)$ *contains a function whose image induces an independent set, or* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(\Gamma)} = \overline{\mathrm{Aut}(H_n, E)}$.

In the first case of Proposition 9, the model-complete core of the reduct is in fact a reduct of equality. Since the CSPs of reducts of equality have been classified [6], we do not have to consider any further reducts with an endomorphism whose image induces an independent set.

▶ **Lemma 10.** *Let Γ be a reduct of (H_n, E), and assume that* $\mathrm{End}(\Gamma)$ *contains a function whose image is an independent set. Then Γ is homomorphically equivalent to a reduct of $(H_n, =)$.*

In the second case of Proposition 9, it turns out that all polymorphisms preserve the relations E, N, and $\neq$, by the following lemma and Theorem 5.

▶ **Lemma 11.** *Let Γ be such that* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(H_n, E)}$. *Then E, N, and $\neq$ have primitive positive definitions in Γ.*

Before moving on to binary polymorphisms, we observe the following corollary of Proposition 9, first mentioned in [28].

▶ **Corollary 12.** *For every $n \geq 3$, the permutation group $\mathrm{Aut}(H_n, E)$ is a maximal closed subgroup of the full symmetric group on H_n, i.e., every closed subgroup of the full symmetric group containing $\mathrm{Aut}(H_n, E)$ either equals $\mathrm{Aut}(H_n, E)$ or the full symmetric group.*

3.2 Binary polymorphisms

We investigate binary functions preserving E, N, and $\neq$. A finitary operation $f(x_1, \ldots, x_n)$ on a set is *essential* if it does not depend on only one of its arguments x_i.

▶ **Lemma 13.** *Every essential function $f\colon H_n^k \to H_n$ that preserves E, N, and $\neq$ generates a binary injection.*

▶ **Lemma 14.** *Let $f\colon H_n^2 \to H_n$ be a function of behaviour* min *that preserves E and N. Then f generates a binary function of behaviour* min *that is N-dominated.*

By Proposition 9, Lemma 11 and Lemma 13, we may assume that $\mathrm{Pol}(\Gamma)$ contains a binary injection f, as otherwise the complexity of $\mathrm{CSP}(\Gamma)$ is known: see the explanation in the end of this subsection. After an analysis of the possible behaviours of f, we can make further assumptions on the binary injection in $\mathrm{Pol}(\Gamma)$.

▶ **Lemma 15.** *Let $f\colon H_n^k \to H_n$ be an essential function that preserves E, N, and $\neq$. Then f generates one of the following binary canonical injections: of behaviour* min *and N-dominated; or of behaviour p_1, balanced in the first, and N-dominated in the second argument.*

We conclude this section by summarising the results we have so far.

▶ **Proposition 16.** *Let Γ be a reduct of (H_n, E), where $n \geq 3$. Then either*
(1) *Γ is homomorphically equivalent to a reduct of $(H_n, =)$, or*
(2) *Γ pp-defines E, N, and $\neq$.*
In the latter case we have that either
(2a) *every function in* $\mathrm{Pol}(\Gamma)$ *is essentially unary, or*
(2b) $\mathrm{Pol}(\Gamma)$ *contains one of the two binary canonical injections of Lemma 15.*

Note that if item (1) holds then $\mathrm{CSP}(\Gamma)$ is either in P or NP-complete [6], and if item (2a) holds then $\mathrm{CSP}(\Gamma)$ is NP-complete (Theorem 10 in [5]). In case (2b), when $\mathrm{Pol}(\Gamma)$ contains a binary canonical injection of behaviour min which is N-dominated then $\mathrm{CSP}(\Gamma)$ is in P, as we will discuss later. It thus remains to further consider the second case of Lemma 15, which we do in the next subsection.

3.3 The relation H

We investigate Case (2b) of Proposition 16. The following relation characterises the NP-complete cases in this situation.

▶ **Definition 17.** We define a 6-ary relation $H(x_1, y_1, x_2, y_2, x_3, y_3)$ on H_n by

$$\begin{aligned}
&\bigwedge_{i,j\in\{1,2,3\},i\neq j,u\in\{x_i,y_i\},v\in\{x_j,y_j\}} N(u,v) \\
&\wedge \big((E(x_1,y_1) \wedge N(x_2,y_2) \wedge N(x_3,y_3)) \\
&\quad \vee\ (N(x_1,y_1) \wedge E(x_2,y_2) \wedge N(x_3,y_3)) \\
&\quad \vee\ (N(x_1,y_1) \wedge N(x_2,y_2) \wedge E(x_3,y_3))\big)\ .
\end{aligned}$$

The importance of the relation H is reflected in the following proposition, which states that if Γ is a reduct of (H_n, E) with E and N primitive positive definable in Γ, then either H has a primitive positive definition in Γ, in which case CSP(Γ) is NP-complete, or Pol(Γ) has a certain canonical polymorphism which will imply tractability of the CSP. NP-completeness and tractability for those cases will be discussed in Sections 4 and 5.

▶ **Proposition 18.** *Let Γ be a reduct of (H_n, E) with E and N primitive positive definable in Γ. Then at least one of the following holds:*
(a) *There is a primitive positive definition of H in Γ.*
(b) Pol(Γ) *contains a canonical binary injection of behaviour* min.

4 CSPs over Henson graphs

We now explain why any reduct of (H_n, E) which has H among its relations, and hence by Lemma 4 every reduct which pp-defines H, has an NP-hard CSP. While it would be possible to show NP-hardness of CSP(H_n, H) directly by reduction of, say, the NP-hard problem positive 1-in-3-SAT, we will use results from [14], and in fact a recent strengthening thereof from [2], to prove hardness more elegantly via a structural property of Pol(H_n, H).

▶ **Definition 19.** Let Γ be a structure. A *projective clone homomorphism* of Γ is a mapping from Pol(Γ) onto its projections which: preserves arities; fixes each projection; and preserves composition.

A *projective strong h1 clone homomorphism* of Γ is a mapping as above, where the third condition is weakened to preservation of composition with projections.

▶ **Theorem 20** (from [2])**.** *Let Γ be a countable ω-categorical structure in a finite relational language which has a uniformly continuous strong h1 clone homomorphism. Then* CSP(Γ) *is NP-hard.*

▶ **Proposition 21.** *The structure (H_n, H) has a uniformly continuous strong h1 clone homomorphism. Consequently,* CSP(H_n, H) *is NP-hard.*

5 Summary for the Henson graphs

We can restate Theorem 1 in a more detailed fashion as follows.

▶ **Theorem 22.** *Let Γ be a reduct of a Henson graph (H_n, E). Then one of the following holds.*
(1) *Γ has an endomorphism whose image induces an independent set, and is homomorphically equivalent to a reduct of $(H_n, =)$.*
(2) Pol(Γ) *has a uniformly continuous projective clone homomorphism.*
(3) Pol(Γ) *contains a binary canonical injection which is of behaviour* min *and N-dominated.*
Items (2) *and* (3) *cannot simultaneously hold, and when Γ has a finite relational signature, then* (2) *implies NP-completeness and* (3) *implies tractability of its CSP.*

The first statement follows directly from the proof of Theorem 1, with the additional observation that the strong h1 clone homomorphism defined in Proposition 21 is in fact a clone homomorphism. When (3) holds for a reduct, then (2) cannot hold, because (3) implies the existence of $f(x, y) \in$ Pol(Γ) and $\alpha \in \overline{\mathrm{Aut}(\Gamma)}$ such that $f(x, y) = \alpha f(y, x)$ holds, and equation impossible to satisfy by projections. In fact, by further analysing case (1), one can easily show that it also implies either (2) or (3), so that we have the following.

▶ **Corollary 23.** *For every reduct* Γ *of a Henson graph* (H_n, E)*, precisely one of the following holds:* $\mathrm{Pol}(\Gamma)$ *has a uniformly continuous projective clone homomorphism; or* $\mathrm{Pol}(\Gamma)$ *contains* $f(x,y) \in \mathrm{Pol}(\Gamma)$ *and* $\alpha \in \overline{\mathrm{Aut}(\Gamma)}$ *such that* $f(x,y) = \alpha f(y,x)$*.*

When Γ *has a finite relational signature, then the first case possesses NP-completeness and the second case the tractability of its CSP.*

6 Reducts of homogeneous equivalence relations

We now investigate polymorphisms of reducts of the graphs (C_n^s, E), for $2 \leq n, s \leq \omega$, with precisely one of n, s equal to ω. Recall from the preliminaries that we write Eq for the reflexive closure of E.

Similarly to the case of the Henson graphs, we start with unary polymorphisms in Section 6.1, reducing the problem to model-complete cores.

We then turn to higher-arity polymorphisms; here, the organisation somewhat differs from the case of the Henson graphs. The role of the NP-hard relation H from the Henson graphs is now taken by the two sources of NP-hardness mentioned in the introduction: the first source being that factoring by the equivalence relation Eq yields a structure with an NP-hard problem, and the second source being that restriction to some equivalence class yields a structure with an NP-hard problem. In Section 6.2, we show that in fact, one of the two sources always applies for model-complete cores when $2 < n < \omega$ or $2 < s < \omega$. Consequently, only the higher-arity polymorphisms of the reducts of (C_2^ω, E) and (C_ω^2, E) require deeper investigation using Ramsey theory; this will be dealt with in Sections 6.3 and 6.4, respectively.

6.1 The unary case: model-complete cores

▶ **Proposition 24.** *Let* Γ *be a reduct of* (C_n^s, E)*, where* $1 \leq n, s \leq \omega$*, and at least one of* n, s *equals* ω*. Then* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(\Gamma)} = \overline{\mathrm{Aut}(C_n^s, E)}$*, or* $\mathrm{End}(\Gamma)$ *contains an endomorphism onto a clique or an independent set.*

In the following sections, we investigate essential polymorphisms of reducts Γ of (C_n^s, E) which are model-complete cores, i.e., $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(C_n^s, E)}$. The following proposition implies that in that situation, the equivalence relation Eq is invariant under $\mathrm{Pol}(\Gamma)$.

▶ **Proposition 25.** *Let* Γ *be a reduct of* (C_n^s, E)*, where* $1 \leq n, s \leq \omega$*. If* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(C_n^s, E)}$*, then* E*,* N *and* Eq *are preserved by the polymorphisms of* Γ*.*

Therefore, in the above situation Eq is an equivalence relation which is invariant under $\mathrm{Pol}(\Gamma)$, and so $\mathrm{Pol}(\Gamma)$ acts naturally on the equivalence classes of Eq. Moreover, if we fix any $c \in C_n^s$ and expand the structure Γ by the constant c, then the equivalence class C of c has a primitive positive definition in that expansion (Γ, c), since Eq and c do. Hence, C is invariant under $\mathrm{Pol}(\Gamma, c)$, and so $\mathrm{Pol}(\Gamma, c)$ acts naturally on C via restriction. In the following sections, we analyse these actions.

6.2 The case $2 < n < \omega$ or $2 < s < \omega$

It turns out that in these cases, one of the sources of hardness always applies. We will use the following fact about function clones on a finite domain.

▶ **Proposition 26** (from [20])**.** *Every function clone on a finite domain of at least three elements which contains all permutations as well as an essential function contains a unary constant function.*

▶ **Proposition 27.** *Let* Γ *be a reduct of* (C_n^ω, E)*, where* $2 < n < \omega$*, such that* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(C_n^\omega, E)}$*. Then the action of* $\mathrm{Pol}(\Gamma)$ *on the equivalence classes of Eq has no essential and no constant operation.*

▶ **Proposition 28.** *Let* Γ *be a reduct of* (C_ω^s, E)*, where* $2 < s < \omega$*, such that* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(C_\omega^s, E)}$*. Then for any* $c \in C_\omega^s$*, the action of* $\mathrm{Pol}(\Gamma, c)$ *on the equivalence class of* c *has no essential and no constant operation.*

6.3 The case of two infinite classes: $n = 2$ and $s = \omega$

The following proposition states that either one of the two sources of hardness applies, or $\mathrm{Pol}(\Gamma)$ contains a ternary canonical function with a certain behaviour.

▶ **Proposition 29.** *Let* Γ *be a reduct of* (C_2^ω, E) *such that* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(C_2^\omega, E)}$*. Then one of the following holds:*

- *the action of* $\mathrm{Pol}(\Gamma)$ *on the classes of Eq has no essential function;*
- *the action of* $\mathrm{Pol}(\Gamma, c)$ *on the equivalence class of* c *has no essential function, for some* $c \in C_2^\omega$*;*
- $\mathrm{Pol}(\Gamma)$ *contains a canonical ternary injection of behaviour minority which is hyperplanely of behaviour E-dominated projection.*

To prove the proposition, we need to recall a special case of Post's classical result about function clones acting on a two-element set, as well as a result on function clones on a countable set which contain all permutations. Comparing this statement with Proposition 26 sheds light on why the case of this section is more involved than the cases of the preceding section.

▶ **Proposition 30** (Post [25])**.** *Every function clone with domain* $\{0, 1\}$ *containing both permutations of* $\{0, 1\}$ *as well as an essential function contains a unary constant operation or the ternary addition modulo 2.*

▶ **Proposition 31** (from [6])**.** *Every closed function clone on a countably infinite set which contains all permutations as well as an essential operation contains a binary injection.*

▶ **Proposition 32.** *Let* Γ *be preserved by a ternary injection* h *of behaviour minority which is hyperplanely an E-dominated projection. Then* $\mathrm{CSP}(\Gamma)$ *can be solved in polynomial time.*

6.4 The case of infinitely many classes of size two: $n = \omega$ and $s = 2$

As in the preceding section, we show that either one of the two sources of hardness applies, or $\mathrm{Pol}(\Gamma)$ contains a ternary canonical function of a certain behaviour.

▶ **Proposition 33.** *Let* Γ *be a reduct of* (C_ω^2, E) *such that* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(C_\omega^2, E)}$*. Then one of the following holds:*

- *the action of* $\mathrm{Pol}(\Gamma)$ *on the classes of Eq has no essential function;*
- *the action of* $\mathrm{Pol}(\Gamma, c)$ *on the equivalence class of* c *has no essential function, for some* $c \in C_\omega^2$*;*
- $\mathrm{Pol}(\Gamma)$ *contains a ternary canonical function* h *with* $h(N, \cdot, \cdot) = h(\cdot, N, \cdot) = h(\cdot, \cdot, N) = N$ *and which behaves like a minority on* $\{E, =\}$*.*

To prove the proposition, we are again going to make use of Propositions 30 and 31, and the following lemma.

▶ **Lemma 34.** *Let* Γ *be a reduct of* (C^2_ω, E) *such that* $\mathrm{End}(\Gamma) = \overline{\mathrm{Aut}(C^2_\omega, E)}$. *If* $\mathrm{Pol}(\Gamma)$ *contains a ternary function which behaves like* $x + y + z$ *modulo 2 on some equivalence class, then it contains a ternary function which behaves like* $x + y + z$ *modulo 2 on all equivalence classes.*

Let Γ be a reduct of (C^2_ω, Eq) where Eq is an equivalence relation with infinitely many classes of size two such that $\mathrm{Pol}(\Gamma)$ contains a ternary canonical function h as in item 3 of Proposition 33.

▶ **Proposition 35.** *A relation with a first-order definition in* (C^2_ω, Eq) *is preserved by* h *if and only if it can be defined by a conjunction of formulas of the form*

$$N(x_1, y_1) \vee \cdots \vee N(x_k, y_k) \vee Eq(z_1, z_2) \tag{1}$$

for $k \geq 0$, *or of the form*

$$N(x_1, y_1) \vee \cdots \vee N(x_k, y_k) \vee (|\{i \in S : x_i \neq y_i\}| \equiv_2 p) \tag{2}$$

where $p \in \{0, 1\}$ *and* $S \subseteq \{1, \ldots, k\}$.

▶ **Proposition 36.** *There is a polynomial-time algorithm that decides whether a given set* Φ *of formulas as in the statement of Proposition 35 is satisfiable.*

▶ **Corollary 37.** *Let* Γ *be a reduct of* (C^2_ω, Eq) *with finite signature and such that* $\mathrm{Pol}(\Gamma)$ *contains the operation* h. *Then* $\mathrm{CSP}(\Gamma)$ *is in P.*

We close the section with a more detailed variant of Theorem 2.

▶ **Theorem 38.** *Let* (C^s_n, E) *be an infinite graph whose reflexive closure* Eq *is an equivalence relation with* n *classes of size* s, *where* $1 \leq n, s \leq \omega$. *Let* Γ *be a reduct of* (C^s_n, E). *Then one of the following holds.*

(1) Γ *has an endomorphism whose image induces a clique or an independent set, and is homomorphically equivalent to a reduct of* $(C^s_n, =)$.

(2) Γ *is a model complete core and* $\mathrm{Pol}(\Gamma, c)$ *has a uniformly continuous projective clone homomorphism for some* $c \in (C^s_n, E)$.

(3) $n = 2, s = \omega$, Γ *is a model complete core, and* $\mathrm{Pol}(\Gamma)$ *contains a canonical ternary injection of behaviour minority which is hyperplanely of behaviour E-dominated projection.*

(4) $n = \omega, s = 2$, Γ *is a model complete core, and* $\mathrm{Pol}(\Gamma)$ *contains a ternary canonical function* h *with* $h(N, \cdot, \cdot) = h(\cdot, N, \cdot) = h(\cdot, \cdot, N) = N$ *and which behaves like a minority on* $\{E, =\}$.

Neither items (2) *and* (3), *nor items* (2) *and* (4) *can simultaneously hold, and when* Γ *has a finite relational signature, then* (2) *implies NP-completeness and both* (3) *and* (4) *imply tractability of its CSP.*

7 Outlook

We have classified the computational complexity of CSPs for reducts of the infinite homogeneous graphs. Our proof shows that the scope of the classification method from [13] is much larger than one might expect at first sight. The general research goal here is to identify larger and larger classes of infinite-domain CSPs where systematic complexity classification is possible; a general dichotomy conjecture is open for CSPs of reducts of finitely bounded homogeneous structures [15, 2]. The next step in this direction might be to show a general complexity dichotomy for reducts of homogeneous structures whose age is finitely bounded and has the *free amalgamation property* (the Henson graphs provide natural examples for such structures). The present paper indicates that this problem might be within reach.

References

1 Libor Barto, Marcin Kozik, and Todd Niven. The CSP dichotomy holds for digraphs with no sources and no sinks (a positive answer to a conjecture of Bang-Jensen and Hell). *SIAM Journal on Computing*, 38(5), 2009.

2 Libor Barto, Jakub Opršal, and Michael Pinsker. The wonderland of reflections. Preprint arXiv:1510.04521, 2015.

3 Manuel Bodirsky. Cores of countably categorical structures. *Logical Methods in Computer Science*, 3(1):1–16, 2007.

4 Manuel Bodirsky. Complexity classification in infinite-domain constraint satisfaction. Mémoire d'habilitation à diriger des recherches, Université Diderot – Paris 7. Available at arXiv:1201.0856, 2012.

5 Manuel Bodirsky, Hubie Chen, Jan Kára, and Timo von Oertzen. Maximal infinite-valued constraint languages. *Theoretical Computer Science (TCS)*, 410:1684–1693, 2009. A preliminary version appeared at ICALP'07.

6 Manuel Bodirsky and Jan Kára. The complexity of equality constraint languages. *Theory of Computing Systems*, 3(2):136–158, 2008. A conference version appeared in the proceedings of Computer Science Russia (CSR'06).

7 Manuel Bodirsky and Jan Kára. The complexity of temporal constraint satisfaction problems. *Journal of the ACM*, 57(2):1–41, 2009. An extended abstract appeared in the Proceedings of the Symposium on Theory of Computing (STOC'08).

8 Manuel Bodirsky, Barnaby Martin, and Antoine Mottet. Constraint satisfaction problems over the integers with successor. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 256–267, 2015.

9 Manuel Bodirsky and Antoine Mottet. Reducts of finitely bounded homogeneous structures, and lifting tractability from finite-domain constraint satisfaction. Submitted. Preprint available under ArXiv:1601.04520, 2016.

10 Manuel Bodirsky and Jaroslav Nešetřil. Constraint satisfaction with countable homogeneous templates. *Journal of Logic and Computation*, 16(3):359–373, 2006.

11 Manuel Bodirsky and Michael Pinsker. Reducts of Ramsey structures. *AMS Contemporary Mathematics, vol. 558 (Model Theoretic Methods in Finite Combinatorics)*, pages 489–519, 2011.

12 Manuel Bodirsky and Michael Pinsker. Minimal functions on the random graph. *Israel Journal of Mathematics*, 200(1):251–296, 2014.

13 Manuel Bodirsky and Michael Pinsker. Schaefer's theorem for graphs. *Journal of the ACM*, 62(3):Article no. 19, 1–52, 2015. A conference version appeared in the Proceedings of STOC 2011, pages 655–664.

14 Manuel Bodirsky and Michael Pinsker. Topological Birkhoff. *Transactions of the American Mathematical Society*, 367:2527–2549, 2015.

15 Manuel Bodirsky, Michael Pinsker, and András Pongrácz. Projective clone homomorphisms. Preprint arXiv:1409.4601, 2014.

16 Manuel Bodirsky and Michał Wrona. Equivalence constraint satisfaction problems. In *Proceedings of Computer Science Logic*, volume 16 of *LIPICS*, pages 122–136. Dagstuhl Publishing, September 2012.

17 Andrei A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006.

18 Andrei A. Bulatov, Andrei A. Krokhin, and Peter G. Jeavons. Classifying the complexity of constraints using finite algebras. *SIAM Journal on Computing*, 34:720–742, 2005.

19 Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.

20 L. Haddad and Ivo G. Rosenberg. Finite clones containing all permutations. *Canadian Journal of Mathematics*, 46(5):951–970, 1994.

21 Pavol Hell and Jaroslav Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

22 Peter Jeavons, David Cohen, and Marc Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

23 Alistair H. Lachlan and Robert E. Woodrow. Countable ultrahomogeneous undirected graphs. *Transactions of the AMS*, 262(1):51–94, 1980.

24 Jaroslav Nešetřil and Vojtěch Rödl. The partite construction and Ramsey set systems. *Discrete Mathematics*, 75(1-3):327–334, 1989.

25 Emil L. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematics Studies*, 5, 1941.

26 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 216–226, 1978.

27 Katrin Tent and Martin Ziegler. *A course in model theory*. Lecture Notes in Logic. Cambridge University Press, 2012.

28 Simon Thomas. Reducts of the random graph. *Journal of Symbolic Logic*, 56(1):176–181, 1991.

Sensitivity of Counting Queries

Myrto Arapinis[1], Diego Figueira[2], and Marco Gaboardi[*3]

1 **University of Edinburgh, Edinburgh, United Kingdom**
marapini@inf.ed.ac.uk
2 **CNRS, LaBRI, Talence, France**
dfigueir@labri.fr
3 **University at Buffalo, SUNY, Buffalo, USA**
gaboardi@buffalo.edu

Abstract

In the context of statistical databases, the release of accurate statistical information about the collected data often puts at risk the privacy of the individual contributors. The goal of differential privacy is to maximise the utility of a query while protecting the individual records in the database. A natural way to achieve differential privacy is to add statistical noise to the result of the query. In this context, a mechanism for releasing statistical information is thus a trade-off between utility and privacy. In order to balance these two "conflicting" requirements, privacy preserving mechanisms calibrate the added noise to the so-called *sensitivity* of the query, and thus a precise estimate of the sensitivity of the query is necessary to determine the amplitude of the noise to be added.

In this paper, we initiate a systematic study of sensitivity of counting queries over relational databases. We first observe that the sensitivity of a Relational Algebra query with counting is not computable in general, and that while the sensitivity of Conjunctive Queries with counting is computable, it becomes unbounded as soon as the query includes a join. We then consider restricted classes of databases (databases with constraints), and study the problem of computing the sensitivity of a query given such constraints. We are able to establish bounds on the sensitivity of counting conjunctive queries over constrained databases. The kind of constraints studied here are: functional dependencies and cardinality dependencies. The latter is a natural generalisation of functional dependencies that allows us to provide tight bounds on the sensitivity of counting conjunctive queries.

1998 ACM Subject Classification F.4.1 [Mathematical Logic] Model theory, H.2.3 [Languages] query languages

Keywords and phrases Differential privacy, sensitivity, relational algebra

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.120

1 Introduction

With the emergence of new systems and services such as eHealth, electronic tickets (*e.g.*, London Oyster card), mobile phones, or social networks, important amounts of information concerning our everyday activities are collected in various databases. Statistical analysis of such datasets could be very useful for improving services, or enabling research and market studies for example. But at the same time, the collection and storage of all this data puts at risk our individual privacy. A solution to address this problem is not to release the *exact*

* Marco Gaboardi's work was partially supported by NSF grants CNS-1237235 and by EPSRC grant EP/M022358/1.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 120; pp. 120:1–120:13

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

result of any query on a sensitive dataset, but rather to perturb the released results by adding some noise. Differential privacy [3, 6] precisely characterises the level of privacy provided by such randomized mechanisms. It offers a worst-case statistical guarantee on the increase in harm that an individual can be exposed to, if deciding to contribute her data to the dataset.

The concept of differential privacy is rooted in the notion of *neighboring databases*, that is, databases that differ in the presence or not of the information regarding one participant. More precisely, a mechanism $\mathcal{M}$ is ε-differentially private, for $\varepsilon \geq 0$ if for any two neighboring databases D and D' and for any subset $S \subseteq \mathcal{R}$ of possible outputs we have:

$$\Pr[\mathcal{M}(D) \in S] \quad \leq \quad \mathbf{e}^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S].$$

That is, the probability that $\mathcal{M}$ releases an element of S on D is almost the same as the probability that $\mathcal{M}$ releases an element of S on D'. In the definition of differential privacy the parameter ε plays a central role. It gives the concrete bound on the increase in *harm* that an individual I can be exposed to, by contributing her data to the database.

Several mechanisms have been proposed to turn a deterministic query into a differentially private one, like the Laplace mechanism, the exponential mechanism, the Gaussian mechanism, etc. An extended introduction to these and other mechanisms (and more generally to differential privacy) is the recent monograph by Dwork and Roth [7]. In order to provide a good balance between privacy and utility, such ε-differential private mechanisms calibrate the added noise to the so-called *sensitivity* of the query. The sensitivity of a query Q captures the influence that an individual's data can have on the output of the query. More precisely, let us denote by $D \sim D'$ the fact that two databases D and D' are *neighbors*. The sensitivity of a numeric query Q is then

$$\max_{D \sim D'} |Q(D) - Q(D')|.$$

This measure is generally referred to as the *global sensitivity* of the query to distinguish it from other notions of local or smooth sensitivity [13].

To avoid adding too much noise and thus sacrificing too much utility to achieve the intended level of differential privacy, the sensitivity of the query needs to be computed as accurately as possible. However, this problem is undecidable in general as we shall see. In this paper we propose algorithms for computing upper bounds on the sensitivity of queries. Our results hold in a rather general setting: we consider counting conjunctive queries over multi-table databases. Further, our results are not tied to any particular neighboring relation, but hold for any relation of *bounded order*. This work is a first step towards understanding the class of queries and neighboring relations that are amenable to differential privacy.

Relational databases. Most of the works on differential privacy assume the simplified situation where the database is a monolithic table [7]. However, real life databases consist of not one, but many tables containing the information scattered. Of course, one could build a unique table from all these tables, by simply producing the cartesian product of all the tables in the database. Nevertheless, this immediately raises two problems. First, materialising the cartesian product of many—possibly big—tables is impractical, and often plain impossible due to space and time requirements. Second, the notion of neighboring databases now becomes *unbounded* which makes queries have unbounded sensitivity, and thus not amenable to differential privacy mechanisms. For example, given two tables (T_1, T_2) and a neighbor $T_1' = T_1 \setminus \{\bar{t}\}$ of T_1 for some record $\bar{t} \in T_1$, we have that, whereas (T_1', T_2) is the neighbor of (T_1, T_2) resulting from removing *one* record, $T_1 \times T_2$ differs from $T_1' \times T_2$ in a number of records equal to $|T_2|$. This in general makes it impossible for non-trivial queries to have bounded sensitivity, unless further restrictions on the databases are assumed.

Neighboring relation. Most works on differential privacy define neighboring databases as those that differ in exactly one record. This corresponds to assuming that each individual contributes at most one record in the database. However, as pointed out in [10] this assumption does not hold for many applications such as social networks or tabular data. So the definition of neighboring databases needs to be tailored to the application at hand with privacy in mind. Indeed, neighboring databases should, strictly speaking, differ in the complete set of information pertaining to one individual, which could mean more than one record. Alternative definitions of neighboring have been proposed [10, 5]. In particular, our results are not tied to any particular definition of the neighboring relation.

SQL. SQL is arguably the prevalent query language for relational databases. It is equivalent to first order logic (FO) over relational structures and to Relational Algebra (RA). Here, we focus on SQL with aggregation, and study the static analysis problem of computing the sensitivity of SQL queries. As a first step in the larger programme of studying aggregate queries, we study the *counting* operator. We concentrate our investigation on one of the most prominent fragments of SQL, namely the Conjunctive Queries, corresponding to positive "select-from-where" queries [1].

Contributions. We first establish, in Section 2, that finding the sensitivity of a SQL query with counting is not computable in general. In the remaining sections we restrict our study to counting Conjunctive Queries. Section 3 shows that the sensitivity for this fragment is computable, although the characterisation shows that sensitivity becomes unbounded as soon as we have a 'non-trivial' join.

Now, in most scenarios the class of databases of interest for the application at hand are restricted (or constrained), and oftentimes the sensitivity of a query Q restricted to a constrained class of databases can become bounded. Following this idea, we then study the problem of computing global sensitivity restricted to databases from a constrained class. In Section 4, we focus on *Functional Dependencies* (FD), that allow constraining databases by rules of the form "in the table T, the i-th column determines the j-th column", in other words, "there are no 2 rows of T with the same datum in the i-th column but distinct data in their j-th columns". Further, in Section 5 we study *Cardinality Dependencies*, which are a generalisation of FDs, with rules of the form "there are no more than k rows of T with the same datum in the i-th column but pairwise distinct data in their j-th columns". Finally, Section 6 concludes and discusses future work.

Related work. Several works have studied methods for computing the sensitivity of a given query or program. The work most related to ours is the one of Palamidessi and Stronati [14]. They study the problem of computing the sensitivity of queries in relational algebra. Their approach is based on the use of constraints on attributes: every attribute comes with a bounded range, e.g. $0 \leq \text{age} \leq 100$. They are able to provide tight bounds on the sensitivity of the query Q. This approach can be applied to general SQL queries but it has the drawback that it requires to constrain the ranges for all the attributes. In this paper, instead, we focus on counting queries and on more lax semantic restrictions, namely functional dependencies and cardinality dependencies.

Pierce and Reed [15] and Gaboardi et al. [8] use relational algebra operations with a fixed, predetermined sensitivity, and a linear type system to track the use of the data in programs. This combination permits to have sensitivity analyses that extend, beyond SQL, to a full functional programming language. Their approach can provide "bad" estimates on

the sensitivity of given queries due to the use of fixed sensitivity for relational operations. Our approach could provide a kernel query language providing more precise estimates that could be then combined with their type systems.

Chaudhuri et al. in [4] study automatic program analyses that provide bounds on the sensitivity of numerical imperative programs. Their approach is not directly related to specific query languages but our work could, in principle, be combined with their techniques to design a general purpose programming language for differential privacy.

Several works have pointed out and studied the problem of providing a bound to the sensitivity of queries in disconnected structures. McSherry [12], in the setting of tabular data, considers a restricted form of join where the data of the two tables are grouped by their join keys, and then groups are joined using their group keys. The same solution has been used also in [15, 8]. A similar approach, with different restrictions, has also been used by Palamidessi and Stronati in [14]. This approach limits the situations where differential privacy can be used with a good utility. To overcome this problem, several approaches considered alternative notions of sensitivity such as *local sensitivity* [13] or *empirical sensitivity* [5].

2 Preliminaries

Let $\mathbb{N} = \{0, 1, 2, \dots\}$ and let $\underline{n} = \{1, \dots n\}$ for every $n \in \mathbb{N}$. We write $\bar{a}$ to denote a vector of elements, whose i-th element is denoted by $\bar{a}[i]$. We write A^* [resp. A^+, A^n] for the set of strings [resp. non-empty strings, length-n strings] over A, and ε for the empty string.

2.1 Relational structures

A **relational vocabulary** $\sigma = (\mathcal{K}, \mathcal{R})$ consists of a collection $\mathcal{K}$ of **constants** (usually denoted by $c_1, c_2, \dots$), and a collection $\mathcal{R}$ of **relation symbols**, each with a specified **arity**. By σ_n we denote a vocabulary $\sigma_n = (\mathcal{K}, \mathcal{R})$ where $\mathcal{K} = \{c_1, \dots, c_n\}$. For a relation R we write $\mathit{arity}(R) \in \mathbb{N}$ to denote its arity; and we sometimes write $R^{(r)}$ to specify that R has arity r. A **σ-structure** $\mathbb{A}$ consists of a universe A containing $\mathcal{K}$, or **domain**, and an **interpretation** which associates to each relation symbol $R \in \mathcal{R}$, a relation $R^{\mathbb{A}} \subseteq A^{\mathit{arity}(R)}$, and for each constant $c \in \mathcal{K}$, $c^{\mathbb{A}} = c$. An **isolated element** of $\mathbb{A}$ is an element $a \in A$ which does not appear in any interpretation. Let STR be the set of all finite structures (we write $\mathit{STR}[\sigma]$ to make explicit the vocabulary). We use $\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{A}', \mathbb{B}', \dots$ to denote relational structures from STR, and $A, B, C, A', B', \dots$ to denote their respective domains.

▶ **Example 1.** As our running example, we will consider a database of patients, doctors and hospitals, with tables

- *Hos(id, loc)*, containing the hospitals with their location,
- *Pat(id, sex, hos)*, listing the patients with an identifier, gender and the hospital where they are being treated,
- *Doc(id, specialty, hos)*, listing the doctors with their identifier, their specialty and the hospital where they practice,
- *PatDoc(pat, doc)*, containing the patients and their current attending doctor.

Such a database can be described over the vocabulary $\sigma = (\mathcal{K}, \mathcal{R})$ containing relations $\mathcal{R} = \{\mathit{Hos}^{(2)}, \mathit{Pat}^{(3)}, \mathit{Doc}^{(3)}, \mathit{PatDoc}^{(2)}\}$ and some constants such as $\mathcal{K} = \{c_{\mathrm{F}}, c_{\mathrm{O}}\}$.

A **graph** is a structure $G = (V, E)$, where E is a binary relation that is symmetric and irreflexive. Thus, our graphs are undirected, loopless, and without parallel edges. The **Gaifman graph** of a σ-structure $\mathbb{A}$, denoted by $\mathcal{G}(\mathbb{A})$, is the (undirected) graph whose set of nodes is the universe of $\mathbb{A}$, and whose set of edges consists of all pairs (a, a') of distinct

elements of A such that a and a' appear together in some tuple of a relation in $\mathbb{A}$. Recall that the **distance** between two vertices u, v of a graph is the length of the shortest path from u to v. We define the distance between two elements a, b of a structure $\mathbb{A}$ as their distance in $\mathcal{G}(\mathbb{A})$, which we denote by $dist_{\mathbb{A}}(a, b)$. We write $A \sqcup B$ for the disjoint union of A and B.

A **homomorphism** from a $(\mathcal{K}, \mathcal{R})$-structure $\mathbb{A}$ to a $(\mathcal{K}', \mathcal{R}')$-structure $\mathbb{B}$ such that $\mathcal{K} \subseteq \mathcal{K}'$ and $\mathcal{R} \subseteq \mathcal{R}'$ is a mapping $h : A \to B$ so that for each relation symbol $R \in \mathcal{R}$, if $(a_1, \ldots, a_r) \in R^{\mathbb{A}}$, then $(h(a_1), \ldots, h(a_r)) \in R^{\mathbb{B}}$, and for every constant $c \in \mathcal{K}$, $h(c) = c$. We will sometimes write $h(a_1, \ldots, a_r)$ as short for $(h(a_1), \ldots, h(a_r))$. We write $\mathbb{A} \to \mathbb{B}$ to denote that there is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$, and we write $h : \mathbb{A} \to \mathbb{B}$ to denote that h is a homomorphism from $\mathbb{A}$ to $\mathbb{B}$. If $\mathbb{A} \to \mathbb{B}$ and $\mathbb{B} \to \mathbb{A}$ we say that $\mathbb{A}$ and $\mathbb{B}$ are **hom-equivalent**. We use $\cong$ for the isomorphism relation. Given a σ-structure $\mathbb{A}$ and a set $B \subseteq A$ there is (up to isomorphism) a unique structure $\mathbb{A}'$ so that

- it is hom-equivalent to $\mathbb{A}$, that is, there are $h : \mathbb{A} \to \mathbb{A}'$ and $h' : \mathbb{A}' \to \mathbb{A}$,
- $h(a) = h'(a) = a$ for all $a \in B$,
- it has the minimal number of elements.

Such a structure $\mathbb{A}'$ is called the **core preserving** B (or simply **core** if $B = \emptyset$). We write $core(\mathbb{A}, B)$ [resp. $core(\mathbb{A})$] to denote the core of $\mathbb{A}$ preserving B [resp. the core of $\mathbb{A}$].

2.2 Logic

Let $\mathcal{V}$ be a collection of first-order variables equipped with a linear order $<$. Let σ be a relational vocabulary. A **term** is either a first order variable $x \in \mathcal{V}$ or a constant from σ. The **atomic formulas** of σ are those of the form $R(t_1, \ldots, t_n)$, where $R \in \sigma$ is a relation symbol of arity r, and $t_1, \ldots, t_r$ are terms. Formulas of the form $t = t'$ are also atomic formulas, and we refer to them as **equalities**. The collection of **first-order formulas** (FO formulas) is obtained by closing the atomic formulas under negation, conjunction, disjunction, universal and existential first-order quantification. The semantics of first-order logic is standard. The set of variables of φ is denoted by $var(\varphi)$, and the set of free variables by $free(\varphi)$. We often write $\varphi(x_1, \ldots, x_n)$ where $\{x_1, \ldots, x_n\} = free(\varphi)$ and $x_1 < \cdots < x_n$, to stress the free variables. If $\mathbb{A}$ is a σ-structure and $\varphi(\bar{x})$ is a first-order formula, we use the notation $\mathbb{A} \models \varphi[\bar{a}]$ to denote the fact that φ is true in $\mathbb{A}$ when its free variables $\bar{x}$ are interpreted by the tuple of elements $\bar{a}$. When φ contains no free variables, we say that it is a **sentence**, and in this case we simply write $\mathbb{A} \models \varphi$. For any formula $\varphi(x_1, \ldots, x_n)$ and structure $\mathbb{A}$, we write $\varphi(\mathbb{A})$ to denote $\{(a_1, \ldots, a_n) \in A^n \mid \mathbb{A} \models \varphi[a_1, \ldots, a_n]\}$. We use '()' to denote the 0-ary tuple of elements. Hence, if φ has no free variables we interpret $\varphi(\mathbb{A})$ as $\{()\}$ if $\mathbb{A} \models \varphi$ or $\emptyset$ otherwise. Note that, in this case, $|\varphi(\mathbb{A})| = 1$ iff $\mathbb{A} \models \varphi$. We use $\equiv$ for the logical equivalence relation and $\equiv_{\mathcal{C}}$ for the equivalence relation restricted to a class of structures $\mathcal{C}$.

Given a class of FO formulas $\mathcal{L}$, by $\mathcal{L}^{\#}$ we denote the class of **counting queries** $\{\#\varphi \mid \varphi \in \mathcal{L}\}$. The evaluation of $\#\varphi$ in $\mathbb{A}$, denoted $\#\varphi(\mathbb{A})$, is defined as $|\varphi(\mathbb{A})|$, that is, as the number of distinct tuples making φ true in $\mathbb{A}$.

▶ **Example 2.** Continuing our running example, we consider the query that counts the number of oncology doctors that are treating female patients in the same hospital as they practice:

```
SELECT count distinct Doc.id
FROM Pat, Doc, PatDoc
WHERE Doc.specialty = 'O' and
      Pat.sex = 'F' and
      Pat.hos = Doc.hos and
      PatDoc.pat = Pat.id and
      PatDoc.doc = Doc.id
```

This can be equivalently expressed with the formula $\#\varphi$, where

$$\varphi(x_{doc}) = \exists x_{pat}, x_{hos} \; . \; Doc(x_{doc}, c_{\mathrm{O}}, x_{hos}) \wedge Pat(x_{pat}, c_{\mathrm{F}}, x_{hos}) \wedge PatDoc(x_{pat}, x_{doc}) \, .$$

2.3 Global sensitivity

In its standard formulation, Differential Privacy requires the privacy bound to be valid for every pair of structures that differ in one record. However, it is possible that an individual contributes more than a single record to the database. Further it may be that the database contains tables with public information. For this reason we do not set for our study a particular neighboring relation. Our results hold for any **neighboring relation** $\mathcal{N} \subseteq STR[\sigma] \times STR[\sigma]$.

Having said that, a specific neighboring relation, called **1-neighboring**, will be particularly useful for our proofs. Given two σ-structures $\mathbb{A}, \mathbb{B}$ with $\sigma = (\mathcal{K}, \mathcal{R})$, we say that $\mathbb{A}$ is a **substructure** of $\mathbb{B}$ (noted $\mathbb{A} \subseteq \mathbb{B}$) if $A \subseteq B$, and $R^{\mathbb{A}} \subseteq R^{\mathbb{B}}$ for all $R \in \sigma$. We write $\mathbb{A} \prec \mathbb{A}'$ if $\mathbb{A} \subsetneq \mathbb{A}'$ and there is no $\mathbb{B}$ so that $\mathbb{A} \subsetneq \mathbb{B} \subsetneq \mathbb{A}'$. We say that $\mathbb{A}, \mathbb{B}$ are 1-neighboring structures, noted $\mathbb{A} \sim_1 \mathbb{B}$, if $\mathbb{A} \prec \mathbb{B}$ or $\mathbb{B} \prec \mathbb{A}$. In other words, $\mathbb{A} \sim_1 \mathbb{B}$ if $\mathbb{A}$ can be obtained from $\mathbb{B}$ (and B from A) by removing/adding a tuple or an isolated node.

We say that the neighboring relation $\mathcal{N}$ is **of order** $k \in \mathbb{N}$, if any two neighboring relational structures differ in at most k elements. More formally, $\mathcal{N}$ is of order k if for any $(\mathbb{A}, \mathbb{B}) \in \mathcal{N}$, there exist $\mathbb{A}_0, \ldots, \mathbb{A}_\ell$ such that $\ell \leq k$, $\mathbb{A} = \mathbb{A}_0$, $\mathbb{B} = \mathbb{A}_\ell$ and $\mathbb{A}_{i-1} \sim_1 \mathbb{A}_i$ for all $i \in \underline{\ell}$. We say that the neighboring relation is unbounded if no such k exists.

The **global sensitivity** of a function $f : STR \to \mathbb{N}$ over a class of models $\mathcal{C} \subseteq STR$ with respect to a neighboring relation $\mathcal{N} \subseteq \mathcal{C} \times \mathcal{C}$ is:

$$\mathrm{GS}^{\mathcal{N}}_{\mathcal{C}}(f) \stackrel{def}{=} \max_{(\mathbb{A},\mathbb{A}') \in \mathcal{N}} |f(\mathbb{A}) - f(\mathbb{A}')|.$$

▶ **Example 3.** Suppose now that we want to find out the number of oncological patients in the state of New York with the query

$$\varphi(x_{pat}) = \exists x_{hos}, x_{doc}, x_{sex} \; . \\ Doc(x_{doc}, c_{\mathrm{O}}, x_{hos}) \wedge Pat(x_{pat}, x_{sex}, x_{hos}) \wedge PatDoc(x_{pat}, x_{doc}) \wedge Hos(x_{hos}, x_{loc})$$

It is not hard to see that this query has unbounded global sensitivity when all relations are considered sensitive, and thus all databases that differ in any one element are neighbors. Indeed changing the location of a hospital from Indiana to New-York can increase the number of ontological patients in the state of New York by any number.

▶ **Observation 1.** *For any neighboring relation $\mathcal{N}$ of order k and any class of databases $\mathcal{C}$, the global sensitivity of a query Q is bounded with respect to $\mathcal{N}$ over $\mathcal{C}$ iff it is bounded with respect to $\sim_1$ over $\mathcal{C}$. Further, the global sensitivity with respect to $\mathcal{N}$ and relative to the class $\mathcal{C}$ is bounded by $k \cdot GS^{\sim_1}_{\mathcal{C}}(Q)$. So in the remaining of the paper we focus on 1-neighboring.*

We will study the following problem, given a query language $\mathcal{L}$, and a class of relational structures $\mathcal{C}$

Problem:	GlobalSensitivity($\mathcal{L}$, $\mathcal{C}$)
Input:	$Q \in \mathcal{L}$
Output:	$\mathrm{GS}^{\sim_1}_{\mathcal{C}}(Q)$

Unfortunately, this problem is undecidable already for counting first-order logic (and therefore for counting Relational Algebra [1]).

▶ **Theorem 4.** GlobalSensitivity*(FO$^{\#}$, STR) is non-computable.*

The fact that the global sensitivity problem for FO is undecidable is not really surprising since most static analysis problems for FO on unrestricted structures are undecidable. This is why in the next sections we will focus on Conjunctive Queries.

3 Conjunctive queries

One of the most studied fragments of FO in relation to database queries is the fragment of *Conjunctive Queries* (CQ). We now, and for the rest of the paper, restrict our study to counting conjunctive queries, and show that sensitivity for this fragment is computable.

The class of Conjunctive Queries (also known as Primitive Positive Logic, or Existential Positive FO) is the fragment of FO corresponding to positive '*select-project-join*' queries of the Relational Algebra or to positive '*select-from-where*' queries of SQL, where by 'positive' we mean that there are no inequalities in the *select* [resp. *where*] conditions (we refer the reader to [1, §4] for more details). These are formulae of the form

$$\varphi(x_1,\dots,x_n) = \exists y_1,\dots,y_m\, \theta, \tag{$\dagger$}$$

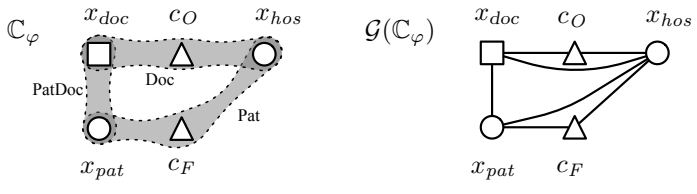
where θ is a conjunction of atomic formulae. Since we deal with constants, and, in future sections, with constrained databases, a conjunctive query can also be *false* (noted $\bot$). However, all the results that we show will assume that the input formula is not equivalent to $\bot$ (*i.e.*, that it is satisfiable, which can be checked in polynomial time)—for the particular case where formulae are unsatisfiable all the results are trivial, and this will avoid lengthy statements. For simplicity, we assume that the formulae do not contain equalities.

Every conjunctive query of the form (†) over a relational vocabulary σ_k gives rise to a **canonical structure** (sometimes called *tableau*) $\mathbb{C}_\varphi$ with $n+m+k$ elements, where the elements of $\mathbb{C}_\varphi$ are the variables $x_1,\dots,x_n,y_1,\dots,y_m$ plus the constants $c_1,\dots,c_k$, the relations of $\mathbb{C}_\varphi$ consist of the tuples of terms in the conjuncts of θ. Given a CQ φ, we write $\mathbb{C}_\varphi$ for the canonical structure of φ, and C_φ for its domain (*i.e.*, the variables $x_1,\dots,x_n,y_1,\dots,y_m$ and constants $c_1,\dots,c_k$). We also define $\mathbb{C}_\varphi^-$ as the result of removing all isolated constants from $\mathbb{C}_\varphi$ (note that $\mathbb{C}_\varphi^-$ may not necessarily be a structure over the same vocabulary of φ due to the absence of some constants). Likewise, any σ_k-structure $\mathbb{A}$ with domain $A = \{x_1,\dots,x_n\} \cup \{c_1,\dots,c_k\}$ gives rise to a canonical CQ $\varphi(x_1,\dots,x_n)$ where $\mathit{var}(\varphi) = \mathit{free}(\varphi) = \{x_1,\dots,x_n\}$, and φ has a conjunct $R(\bar t)$ iff $\bar t \in R^{\mathbb{A}}$. Note that for every σ_k-structure $\mathbb{A}$ there is $\mathbb{A}' \cong \mathbb{A}$ and φ so that φ is the canonical query of $\mathbb{A}'$.

A CQ φ is **acyclic** if $\mathcal{G}(\mathbb{C}_\varphi)$ is acyclic. We say that a CQ φ is **connected** if $\mathcal{G}(\mathbb{C}_\varphi^-)$ is connected, otherwise it is **disconnected**. Note that every disconnected CQ φ so that $\mathcal{G}(\mathbb{C}_\varphi)$ has n connected components can be equivalently written in the form $\varphi = \bigwedge_{i\in\underline{n}} \psi_i(\bar x_i)$ so that $\psi_i(\bar x_i)$ is a connected CQ for every i, and for all $i \neq j$, $\bar x_i$ and $\bar x_j$ have no variables in common. We say that ψ_i is a **connected conjunct** of φ, and we say that ψ_i is a **sentential connected conjunct** if it is a sentence (*i.e.*, $\bar x_i = ()$). Given $\varphi = \bigwedge_{i\in\underline{n}} \psi_i(\bar x_i)$ a disconnected CQ with each ψ_i being a connected conjunct, we further define $\overline{\varphi}^j$ as the conjunction of all the ψ_s's but ψ_j.

▶ **Example 5** (Cont. from Ex. 2)**.** The canonical σ-structure $\mathbb{C}_\varphi$ has universe $\{x_{pat}, x_{hos}, x_{doc}, c_{\mathrm{O}}, c_{\mathrm{F}}\}$ and relations (shown in Figure 1):

$$\mathit{Doc}^{\mathbb{C}_\varphi} = \{(x_{doc}, c_{\mathrm{O}}, x_{hos})\}, \quad \mathit{Pat}^{\mathbb{C}_\varphi} = \{(x_{pat}, c_{\mathrm{F}}, x_{hos})\}, \quad \mathit{PatDoc}^{\mathbb{C}_\varphi} = \{(x_{pat}, x_{doc})\}.$$



]t]

Figure 1 Depiction of the canonical structure of φ as defined in Example 2 as well as its Gaifman graph. Square vertices denote free variables and triangle vertices denote constants.

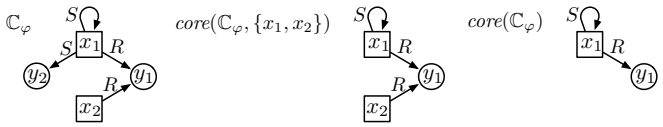


Figure 2 Core of CQ's.

Core of CQ's. For a CQ query $\varphi(\bar{x}) = \exists \bar{y}.\theta$ over σ_k we define $core(\varphi)$ as the CQ query $\varphi'(\bar{x}) = \exists \bar{y}'.\theta'$ where θ' is the canonical query of $core(\mathbb{C}_\varphi, \bar{x})$ and $\bar{y}'$ is the set of all non-constant elements of $core(\mathbb{C}_\varphi, \bar{x})$ that are not in $\bar{x}$. Note that $\mathbb{C}_{core(\varphi)} \cong core(\mathbb{C}_\varphi, \bar{x})$. We say that $\varphi(\bar{x})$ is a **core-CQ** if $\mathbb{C}_{core(\varphi)} \cong core(\mathbb{C}_\varphi)$, and we write CQ_{core} for the class of all core-CQ's. We define $core(\#\varphi)$ as $\#core(\varphi)$ for every CQ φ.

▶ **Example 6.** Given $\varphi(x_1, x_2) = \exists y_1, y_2.S(x_1, x_1) \wedge S(x_1, y_2) \wedge R(x_1, y_1) \wedge R(x_2, y_1)$, whose canonical structure is depicted in Figure 2, we have that $core(\varphi) \equiv \exists y_1.S(x_1, x_1) \wedge R(x_1, y_1) \wedge R(x_2, y_1)$, and that φ is not a core-CQ since $core(\mathbb{C}_\varphi, \{x_1, x_2\})$ is not isomorphic to $core(\mathbb{C}_\varphi)$, as shown in Figure 2.

Given a connected CQ φ, let us define

$$\Delta_{STR}(\#\varphi) = \begin{cases} \infty & \text{if } \exists x \in free(\varphi).\ \exists R \in \mathcal{R}.\ \exists \bar{a} \in R^{core(\varphi)}.\ x \notin \bar{a} \\ 1 & \text{otherwise} \end{cases}$$

▶ **Proposition 7.** *For every connected CQ$^\#$ Q, we have $GS^{\sim_1}_{STR}(Q) = \Delta_{STR}(Q)$.*

▶ **Example 8** (Cont. from Ex. 5)**.** Note that we have $\Delta_{STR}(\#\varphi) = \infty$ since $core(\varphi) = \varphi$ and x_{doc} is not in the tuple (x_{pat}, c_F, x_{hos}) of the relation $Pat^{\mathbb{C}_\varphi}$, and thus that $\mathrm{GS}^{\sim_1}_{STR}(Q) = \infty$.

We extend the definition of Δ_{STR} to disconnected CQ$^\#$ as follows. For any $\varphi = \bigwedge_{i \in \underline{n}} \varphi_i$ disconnected CQ so that each φ_i is a connected conjunct, we define

$$\Delta_{STR}(\#\varphi) = \begin{cases} \Delta_{STR}(\#\varphi_k) & \text{if } \exists k \in \underline{n}.\ free(\varphi) = free(\varphi_k) \ \wedge\ \mathbb{C}_{\overline{\varphi}^k} \to \mathbb{C}_{\varphi_k} \\ \infty & \text{otherwise} \end{cases}$$

▶ **Theorem 9.** *For every CQ$^\#$ Q, we have $GS^{\sim_1}_{STR}(Q) = \Delta_{STR}(Q)$.*

The above characterization shows that, even when we deal with *connected* CQ's (arguably the most common), we obtain unbounded sensitivity very easily. Indeed, as soon as one has a 'join' with a free variable which is not the joining attribute, such as $\#\varphi(x) = \#\,\exists y, z\,.\,R(x, y) \wedge S(y, z)$ the global sensitivity is unbounded. Although this means that for every $N \in \mathbb{N}$ there are structures $\mathbb{A} \sim_1 \mathbb{A}'$ so that $\#\varphi(\mathbb{A}) - \#\varphi(\mathbb{A}') > N$, it may be that $\mathbb{A}, \mathbb{A}'$ do not correspond to databases that could arise in the domain of application at hand. However, when restricting the set of considered structures to ones satisfying some constraints, it may well be that the sensitivity becomes bounded. The next two sections will focus on evaluating sensitivity of queries over constrained structures.

4 Functional Dependencies

In this section we show bounds for the sensitivity of queries in the presence of what are called *functional dependencies*. In databases, it is often the case that a set of attributes determines another attribute. Such constraints are called functional dependencies. In this section functional dependencies where one attribute determine another are considered.

▶ **Example 10** (Cont. from Ex. 2)**.** Note that, the global sensitivity of $\#\varphi$ is unbounded. Indeed, this is a consequence of the possibility of having patients with unbounded number of attending doctors and doctors working in any number of hospitals. However, this does not correspond to databases that could occur in practice, since patients have normally one attending doctor and doctors work in at most one hospital. This is why the use of database constraints becomes useful, to restrict the collection of databases we are interested in, and thus to improve the bounds of the sensitivity of queries.

We write $R[i \to j]$ to denote a **functional dependency of a relation** R **of arity** n **between components** $i \in \underline{n}$ **and** $j \in \underline{n}$. A structure $\mathbb{A}$ satisfies a functional dependency (henceforth "FD") $R[i \to j]$ if $\max_{a \in A}(|\{\bar{b}[j] \mid \bar{b} \in R^{\mathbb{A}}, \bar{b}[i] = a\}|) \leq 1$. We use the symbol Σ to denote a set of FDs, and we write $\#_\Sigma R[i \to j]$ to denote 1 if $R[i \to j] \in \Sigma$, or ∞ otherwise. We write $\mathcal{C}_\Sigma$ for the class of all relational structures satisfying all FDs in Σ.

Given a CQ query φ and a set of FDs Σ we define the **Σ-chase** [11, 2] of φ, noted $chase_\Sigma(\varphi)$, as the closure of the application of the following rule:

- For every $R[i \to j] \in \Sigma$ and every pair of conjuncts $R(\bar{t})$ and $R(\bar{s})$ of φ so that $\bar{t}[i] = \bar{s}[i]$ and $\bar{t}[j] \neq \bar{s}[j]$,
  - if $\bar{s}[j]$ is a variable, replace every occurrence of $\bar{s}[j]$ with $\bar{t}[j]$;
  - if $\bar{s}[j]$ and $\bar{t}[j]$ are constants, output $\bot$.

It can be seen that the application of these rules is terminating and Church-Rosser confluent, up to renaming of variables [1].

The following result shows that, as soon as we have a disconnected query, the sensitivity is likely to be unbounded.

▶ **Proposition 11.** *For every disconnected CQ query φ containing a conjunct without constants and at least one free variable, and for every set Σ of FD's, we have $GS^{\sim_1}_{\mathcal{C}_\Sigma}(\#\varphi) = \infty$.*

Paths. A **path** of a $(\mathcal{K}, \mathcal{R})$-structure $\mathbb{A}$ between an element $a \in A$ and $b \in A$, is a string

$$p = (R_1, i_1, a_1, j_1, b_1) \cdots (R_n, i_n, a_n, j_n, b_n) \in (\mathcal{R} \times \mathbb{N} \times A \times \mathbb{N} \times A)^* \qquad (\star)$$

so that either $p = \varepsilon$ and $a = b$ (*i.e.*, the empty path); or $a_1 = a$, $b_n = b$, $a_i = b_{i-1}$ for all $1 < i \leq n$, and for every $\ell \in \underline{n}$ we have $i_\ell, j_\ell \in \underline{arity(R_i)}$ and there is $\bar{a} \in R^{\mathbb{A}}_\ell$ so that $\bar{a}[i_\ell] = a_\ell$ and $\bar{a}[j_\ell] = b_\ell$. A path of the form $(\star)$ is **simple** if $a_i \neq b_i \neq b_j$ for all $1 \leq i < j \leq n$. Note that in particular the empty path ε is simple. We write $p : A_1 \rightsquigarrow_{\mathbb{A}} A_2$ to denote that p is a simple path of $\mathbb{A}$ from an element of $A_1 \subseteq A$ to an element of $A_2 \subseteq A$. We write $a \rightsquigarrow_{\mathbb{A}} b$, $A_1 \rightsquigarrow_{\mathbb{A}} b$, $a \rightsquigarrow_{\mathbb{A}} A_2$ as short for $\{a\} \rightsquigarrow_{\mathbb{A}} \{b\}$, $A_1 \rightsquigarrow_{\mathbb{A}} \{b\}$, $\{a\} \rightsquigarrow_{\mathbb{A}} A_2$ respectively.

▶ **Example 12.** For a structure $\mathbb{A}$ with relations $R^{\mathbb{A}} = \{(a_1, a_2, a_3), (a_1, a_4, a_6)\}$, $S^{\mathbb{A}} = \{(a_5, a_6)\}$, and $T^{\mathbb{A}} = \{(a_3, a_5, a_4)\}$, we have that $p : a_1 \rightsquigarrow_{\mathbb{A}} a_4$ for $p = (R, 1, a_1, 3, a_3)$ $(T, 1, a_3, 2, a_5)$ $(S, 1, a_5, 2, a_6)$ $(R, 3, a_6, 2, a_4)$, as depicted in Figure 3

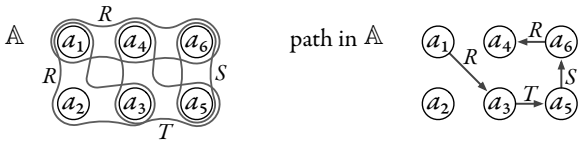


Figure 3 A path in a structure.

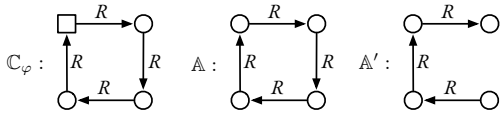


Figure 4 Structures of Example 14. Square vertices denote free variables.

Given a vocabulary $\sigma = (\mathcal{K}, \mathcal{R})$ and a path p of the form $(\star)$, let $m \in \underline{n}$ be the greatest index m so that $b_m \in \mathcal{K}$, or 0 otherwise. We define the **cardinality of path** p as

$$\#_\Sigma(p) \stackrel{def}{=} \prod_{m<\ell\leq n} \#_\Sigma R_\ell[i_\ell \to j_\ell] \tag{1}$$

where as usual the product of the empty sequence is 1, and ∞ is absorbing wrt the product ($\infty \cdot N = N \cdot \infty = \infty$). Note that $\#_\Sigma(\varepsilon) = 1$. The intuition is that $\#_\Sigma(p)$ gives a bound on how many different elements b can be reached from a through p on any structure $\mathbb{A} \in \mathcal{C}_\Sigma$ (*i.e.*, so that $p : a \rightsquigarrow_{\mathbb{A}} b$).

Let $Q = \#\psi(x_1, \ldots, x_n)$ be a connected CQ$^\#$ over a vocabulary $\sigma = (\mathcal{K}, \mathcal{R})$, and let $\varphi = \mathit{core}(\mathit{chase}_\Sigma(\psi))$. We define

$$\Delta^+_\Sigma(Q) \stackrel{def}{=} \max_{R\in\mathcal{R}} \sum_{\bar{a}\in R^{\mathbb{C}_\varphi}} \max_{i\in\underline{n}} \left(\min_{p_i : \bar{a} \rightsquigarrow_{\mathbb{C}_\varphi} x_i} \#_\Sigma(p_i) \right)$$

$$\Delta^-_\Sigma(Q) \stackrel{def}{=} \max_{R\in\mathcal{R}} \max_{\bar{a}\in R^{\mathbb{C}_\varphi}} \max_{i\in\underline{n}} \left(\min_{p_i : \bar{a} \rightsquigarrow_{\mathbb{C}_\varphi} x_i} \#_\Sigma(p_i) \right).$$

▶ **Observation 2.** *Note that $\Delta^-_\Sigma(Q)$ is either 1 or ∞ and that $\Delta^-_\Sigma(Q) = \infty$ iff $\Delta^+_\Sigma(Q) = \infty$. Further, observe that $\Delta^+_\Sigma(Q) - \Delta^-_\Sigma(Q) \leq n_Q - 1$, where n_Q is the maximum number of elements in a relation of the canonical structure of $\mathit{core}(\mathit{chase}_\Sigma(\psi))$, assuming $Q = \#\psi$.*

▶ **Theorem 13.** *Given a set Σ of functional dependencies and a connected CQ$^\#$ query Q, we have that $GS^{\sim_1}_{\mathcal{C}_\Sigma}(Q) \leq \Delta^+_\Sigma(Q)$. Further, if $Q \in \mathrm{CQ}^\#_{core}$, we have $GS^{\sim_1}_{\mathcal{C}_\Sigma}(Q) \geq \Delta^-_\Sigma(Q)$.*

▶ **Observation 3.** *When computing lower and upper bounds for global sensitivity of query $Q = \#\psi(x_1, \ldots, x_n)$ in the presence of functional dependencies, we consider $\mathit{core}(\mathit{chase}_\Sigma(\psi))$ (rather than $\mathit{core}(\psi)$) as it gives a corresponding* canonical minimal *query. This allows us to obtain tighter bounds than if we hadn't taken the chase of ψ.*

▶ **Example 14.** Take for instance the CQ with one free variable of Figure 4. Observe that, for $\Sigma = \{R[1 \to 2],\ R[2 \to 1]\}$, we have that $\mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(\#\varphi) \leq \Delta^+_\Sigma(\#\varphi) = 4$, which is tight since $\#\varphi(\mathbb{A}) = 4$, and $\#\varphi(\mathbb{A}') = 0$. Further, this example can be easily generalized, obtaining that for every $n \in \mathbb{N}$ there is a CQ Q so that $\mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(Q) = n = \Delta^+_\Sigma(Q)$.

▶ **Example 15** (Cont. from Ex. 2)**.** As noted in Example 10, $\#\varphi$ has unbounded global sensitivity. However, if every patient has no more than one attending doctor, the sensitivity

of $\#\varphi$ becomes bounded. Indeed, if $\Sigma = \{PatDoc[1 \to 2]\}$, then

$$\Delta^{-}_{\Sigma}(\#\varphi) \leq \mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(\#\varphi) \leq \Delta^{+}_{\Sigma}(\#\varphi)$$

by Theorem 13—observe that $\varphi \in \mathrm{CQ}_{core}$ since it is unary. Since $\Delta^{-}_{\Sigma}(\#\varphi) = \Delta^{+}_{\Sigma}(\#\varphi) = 1$, it thus follows that $\mathrm{GS}^{\sim_1}_{\mathcal{C}_\Sigma}(\#\varphi) = 1$.

As we have shown, adding functional dependencies immediately improves the global sensitivity of queries. However, functional dependencies are often very restrictive, and it may not always be possible to impose such restrictions. This leads to a more general notion of dependencies, that we call *cardinality dependencies*. These dependencies bound the number of elements associated with component i of a relation R for each fixed element of a component j. This will be the object of study of our next section.

5 Cardinality Dependencies

While functionality constraints are a very natural restriction of databases, there are many scenarios in which, although we don't have an attribute i functionally determining an attribute j in a relation, we have a **cardinality dependency** nonetheless. This is a dependency of the form "there are at most n different attributes j sharing the same attribute i in the relation R"—functional dependencies being the special case when $n = 1$.

These dependencies arise naturally when modelling relations between entities (such as in ER modelling [9]). For example, the business rules underlying a company database may allow that an employee has more than one manager, but no more than 2. Another example is for bounded domain attributes: whereas the name of a person does not determine the gender, there cannot be more than two possibilities of gender for any given name. As we will see next, cardinality dependencies provide further means to give tighter bounds for the global sensitivity of CQ's.

▶ **Example 16** (Cont. from Ex. 15)**.** We already noticed that constraining each patient to have at most one attending doctor, brings the sensitivity of $\#\varphi$ down to 1. However, it may be that a patient can have more than one attending doctor, although it can't have an *unbounded* number of attending doctors. For example, a scenario in which a patient has *at most* 3 attending doctors.

More formally, we write $R[i \xrightarrow{k} j]$ to denote a **k-cardinality dependency of a relation R of arity n between components $i \in \underline{n}$ and $j \in \underline{n}$**. A structure $\mathbb{A}$ satisfies a cardinality dependency (henceforth "CD") $R[i \xrightarrow{k} j]$ if $\max_{a \in A}(|\{\bar{b}[j] \mid \bar{b} \in R^{\mathbb{A}}, \bar{b}[i] = a\}|) \leq k$. For the particular case where $k = 1$, note that $R[i \xrightarrow{k} j]$ is a *functional* dependency. We use the symbol Σ to denote a set of CD's, and we write $\#_\Sigma R[i \to j]$ to denote the minimum k so that $R[i \xrightarrow{k} j] \in \Sigma$, or ∞ otherwise. As before, we write $\mathcal{C}_\Sigma$ for the class of all relational structures satisfying all CDs in Σ. We define the cardinality of a path $\#_\Sigma(p)$ as in (1), where now Σ is a set of CD's, and in the definition $\#_\Sigma R[i \to j]$ is interpreted as defined above, over CD's.

Upper bound. Given a connected $\mathrm{CQ}^{\#}$ query Q over a vocabulary $\sigma = (\mathcal{R}, \mathcal{K})$ so that $core(Q) = \#\varphi(x_1, \ldots, x_n)$, let us define

$$\Delta^{+}_{\Sigma}(Q) \stackrel{def}{=} \max_{R \in \mathcal{R}} \left(\sum_{\bar{a} \in R^{\mathbb{C}_\varphi}} \left(\min_{\substack{p_1,\ldots,p_n \text{ s.t.} \\ p_i : \bar{a} \rightsquigarrow_{\mathbb{C}_\varphi} x_i \text{ for } i \in \underline{n}}} \left(\prod_i \#_\Sigma(p_i) \right) \right) \right).$$

▶ **Observation 4.** *Note that in the presence of cardinality dependencies, when computing upper bounds for global sensitivity of a query $Q = \#\psi(x_1, \ldots, x_n)$, we now consider $core(\psi)$ (rather than $core(chase_\Sigma(\psi))$). This is because $core(chase_\Sigma(\psi))$ is not necessarily a conjunctive query, but rather a union of conjunctive queries which we do not handle.*

▶ **Theorem 17.** *Given a set of cardinality dependencies Σ, for all connected $CQ^\#$ queries Q we have $GS^{\sim_1}_{\mathcal{C}_\Sigma}(Q) \leq \Delta^+_\Sigma(Q)$.*

▶ **Example 18** (Cont. from Ex. 16)**.** If every patient has at most 3 attending doctors, the sensitivity of $\#\varphi$ becomes bounded. Indeed, if $\Sigma = \{PatDoc[1 \xrightarrow{3} 2]\}$, then $\mathrm{GS}^{\mathcal{R}}_{\mathcal{C}_\Sigma}(\#\varphi) \leq \Delta^+_{\mathcal{R},\Sigma}(\#\varphi) = 3$ by Theorem 17.

6 Conclusion

We have given bounds for the global sensitivity of counting Conjunctive Queries under the functionality or cardinality constraints. These bounds can be used to turn those queries in differentially private ones by using mechanisms like the Laplacian or the Gaussian mechanisms without adding too much noise. The proposed algorithms for computing these bounds have exponential complexity, but since effectively many interesting queries are often small, our results are still practical.

There are several interesting directions that we will pursue in future work. We will study other aggregation operations already present in SQL such as *average* or *sum*. We will also investigate sensitivity of queries with negation, where one can ask for example for the number of patients that are *not* treated by a given doctor. Further, we have focused here on global sensitivity but there are other notions of sensitivity that have been proposed. In particular, the so-called *local sensitivity* is studied in [13]. The local sensitivity is defined by quantifying not over all possible databases but only over the ones in the neighborhood of the particular database under analysis. The local sensitivity is often lower than the global sensitivity, but adding noise proportional to the local sensitivity does not ensure differential privacy. Nevertheless, adding the noise proportional to a smooth approximation of the local sensitivity permits to recover differential privacy.

References

1 Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995. URL: `http://webdam.inria.fr/Alice/`.

2 A. V. Aho, C. Beeri, and J. D. Ullman. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 4(3):297–314, September 1979. `doi:10.1145/320083.320091`.

3 Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The sulq framework. In *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS'05, pages 128–138, New York, NY, USA, 2005. ACM. `doi:10.1145/1065167.1065184`.

4 Swarat Chaudhuri, Sumit Gulwani, Roberto Lublinerman, and Sara Navidpour. Proving programs robust. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE'11, pages 102–112, New York, NY, USA, 2011. ACM. `doi:10.1145/2025113.2025131`.

5 Shixi Chen and Shuigeng Zhou. Recursive mechanism: Towards node differential privacy and unrestricted joins. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD'13, pages 653–664, New York, NY, USA, 2013. ACM. `doi:10.1145/2463676.2465304`.

6 Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 1–12, 2006. doi:10.1007/11787006_1.

7 Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014. doi:10.1561/0400000042.

8 Marco Gaboardi, Andreas Haeberlen, Justin Hsu, Arjun Narayan, and Benjamin C. Pierce. Linear dependent types for differential privacy. *SIGPLAN Not.*, 48(1):357–370, January 2013. doi:10.1145/2480359.2429113.

9 Sven Hartmann. On interactions of cardinality constraints, key, and functional dependencies. In *Foundations of Information and Knowledge Systems, First International Symposium, FoIKS 2000, Burg, Germany, February 14-17, 2000, Proceedings*, pages 136–155, 2000. doi:10.1007/3-540-46564-2_9.

10 Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD'11, pages 193–204, New York, NY, USA, 2011. ACM. doi:10.1145/1989323.1989345.

11 David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, December 1979. doi:10.1145/320107.320115.

12 Frank D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD'09, pages 19–30, New York, NY, USA, 2009. ACM. doi:10.1145/1559845.1559850.

13 Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, STOC'07, pages 75–84, New York, NY, USA, 2007. ACM. doi:10.1145/1250790.1250803.

14 Catuscia Palamidessi and Marco Stronati. Differential privacy for relational algebra: Improving the sensitivity bounds via constraint systems. In *Proceedings 10th Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL 2012, Tallinn, Estonia, 31 March and 1 April 2012.*, pages 92–105, 2012. doi:10.4204/EPTCS.85.7.

15 Jason Reed and Benjamin C. Pierce. Distance makes the types grow stronger: A calculus for differential privacy. In *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*, ICFP'10, pages 157–168, New York, NY, USA, 2010. ACM. doi:10.1145/1863543.1863568.

The Complexity of Rational Synthesis

Rodica Condurache[1], Emmanuel Filiot[*2], Raffaella Gentilini[†3], and Jean-François Raskin[‡4]

1 **Université Libre de Bruxelles, Computer Science Department, Brussels, Belgium, and**
Université Paris Est, LACL(EA 4219), UPEC, Créteil Cedex, France
2 **Université Libre de Bruxelles, Computer Science Department, Brussels, Belgium**
3 **University of Perugia, Dept. of Mathematics and Computer Science, Perugia, Italy**
2 **Université Libre de Bruxelles, Computer Science Department, Brussels, Belgium**

Abstract

We study the computational complexity of the cooperative and non-cooperative rational synthesis problems, as introduced by Kupferman, Vardi and co-authors. We provide tight results for most of the classical omega-regular objectives, and show how to solve those problems optimally.

1998 ACM Subject Classification B.6.3 Automatic Synthesis, F.2 Analysis of algorithms and problem complexity, F.1.1 Automata

Keywords and phrases Non-zero sum games, reactive synthesis, omega-regular objectives

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.121

1 Introduction

Rational synthesis [14, 19] asks to synthesize a system that is executed in an environment made of several components that are assumed to be rational, and not fully antagonistic as in the classical two player zero-sum setting [23]. Rationality of the environment is modeled by assuming that the components behave according to a Nash equilibrium (NE). Rational synthesis has been introduced in [14, 19] in two different settings.

In the first setting, called *cooperative rational synthesis* [14], the environment cooperates with the system: components of the environment agree to play a NE that is winning for Player 0 (if it exists). In the second setting, called *non-cooperative rational synthesis* [19], the components of the environment may follow any strategy, provided that it is a NE. In this setting, one has to output (if it exists) a strategy σ_0 for the system which has to be winning against all the possible strategy profiles that include σ_0 for Player 0 and which are NE.

The main contribution of the original papers is to propose and to motivate the definitions above. The only computational complexity results given in those papers are as follows: the cooperative and non-cooperative rational synthesis problems are 2ExpTime-c for specifications expressed in linear temporal logic (LTL), thus matching exactly the complexity of

* E. Filiot is research associate at FNRS. This work was partially supported by ARC project *Transform* (French speaking community of Belgium) and Belgian FNRS PDR project *Flare*.
† R. Gentilini acknowledges the support of *Fondi Ricerca Base 2015* (Automi e Teoria dei Giochi per la Verifica) and InDAM-GNCS.
‡ J.-F. Raskin was supported by ERC Starting Grant inVEST (nr. 279499).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 121; pp. 121:1–121:15

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

Table 1 Complexity of rational synthesis for k players. Full proofs can be found in [12].

	Cooperative		Non-Cooperative	
	Unfixed k	Fixed k	Unfixed k	Fixed k
Safety	NP-c	PTime-c	PSpace-c	PTime-c
Reachability	NP-c	PTime-c	PSpace-c	PTime-c
Büchi	PTime-c[25]	PTime-c[25]	PSpace-c	PTime-c
coBüchi	NP-c[25]	PTime-c	PSpace-c	PTime-c
Parity	NP-c[25]	$UP \cap co-UP$, parity-h	ExpTime, PSpace-h	PSpace, NP-h, coNP-h
Streett	NP-c [25]	NP [25], NP-hard	ExpTime,PSpace-h	PSpace-c
Rabin	P^{NP}, NP-h, coNP-h	P^{NP}, coNP-h	ExpTime, PSpace-h	PSpace-c
Muller	PSpace-c	PSpace-c	ExpTime, PSpace-h	PSpace-c
LTL	2ExpTime-c[14]	2ExpTime-c[14]	2ExpTime-c[19]	2ExpTime-c[19]

classical zero-sum two-player LTL synthesis [21]. The upper bound is obtained by reductions to the satisfiability problem of formulas in Strategy Logic (SL) [20]. The reduction to SL and the use of LTL specifications does not allow one to understand finely the computational complexity aspects of solving the underlying n player non-zero sum games.

Contributions. To better understand the computational complexity of the rational synthesis problems and how to solve their underlying games algorithmically, we consider variants of those problems for games played on turn-based graph structures for reachability, safety, Büchi, coBüchi, parity, Rabin, Streett and Muller objectives for unfixed and fixed number of players. The fixed number of players case is interesting as the number of components forming the environment may be limited to a few in practical applications. Our results are summarized in Table 1.

On the positive side, our results show that for a *fixed* number of players, and for objectives that admit a polynomial time solution in the two-player zero-sum case (reachability, safety, Büchi and coBüchi), cooperative and non-cooperative rational synthesis can be solved in PTime. On the negative side, for rich omega regular objectives defined by parity, Rabin, or Streett objective, the complexity increases. First, games with parity objectives cannot be solved in polynomial time unless PTime equals NP while it is conjectured that this result does not hold for two-player zero sum parity games. Second, games with Rabin or Streett objectives are PSpace-c for the non-cooperative setting while they have solution in nondeterministic polynomial time for their zero-sum two player versions. When the number of players is not fixed, the complexity is usually substantially higher than for the two-player zero-sum case. For example, non-cooperative rational synthesis is PSpace-H for all objectives, so even for safety objectives.

Cooperative rational synthesis is a special case of constrained NE (Player 0 has to be winning). The complexity of constrained NE has been studied in [25] for some classes of objectives: this gives us upper-bounds for cooperative synthesis and Büchi, coBüchi, parity and Streett objectives. For the other objectives, we show how to extend the approach proposed in [25]. Solutions to the *non-cooperative* case are much more involved and based on a fine tuned application of tree automata techniques. This is a central contribution of our paper. In particular, our tree automata have exponential size but we show how to test their emptiness in PSpace to obtain optimal algorithms for Streett, Rabin and Muller objectives and fixed number of players.

The tree automata that we construct not only allow us to test the existence of solution to the non-cooperative rational synthesis problem but also to symbolically represent all the strategies for the system that are solutions. This set is thus regular and can be manipulated

with automata-based techniques. Also, it should be clear that our algorithms are amenable to symbolic implementations when the game structure is given with binary decision diagrams. This is important as it shows that our techniques pave the way to implementations that have proven useful and efficient by the CAV community (see e.g. tools like nuSMV [11]).

To obtain lower-bounds, we design original and informative reductions.

Related work. Non-zero sum games for synthesis are gaining attention recently, see e.g. [4] for a survey. *Secure equilibria* were introduced in [9] and their potential for synthesis was demonstrated in [8]. Secure equilibria is a refinement of NE [24]. *Doomsday equilibria* extend secure equilibria to the n player case [7]. Subgame perfect equilibria, that also refines NE, were first studied in [24, 25]. To model rationality of players, the notion of *admissible strategy* is used in [3, 13] instead of the notion of NE, and computational aspects are studied in [6], potential for synthesis is studied in [5]. All those works consider games played on a game structure with classical ω-regular objectives and provide tight complexity results for almost all the relevant synthesis problems. This is not the case for *cooperative* and *non-cooperative rational synthesis* for which only the complexity for specifications given in LTL was known [14, 19]. This paper provides algorithms and precise computational complexity results.

Structure of the paper. In Sect. 2, we recall the definition of the cooperative and non-cooperative synthesis problem as introduced in [14, 19], together with the game structure variant and objectives that we study here. Sect. 3 provides lower and upper complexity bounds for the cooperative rational synthesis problem. Sect. 4 provides results for the non-cooperative variant. Sect. 5 summarizes complexity results when the number of players is fixed. Due to the lack of space, we provide sketches of proof of our results in this paper and all the detailed proofs can be downloaded at the following address: [12].

2 Multiplayer Games and Rational Synthesis

Multiplayer Games. Let $k \in \mathbb{N}$. A *multiplayer arena* $(k+1)$-players arena is a tuple $\mathcal{A} = \langle \Omega, V, (V_i)_{i\in\Omega}, E, v_0 \rangle$, where $\Omega = \{0, 1, ..., k\}$ is a finite set of players, (V, E) is a finite directed graph whose vertices are called *states*, $v_0 \in V$ is the initial state and $(V_i)_{i\in\Omega}$ is a partition of V where V_i is the set of states controlled by Player $i \in \Omega$. A *play* in $\mathcal{A}$ starts in the initial state v_0 and proceeds in rounds. At each round, the player controlling the current state chooses the next position according to $E \subseteq \bigcup_{i\in\Omega} V_i \times V_{i+1 \bmod k}$.[1]. Formally, a play $\pi = v_0 v_1 \dots$ is an infinite path in V^ω such that v_0 is the initial state and $(v_i, v_{i+1}) \in E$ for each $i \geqslant 0$. The prefix (or history) of π up to v_n is written $\pi[:n]$ and its last state $\pi(n)$. We denote by $\sqsubset$ the prefix relation, by $\mathsf{Plays}(\mathcal{A})$ the set of plays, and by $\mathsf{Prefs}(\mathcal{A})$ for its set of finite prefixes. For $\pi \in V^\omega$, $\inf(\pi)$ is the set of states occurring infinitely many times in π.

A *strategy* of Player $i \in \Omega$ in $\mathcal{A}$ is a total function $\sigma_i : V^*V_i \mapsto V$ s.t. for all $x \in V^*$, for all $v \in V_i$, $(v, \sigma_i(xv)) \in E$. Note that as rounds are ordered, σ_i has type $V^*V_i \mapsto V_{i+1 \bmod k}$. A play π is *consistent* with σ_i if $\pi(n+1) = \sigma_i(\pi[:n])$ for all $n \geqslant 0$ s.t. $\pi(n) \in V_i$. The *outcome* of σ_i is the set of plays $out(\sigma_i) \subseteq \mathsf{Plays}(\mathcal{A})$ that are consistent with σ_i. Given $h \in V^*$, we define $\sigma_i|_h$ as $\sigma_i|_h(h') = \sigma_i(hh')$ for all $h' \in V^*V_i$. A *winning objective* (or just objective) is

[1] Wlog. we assume that each vertex has a successor by E and that player's rounds are ordered according to their index. Otherwise we just add a polynomial number of extra intermediate states and the winning objectives considered in this paper can be modified accordingly.

a set $\mathcal{O} \subseteq V^\omega$. It is *tail* if it is closed under removing prefixes. A Player i's strategy σ_i is *winning*[2] for $\mathcal{O}$ if $out(\sigma_i) \subseteq \mathcal{O}$. We consider the classical classes of objectives [17]:

- *Safety/Reachability*: Given a set $S \subseteq V$ of safe states, $\mathsf{Safe}(S) = \{\pi \in V^\omega \mid \forall n \geqslant 0 : \pi(n) \in S\}$ and given a set T of target states, $\mathsf{Reach}(T) = \{\pi \in V^\omega \mid \exists n \geqslant 0 : \pi(n) \in T\} = \overline{\mathsf{Safe}(\overline{T})}$.
- *Büchi/coBüchi*: Given a set $F \subseteq V$, $\mathsf{Buchi}(F) = \{\pi \in V^\omega \mid \inf(\pi) \cap F \neq \varnothing\}$ and $\mathsf{coBuchi}(F) = \{\pi \in V^\omega \mid \inf(\pi) \cap F = \varnothing\} = \overline{\mathsf{Buchi}(F)}$.
- *Streett/Rabin*: Given a set $\Psi \subseteq 2^V \times 2^V$, $\mathsf{Streett}(\Psi) = \bigcap_{(L,R)\in\Psi}(\mathsf{coBuchi}(L) \cup \mathsf{Buchi}(R))$ and $\mathsf{Rabin}(\Psi) = \bigcup_{(L,R)\in\Psi}(\mathsf{Buchi}(L) \cap \mathsf{coBuchi}(R)) = \overline{\mathsf{Streett}(\Psi)}$.
- *Parity*: For a priority mapping $p{:}V \to \mathbb{N}$, $\mathsf{Parity}(p) = \{\pi \in V^\omega \mid \min\{p(v) | v \in \inf(\pi)\} \text{ is even}\}$.
- *Muller*: Given a Boolean formula μ over V, $\mathsf{Muller}(\mu) = \{\pi \in V^\omega \mid \inf(\pi) \models \mu\}$.

A *multiplayer game* is a pair $\mathcal{G} = \langle \mathcal{A}, (\mathcal{O}_i)_{i\in\Omega} \rangle$ where $(\mathcal{O}_i)_{i\in\Omega}$ is the tuple of objectives for each Player $i \in \Omega$. For all class of objectives X, we say that $\mathcal{G}$ is a multiplayer X-game if all objectives $\mathcal{O}_i$ are in X. The notations Plays and Prefs carries over naturally to $\mathcal{G}$ by considering its underlying arena. For $v \in V$, one denotes by $\mathcal{G}[v]$ the game $\mathcal{G}$ whose initial state is replaced by v (winning objectives are unchanged). A state $v \in V$ is *winning* for Player i if he has a winning strategy in $\mathcal{G}[v]$, and one denotes by $W_i^{\mathcal{G}}$ (or just W_i) the set of *winning states* of Player i, also called the *winning set* of Player i.

Nash Equilibria. A *strategy profile* $\bar{\sigma}$ in a multiplayer game $\mathcal{G} = \langle \mathcal{A}, (\mathcal{O}_i)_{i\in\Omega} \rangle$ is a tuple $\bar{\sigma} = (\sigma_i)_{i\in\Omega}$ of strategies, one for each player. The *outcome* $out(\bar{\sigma})$ of $\bar{\sigma}$ is the only play consistent with all strategies σ_i (it always exists and is unique). Given a strategy τ for Player i, we write $(\bar{\sigma}_{-i}, \tau)$ for the strategy profile obtained by replacing σ_i with τ in $\bar{\sigma}$. For winning objectives $(\mathcal{O}_i)_{i\in\Omega}$ for each player, the *payoff* of a strategy profile $\bar{\sigma}$ is the vector $pay(\bar{\sigma}) \in \{0,1\}^{\{k+1\}}$ defined by $pay(\bar{\sigma})[i] = 1$ iff $out(\bar{\sigma}) \in \mathcal{O}_i$. We write $pay_i(\bar{\sigma})$ for Player i's payoff in $pay(\bar{\sigma})$. Payoffs are compared by the pairwise natural order on their bits, denoted by $\leqslant$, i.e. $pay(\bar{\sigma}) \leqslant pay(\bar{\beta})$ if $pay_i(\bar{\sigma}) \leqslant pay_i(\bar{\beta})$ for all $i \in \Omega$.

A strategy profile $\bar{\sigma} = (\sigma_i)_{i\in\Omega}$ is called a *Nash equilibrium* (NE) if no player can improve his payoff by (unilaterally) switching to a different strategy, i.e. for all players $i \in \Omega$ and all strategies τ of Player i, $pay(\bar{\sigma}_{-i}, \tau) \leqslant pay(\bar{\sigma})$. We say that $\bar{\sigma}$ is a 0-*fixed* Nash equilibrium (0NE) if $pay(\bar{\sigma}_{-i}, \tau) \leqslant pay(\bar{\sigma})$ for all players $i \in \Omega \backslash \{0\}$ and all strategies τ of i. In other words, it is a Nash equilibrium in which Player 0 is not allowed to deviate. Any NE is 0-fixed, but the converse may not hold.

Cooperative and non-cooperative rational synthesis. Rational synthesis aims at finding a winning strategy for the system (Player 0) against an environment composed of several components (Players 1 to k) that are assumed to play rationally. Rationality of the environment is modeled by NE, and following [14, 19], we consider two settings, depending on whether the environment cooperates or not: The cooperative and non-cooperative rational synthesis problems (CRSP and NCRSP resp.) ask, given as input a $(k+1)$-player game $\mathcal{G}$ with winning objectives $(\mathcal{O}_i)_{i\in\Omega}$, the following questions according to the two settings:

cooperative: Is there a 0-fixed Nash equilibrium $\bar{\sigma}$ such that $pay_0(\bar{\sigma}) = 1$?

non-cooperative: Is there a strategy σ_0 for Player 0 such that for every 0-fixed Nash equilibrium $\bar{\sigma} = \langle \sigma_0, \ldots, \sigma_k \rangle$, we have $pay_0(\bar{\sigma}) = 1$?

[2] Here we implicitly consider a two-player zero-sum game in which Player i has objective $\mathcal{O}$ and plays against all the other players in $\Omega \backslash \{i\}$ who have objective $\overline{\mathcal{O}}$.

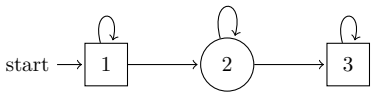


Figure 1 Example for rational synthesis.

▶ **Example 1.** Consider Figure 1 in which Player 0 owns round states and Player 1 square states, with the reachability objectives $R_0 = \{2\}$ and $R_1 = \{3\}$. Consider the Player 0's strategies σ_0 which loops forever in state 2, and σ_0' which eventually moves to state 3.

Let Player 1 cooperate with σ_1 that moves to state 2 (making Player 0 win). Both strategy profiles $\langle \sigma_0, \sigma_1 \rangle$ and $\langle \sigma_0', \sigma_1 \rangle$ are solutions to the cooperative setting: for the first strategy profile Player 1 loses but cannot get better payoff by deviating, and for the later one Player 1 wins. Strategy σ_0 is not a solution to the non-cooperative setting: Player 1 could stay forever in state 1 (according to a strategy σ_1'). The profile $\langle \sigma_0, \sigma_1' \rangle$ is a 0-fixed NE because even by deviating and going to state 2 Player 1 would still lose, and it is losing for Player 0. However, σ_0' is a solution to the non-cooperative setting: The only 0-fixed NE in that case are when Player 1 eventually move to state 2, making him and Player 0 win.

In [14, 19], both CRSP and NCRSP are shown 2EXPTIME-COMPLETE when the winning objectives are defined by LTL formulas, through a reduction to strategy logic. In this paper, we refine this complexity result for particular kinds of winning objectives. In general, the synthesis problem also asks to *synthesise* (i.e. construct) a solution when it exists: Our algorithms also solve the synthesis problem.

3 Cooperative Rational Synthesis Problem (CRSP)

We establish here complexity bounds for CRSP for unfixed number of players. First, some results are obtained as special cases of constrained NE problems [25]. The constrained NE problem asks to decide, given a $k+1$-player game $\mathcal{G} = \langle \mathcal{A}, (\mathcal{O}_i)_{i \in \Omega} \rangle$, and for each player i, a lower bound $l_i \in \{0,1\}$ and an upper bound $u_i \in \{0,1\}$ such that $l_i \leqslant u_i$, whether there exists a NE $\overline{\sigma}$ s.t. $l_i \leqslant pay_i(\overline{\sigma}) \leqslant u_i$ for all $i \in \Omega$. CSRP is a special case of this problem, by setting $l_0 = u_0 = 1$, $l_i = 0$ and $u_i = 1$ for all $i \in \Omega \backslash \{0\}$. The constrained NE problem is known to be in PTIME for Büchi objectives, and in NP for co-Büchi, Streett and parity objectives [25]. So one immediately gets the upper bounds of Table 1 for these measures (and unfixed k).

To establish the remaining upper bounds, we characterize NE by means of LTL formulas. We extend the technique used in [25] for tail objectives to safety and reachability.

Generic solution to cooperative rational synthesis. Syntax and semantics of LTL can be found in [2]. Let V be the set of vertices of $\mathcal{G}$. We use LTL formulas to express properties of infinite paths of $\mathcal{G}$, where we take V as set of atomic propositions. In particular, $s \in V$ is true in s, and false otherwise. For $S \subseteq V$, the formula S is a shortcut for $\bigvee_{s \in S} s$, and LTL($\mathcal{G}$) denotes the set of LTL formulas over V. Let $(W_i^{\mathcal{G}})_{0 \leqslant i \leqslant k}$ be the winning sets of each player and $b \in \{0,1\}$. We define an LTL[$\mathcal{G}$]-formula $\phi_{b\text{Nash}}^{\mathcal{G}}$ that will characterize NE ($b = 1$) and 0-NE ($b = 0$):

$$\phi_{b\text{Nash}}^{\mathcal{G}} = \begin{cases} \bigwedge_{i=1-b}^{k} ((\neg W_i^{\mathcal{G}} \,\mathcal{U}\, \neg S_i) \vee \Box S_i) & \text{if } \mathcal{O}_i \text{ are safety objectives of the form} \\ & \mathcal{O}_i = \mathsf{Safe}(S_i) \text{ for } S_i \subseteq V \\ \bigwedge_{i=1-b}^{k} \neg \varphi_i \rightarrow \Box \neg W_i^{\mathcal{G}} & \text{if } \mathcal{O}_i \text{ are either all reachability or all tail} \\ & \text{objectives definable by a LTL}[\mathcal{G}] \text{ formula } \varphi_i \end{cases}$$

Assume that $b = 1$, and consider the formula for safety objectives. Intuitively, it says that for all players $i \in \{0, \dots, k\}$, either Player i always stays safe, or if eventually he visits an unsafe position, then he should never visit a winning position until he meets an unsafe position for the first time. This is because otherwise he could apply a winning strategy and satisfy his own objective, and therefore has some incentive to deviate. As announced:

▶ **Proposition 2** (Characterization of 0-fixed NE and NE ([25] for tail objectives)). *Let $\mathcal{G}$ be a multiplayer game with either all safety, all reachability, or all tail objectives, definable in LTL[$\mathcal{G}$]. Then, the following hold:*

1. *For all $\pi \in \mathit{Plays}(\mathcal{G})$, if $\pi \models \phi^{\mathcal{G}}_{0Nash}$ (resp. $\pi \models \phi^{\mathcal{G}}_{1Nash}$), then there exists a 0-fixed NE (resp. NE) $\bar{\sigma}$ in $\mathcal{G}$ such that $out(\bar{\sigma}) = \pi$,*
2. *For all 0-fixed NE (resp. NE) $\bar{\sigma}$ in $\mathcal{G}$, $out(\bar{\sigma}) \models \phi^{\mathcal{G}}_{0Nash}$ (resp. $out(\bar{\sigma}) \models \phi^{\mathcal{G}}_{1Nash}$).*

Based on the latter proposition, it is not difficult to design a procedure to decide CRSP: first, compute the winning sets W_i, and then check whether the game $\mathcal{G}$ contains a path which satisfies the formula $\phi^{\mathcal{G}}_{0\text{Nash}} \wedge \phi_0$, where ϕ_0 is the objective of Player 0, expressed in LTL[$\mathcal{G}$]. To establish precise upper bounds, one needs to consider the complexity of computing the winning sets, and then the complexity of model-checking these particular LTL formulas. Due to lack of space, we cannot cover all the cases, but let us briefly expose the case of safety conditions. For safety, it is well-known that computing the sets W_i can be done in PTime. Then, we prove a short witness property: if there exists a path satisfying the formula $\phi^{\mathcal{G}}_{0\text{Nash}} \wedge \phi_0$, then there exists a lasso path uv^ω such that u and v have polynomial length. Therefore, it suffices to guess such a lasso path, and to check that it satisfies the formula $\phi^{\mathcal{G}}_{0\text{Nash}} \wedge \phi_0$, which again can be done in PTime. This yields an NP algorithm for safety CRSP. Similar arguments apply for reachability.

Lower bounds. In [25] was shown the NP-hardness of the threshold NE problem for coBüchi objectives and thresholds $l_0 = u_0 = 1$, $l_i = 0$ and $u_i = 1$, $\forall i \in \{1, \dots, k\}$. Therefore, one immediately gets the NP lower bounds of Table 1 for coBüchi objectives, Streett, Rabin and parity objectives, which can (polynomially) express coBüchi objectives. For the other cases, we provide lower bounds with genuine reductions. For Muller objectives, we show that it is already PSpace-hard even for two players and the precise complexity results for a fixed number of players are discussed in the last section of the paper. We finish the section by showing the NP-hardness proof for the safety case (which was not considered in [25]).

▶ **Lemma 3.** *CRSP for multiplayer safety objectives is NP-hard.*

Proof. By reduction from $3SAT$. Given a set of clauses $S = \{C_1, \dots, C_k\}$, one constructs the $(k+1)$-player safety game of Fig. 2. Up to vertex C_1, the previous states are controlled by Player 0, and each state C_i is controlled by Player i. All states but U_s are safe for Player 0. For all $i \in \{1, \dots, k\}$, U_e is unsafe for Player i, as well as the state 0_x if $\neg x$ appears in C_j, and the state 1_x if x appears in C_j. If S is satisfiable by some valuation ν, then Player 0 chooses, in state x, the successor $\nu(x)_x$. That way, Player 1 to k have visited an unsafe state before reaching C_1, and have no incentive to deviate if their strategy is to go to U_e. Conversely, if there is a solution to CRSP, necessarily the game end up in U_e, as Player 0 must be winning. It means that before reaching the states C_i all safety conditions for Players 1 to k have been violated, otherwise going to U_s could be a profitable deviation. In other words, for all clauses i, there exists a literal ℓ in C_i such that 1_x (resp. 0_x) has been visited if $\ell = x$ (resp. if $\ell = \neg x$). So the truth values chosen by Player 0 satisfies S.

◀

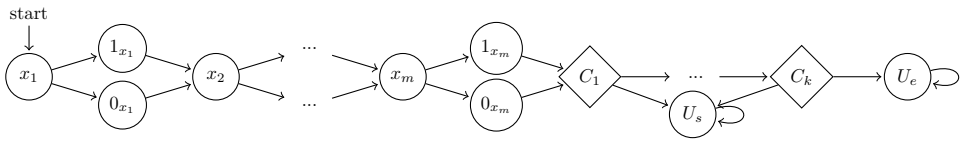


Figure 2 Cooperative Safety: Reduction from 3-SAT.

4 Non-Cooperative Rational Synthesis Problem (NCRSP)

We study here the complexity of NCRSP for unfixed number of players. In this setting, cooperation of the environment is not assumed, and so the system has to win against all 0-fixed NE. In Prop. 2, we have characterised 0-fixed NE by means of an LTL formula $\phi^{\mathcal{G}}_{0\text{Nash}}$. This allowed us to solve CRSP via a reduction to model-checking. It is tempting to think that NCRSP reduces to a two-player zero-sum game between Player 0, whose objective is $\phi^{\mathcal{G}}_{0\text{Nash}} \rightarrow \varphi_0$, and the coalition of the other players. However, Example 1 shows that this is not true in general. Indeed, in this example there is a solution to NCRSP, but no solution to the two-player game with objective $(\Box\bar{R}_1 \rightarrow \Box\bar{W}_1) \rightarrow \Diamond R_0$. Since $W_1 = \{3\}$, whatever the strategy of Player 0 is, if Player 1 stays in state 1 forever, the path $\pi = (1)^\omega$ satisfies $(\Box\bar{R}_1 \rightarrow \Box\bar{W}_1)$ but not $\Diamond R_0$ and therefore Player 0 loses. The intrinsic reason why the reduction to two-player games is incorrect lies in the definition of NCRSP: once a Player 0's strategy σ_0 is fixed, only 0-fixed NE with respect to σ_0 are considered, while the formula $\phi_{0\text{Nash}}$ can be satisfied by paths which are outcomes of other 0-fixed NE, i.e. for a different strategy of Player 0.

The non-cooperative case is more involved and requires tree automata based techniques: we see strategies σ_0 as trees t_{σ_0}, and use *tree automata* to define the set of solutions to NCRSP. Testing existence of a solution then reduces to tree automata non-emptiness.

Strategy trees and good deviations. Let Λ be a finite set of directions and Σ be an alphabet. A Σ-labeled Λ-tree is a mapping $t : \Lambda^* \rightarrow \Sigma$. Its set of nodes is Λ^*. Let $\mathcal{A}$ be a $k+1$-player arena with set of states V, and let $\sigma_0 : V^*V_0 \rightarrow V$ be a strategy of Player 0 in $\mathcal{A}$. We explain how σ_0 is encoded as a tree. The labels are in the set $\Sigma = V \cup \{*_i \mid 1 \leqslant i \leqslant k\} \cup \{\#\}$, and the set of directions is $\Lambda = V$. Therefore, any node of the tree is a history h. Then, if $h = \epsilon$ (root node), we set its label to $\#$. Otherwise, it is of the form $h = h'v$, then there are two cases: (i) if $v \in V_0$, then $t_{\sigma_0}(h) = \sigma_0(h)$, (ii) if $v \in V_i$ for $i \neq 0$, then $t_{\sigma_0}(h) = *_i$ (only the turn i is encoded). We denote by T_0 the set of strategy trees t_{σ_0}. Note that each Σ-labeled V-tree represents a partial function from V^* to Σ, which may not be a strategy, because it is not total and may not be consistent with the edge relation E of the arena. A *branch* in t_{σ_0} is an infinite sequence of directions $\pi \in V^\omega$. It is *compatible* with σ_0 if for all finite prefixes h of π whose last state is in V_0, $h.t_{\sigma_0}(h)$ is a prefix of π.

We now want to characterize the strategy trees t_{σ_0} s.t. σ_0 is a solution to NCRSP in a game $\mathcal{G} = \langle \mathcal{A}, (\mathcal{O}_i)_{i\in\Omega} \rangle$ with either all safety, all reachability, or all tail objectives. Consider a branch π of t_{σ_0} compatible with σ_0: It is not the outcome of a σ_0-fixed NE iff some player loses ($\pi \notin \mathcal{O}_i$ for some $i \neq 0$) and there is a prefix h from which Player i has a winning strategy against all other players (and the strategy σ_0). We call the history h a good deviation point. Formally, h is a *good deviation point* if there exists $i \in \{1, \dots, k\} = \Omega\backslash\{0\}$ s.t. $\pi \notin \mathcal{O}_i$ and there is a strategy σ_i for Player i s.t. for all strategies $(\sigma_j)_{i\in\{1,\dots,k\}\backslash\{i\}}$, $h.out(\sigma_0|_h, ..., \sigma_{i-1}|_h, \sigma_i|_h, \sigma_{i+1}|_h, ..., \sigma_k|_h) \in \mathcal{O}_i$. A branch $\pi \in V^\omega$ has a *good deviation* if some of its prefix h is a good deviation point. Let us denote by $\mathsf{NCRSP}(\mathcal{G})$ the set of strategy trees t_{σ_0} such that σ_0 is a solution to the NCRSP in $\mathcal{G}$. Then:

▶ **Lemma 4.** *For all strategies σ_0 of Player 0, $t_{\sigma_0} \in \mathsf{NCRSP}(\mathcal{G})$ iff for all branches π of t_{σ_0} compatible with σ_0, either $\pi \in \mathcal{O}_0$ or π has a good deviation.*

Reduction to tree-automata emptiness. Based on Lemma 4, we construct a non-deterministic automaton defining $\mathsf{NCRSP}(\mathcal{G})$. A *non-deterministic tree automaton* $\mathcal{T}$ over Σ-labeled Λ-trees is a tuple (Q, Q_0, δ) where Q is a finite set of states, $Q_0 \subseteq Q$ is a finite set of initial states, and δ is a transition relation of the form $\delta : Q \times \Sigma \to 2^{\Lambda \to Q}$, i.e., it maps any pair of states and labels to a set of mappings from directions to states (states sent the children of the current node). A *run* of $\mathcal{T}$ on a tree t is Q-labeled Λ-tree $r : \Lambda^* \to Q$ such that $r(\epsilon) \in Q_0$ and for all $h \in \Lambda^*$, all $d \in \Lambda$, the mapping $d \in \Lambda \mapsto r(hd) \in Q$ is in $\delta(r(h), t(h))$. The image of a branch $\pi = \lambda_1\lambda_2 \cdots \in \Lambda^\omega$ by r is the word in Q^ω defined by $r(\epsilon)r(\lambda_1)r(\lambda_1\lambda_2)\ldots$. With respect to an accepting condition $\alpha \subseteq Q^\omega$, r is accepting if the images of all its branches are in α, and the *language* of $\mathcal{T}$ is the set $\mathcal{L}_\alpha(T)$ of trees for which there exists an accepting run.

▶ **Lemma 5.** *Let $\mathcal{G} = \langle \mathcal{A}, \mathcal{O} = (\mathcal{O}_i)_{i\in\Omega}\rangle$ be a $(k+1)$-player game with n vertices. One can construct a non-deterministic tree automaton $\mathcal{T}_\mathcal{A}$ (with an exponential number of states in k, and polynomial in $|V|$) with an accepting condition $\alpha_\mathcal{A}(\mathcal{O})$ such that $\mathcal{L}_{\alpha_\mathcal{A}(\mathcal{O})}(\mathcal{T}_\mathcal{A}) = \mathsf{NCRSP}(\mathcal{G})$. Moreover, for all runs r of $\mathcal{T}_\mathcal{A}$, for all branches π of r, the number of states appearing in π is polynomial in $|V|$ and k.*

Proof. We sketch the construction of $\mathcal{T}_\mathcal{A}$ and the definition of $\alpha_\mathcal{A}(\mathcal{O})$. First, it is not difficult to make sure that $\mathcal{T}_\mathcal{A}$ only accepts trees that are strategy trees: It has to remember the last direction v taken and make sure that if $v \in V_0$, the current node is labeled by some $v' \in V$ s.t. $(v, v') \in E$, and otherwise by the symbol $*_i$ if $v \in V_i$. This requires only a polynomial number of states. Therefore in the following, we assume that $\mathcal{T}_\mathcal{A}$ only runs on proper tree encodings t_{σ_0} of strategies σ_0.

The construction of $\mathcal{T}_\mathcal{A}$ is based on Lemma 4: For each branch of t_{σ_0}, it will check that either it is not compatible with σ_0, or it belongs to $\mathcal{O}_0$, or it will guess a prefix and check it is a good deviation. To guess good deviations, $\mathcal{T}_\mathcal{A}$ has to guess subtrees in which players have a winning strategy. This information is stored in a set $W \subseteq \Omega$, with the following semantics: if $\mathcal{T}_\mathcal{A}$ is in some state with set W at some node $h \in V^*$ and $i \in W$, then Player i has a winning strategy from h against σ_0 and any strategy of the players in $\Omega\backslash\{0, i\}$ for objective $\mathcal{O}_i$. The set of players for which a good deviation has been guessed is stored in a set $D \subseteq \Omega$, with the following semantics: if $\mathcal{T}_\mathcal{A}$ is in some state with set D and $i \in D$, at some node $h \in V^*$, then some prefix of h is a good deviation. The information on W is maintained as follows: at some node $hv \in V^*$, if $i \in W$ and $v \in V_i$, then $\mathcal{T}_\mathcal{A}$ non-deterministically send W to one of the successor of v (and $W\backslash\{i\}$ in the other ones) and if $v \notin V_i$, $\mathcal{T}_\mathcal{A}$ sends W in all successors of v. The information D is monotonic: either the current node h (owned by Player i) is not guessed to be a good deviation for any player and D is sent to all successors, or it is guessed to be a good deviation for Player i, $i \notin D$, and then $D \cup \{i\}$ (and W) is sent to all successors but one in which is sent D and $W \cup \{i\}$. This monotonic behavior is crucial for obtaining algorithms with optimal worst-case complexities.

Formally, $\mathcal{T}_\mathcal{A} = (Q, \{q_0\}, \delta)$ with $Q = \{q_0, \top\} \cup (2^\Omega \times 2^\Omega \times V)$. Then, we have $\delta(q_0, \#) = \{\rho_0\}$ where $\rho_0(v_0) = (\varnothing, \varnothing, v_0)$ and $\rho_0(v) = \bot$ for all $v \neq v_0$. For all $q = (W, D, v) \in Q$ such that $v \in V_0$ and all $v' \in V$, $\delta(q, v') = \{\rho_{v'}\}$ where $\rho_{v'}(v') = (W, D, v')$ and $\rho_{v'}(v'') = \bot$ for all $v'' \neq v'$ (the latter case correspond to directions v'' which are not compatible with the strategy). For all $v \neq \#$, $\delta(\top, v) = \{\rho_\top\}$ where $\rho_\top(v') = \top$ for all $v' \in V$. Then, for a state $q = (W, D, v)$ and a label $*_i$ ($i \neq 0$), we consider four cases:

1. $i \in D \cap W$: Such a state will never be reachable by construction.
2. $i \in D \cap \overline{W}$: In this case, we just propagate the information D and W. I.e. $\delta(q, *_i) = \{\rho\}$ s.t. $\rho(v') = (W, D, v')$ for all $(v, v') \in E$.
3. $i \in W \cap \overline{D}$: In this case, one has to check that Player i has a winning strategy in some successor of v', which is guessed non-deterministically, and to which the W information is sent. I.e. $\delta(q, *_i) = \{\rho_{v'} \mid (v, v') \in E\}$ where $\rho_{v'}(v') = (W, D, v')$ and $\rho_{v'}(v'') = (W \backslash \{i\}, D \cup \{i\}, v')$ for all $v'' \neq v'$.
4. $i \in \overline{W} \cap \overline{D}$: In this case, either we do not guess anything, or we guess that Player i has a good deviation, and update the sets W and D accordingly. I.e. $\delta(q, *_i) = \{\rho\} \cup \{\gamma_{v'} \mid (v, v') \in E\}$ where $\gamma_{v'}(v') = (W \cup \{i\}, D, v')$ and $\gamma_{v'}(v'') = (W, D \cup \{i\}, v')$ for all $v'' \neq v'$.

Along a path of a run of $\mathcal{T}_{\mathcal{A}}$, there are monotonicity properties for the W and D-components of the states. Indeed, by construction, $\mathcal{T}_{\mathcal{A}}$ never removes a player from D. For W, a player i can be removed (case 3) but then it is added to D and, once a player belongs to D, it can never be added to W again. It is correct since for a history h, if one guesses that Player i has a winning strategy from history hv, then i is added to D for all successors hv' ($v' \neq v$) and there is no need to guess again later on a good deviation for Player i in the subtrees rooted at the nodes hv', and therefore no need to add i in W again. Therefore along a path η of a run, there is only a polynomial number of different components D and W, and they necessarily stabilize eventually, to a set that we denote by $\lim_D(\eta)$ and $\lim_W(\eta)$.

Finally, the accepting condition $\alpha_{\mathcal{A}}(\mathcal{O})$ asks that on each path of the accepting run, either it is of the form $Q^*\{\bot\}^\omega$, or Player 0 wins, or there is a player that loses but belongs to some D eventually. For safety objectives, we also have to add the constraint that the losing player belongs to D before visiting an unsafe state. Additionally, the accepting condition also expresses constraints on the W components: each player $i \in \lim_W(\eta)$ wins. Formally, if we denote by $\mathrm{IRuns}(\mathcal{T}_{\mathcal{A}})$ the set of images of branches of runs of $\mathcal{T}_{\mathcal{A}}$, and by $\eta|_V$ the V-projection of any $\eta \in (Q\backslash\{\bot\})^\omega$, we have:

$$\alpha_{\mathcal{A}}(\mathcal{O}) = Q^*\{\top\}^\omega \cup (\{\eta \in \mathrm{IRuns}(\mathcal{T}_{\mathcal{A}}) \cap (Q\backslash\{\top\})^\omega \mid \eta|_V \in \mathcal{O}_0 \vee \bigvee_{i=1}^{k} \left(\eta|_V \notin \mathcal{O}_i \wedge \varphi_{\exists dev}(i, \eta)\right)\} \cap$$
$$\cap \{\eta \in \mathrm{IRuns}(\mathcal{T}_{\mathcal{A}}) \cap (Q\backslash\{\top\})^\omega \mid \bigwedge_{i \in \lim_W(\eta)} \eta|_V \in \mathcal{O}_i\})$$

where the formula $\varphi_{\exists dev}(i, \eta)$ says that there is a good deviation for Player i. That is, if $D_0 D_1 \ldots$ is the sequence of D-components in η, then $\varphi_{\exists dev}(i, \eta) = i \in \lim_D(\eta)$ for tail or reachability objectives, and $\varphi_{\exists dev}(i, \eta) = \exists p \geqslant 0, i \in D_p \wedge \forall p' \leqslant p, \eta|_V[p] \in S_i$ for safety conditions $\mathsf{Safe}(S_i)$. Details can be found in the technical report. ◀

Tree automata emptiness. We now study the complexity of testing non-emptiness of the languages $\mathcal{L}_{\alpha_{\mathcal{A}}(\mathcal{O})}(\mathcal{T}_{\mathcal{A}})$ for the objectives $\mathcal{O}$ of this paper. Classically, non-deterministic tree automata emptiness is reduced to solving a two-player zero sum game between Eve, who constructs a tree and a run on this tree, and Adam, whose goal is to prove that the run is non-accepting, by choosing directions in the tree and falsifying the acceptance condition. Formally, remind that the alphabet is $\Sigma = V \cup \{*_i \mid 1 \leqslant i \leqslant k\} \cup \{\bot\}$ and for a function $f : V \to Q$, we denote by $\mathsf{Range}(f)$ its range. We construct a zero-sum two-player game $\mathcal{G}' = \langle V_E, V_A, E', q_0 \rangle$ where $V_E = Q$, $V_A = \{\mathsf{Range}(f) \mid \exists q \in Q, \alpha \in \Sigma, f \in \delta(q, \alpha)\}$. Then, the transition relations is defined for all $q \in Q$, all $P \in V_A$, by $(q, P) \in E'$ if there exists $\alpha \in \Sigma$ and $f \in \delta(q, \alpha)$ s.t. $P = \mathsf{Range}(f)$, and $(P, q) \in E'$ if $q \in P$. That is, to go from q to P, Eve chooses a symbol α and a function $f : V \to Q$ in $\delta(q, \alpha)$. Then, Adam chooses a direction in V, but since he wants to construct a sequence of states not in $\alpha_{\mathcal{A}}(\mathcal{O})$, one only

needs to remember $\mathsf{Range}(f)$. Adam then picks a state in that set. Eve's objective is the set $\{q_1w_1q_2w_2\cdots \in (V_EV_A)^\omega \mid q_1q_2\cdots \in \alpha_{\mathcal{A}}(\mathcal{O})\}$. It can be shown that Eve has a winning strategy in $\mathcal{G}'$ iff $\mathcal{L}_{\alpha_{\mathcal{A}}(\mathcal{O})}(\mathcal{T}_{\mathcal{A}}) \neq \varnothing$. By a fine analysis of solving this game, we obtain:

▶ **Proposition 6.** *Let $\mathcal{G} = \langle \mathcal{A}, \mathcal{O} = (\mathcal{O}_i)_{i\in\Omega}\rangle$ be a multiplayer game. Non-emptiness of $\mathcal{L}_{\alpha_{\mathcal{A}}(\mathcal{O})}(\mathcal{T}_{\mathcal{A}})$ can be checked in* PSpace *for $\mathcal{O} \in \{\mathsf{Safety}, \mathsf{Reachability}, \mathsf{Büchi}, \mathsf{coBüchi}\}$, and in* ExpTime *for $\mathcal{O} \in \{\mathsf{Parity}, \mathsf{Streett}, \mathsf{Rabin}, \mathsf{Muller}\}$.*

Proof. It amounts to study the complexity of solving $\mathcal{G}'$ for the objectives of this paper. Note that the arena of $\mathcal{G}'$ has linear size in the size of $\mathcal{T}_{\mathcal{A}}$. For safety, reachability, Büchi and coBüchi winning objectives, we reduce the problem of solving $\mathcal{G}'$ to a finite duration reachability tree game of exponential size, whose duration is polynomial (in the size of the original arena of $\mathcal{G}$). This reduction exploits the monotonicity of the sets W and D, and the fact that the game can be stopped once a cycle has been formed. It is similar in spirit to the technique of *first-cycle game* from [1] but the winning condition of $\mathcal{G}'$ do not fall in the general hypothesis of [1] under which infinite duration games reduce to first-cycle games. Our finite duration tree game, though of exponential size, is not constructed explicitly but solved on-the-fly by a PTime alternating algorithm. This gives a PSpace upper-bound for NCRSP.

We now study the complexity of solving $\mathcal{G}'$ when the original game $\mathcal{G}$ has Muller conditions $\mathsf{Muller}(\mu_i)$ for the $k+1$ players. We transform $\mathcal{G}'$ into a two-player zero-sum parity game with an exponential number of states but a polynomial number of priorities, which can be solved in ExpTime (in the size of $\mathcal{G}$). This reduction is based on the *Last Appearance Record (LAR)* [15, 26], which allows us to identify states in V appearing infinitely often. Formally, V is assumed to be linearly ordered, i.e. $V = \{v_1, \ldots, v_n\}$. We let $P(V)$ the set of permutations of V, which we represent by words of length n over V with pairwise different letters. We define a deterministic finite automaton $LAR_V = (P(V) \times \{0, \ldots, |V|-1\}, (m_0, h_0), \to)$, $m_0 = v_1 \ldots v_n$ and $h_0 = 1$, and $(m, h) \xrightarrow{v} (x_1x_2v, |x_1|)$ where $m = x_1vx_2 \in P(V)$ for some $x_1, x_2 \in V^*$ and $v \in V$. Positions h are called *hits*. LAR_V has the following property: take a sequence $\pi \in V^\omega$ and the associated sequence of LAR_V's states $\ell = (m_0, h_0)(m_1, h_1)\ldots$, let h^π_{min} be the smallest hit appearing infinitely often in ℓ, then the sequence of subsets $(\{m_i[r] \mid r \geqslant h^\pi_{min}\})_{i\geqslant 0}$ eventually equals $\inf(\pi)$.

Remind that $Q = \{q_0, \top\} \cup (2^\Omega \times 2^\Omega \times V)$, therefore each sequence of states not in $Q^*\{\top\}^\omega$ also gives the sequence of visited vertices of $\mathcal{G}$. We take the product of $\mathcal{G}'$ with LAR_V to add LAR information to the game $\mathcal{G}'$, and transform Eve's winning condition in $\mathcal{G}'$ into a parity condition pr in this product as follows: one uses $2|V|+2$ priorities and, for product states with q_0 or $\top$ as first component, the priority is 0, otherwise for states of the form $((W, D, v), (m, h))$, we let:

$$pr((W,D,v),(m,h)) = \begin{cases} 2h & \text{if } \forall i \in W, \{m[r] \mid r \geqslant h\} \models \mu_i \text{ and} \\ & \quad (\{m[r] \mid r \geqslant h\} \models \mu_0 \text{ or } \exists i \in D \text{ s.t. } \{m[r] \mid r \geqslant h\} \models \neg\mu_i) \\ 2h+1 & \text{otherwise} \end{cases}$$

For states whose first component belongs to Adam, we just put priority $2|V|+2$, so that they have no influence. That way we obtain a parity game $\mathcal{G}_{par}$ with an exponential number of states (in $\mathcal{G}$) but a polynomial number of priorities, in which Eve has a winning strategy iff she has a winning strategy in $\mathcal{G}'$. Finally, parity games can be solved in PTime in the number of states and exponential in the number of priorities [18, 22]. ◀

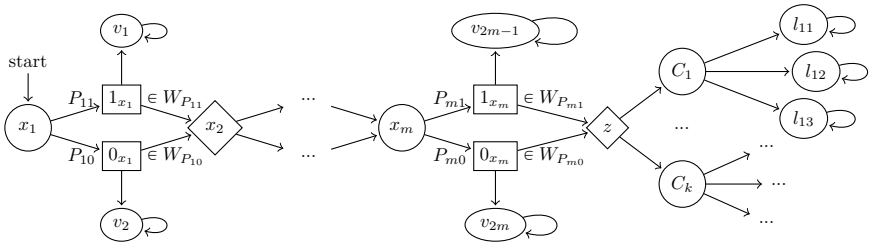


Figure 3 The arena $\mathcal{A}_\psi$ used to show that NCRSP is PSPACE-h.

▶ **Theorem 7.** *For multiplayer games, NCRSP is in* ExpTime *for objectives of type* $X \in \{\mathsf{Parity}, \mathsf{Streett}, \mathsf{Rabin}, \mathsf{Muller}\}$ *and in* PSpace *if* $X \in \{\mathsf{Safety}, \mathsf{Reachability}, \mathsf{Büchi}, \mathsf{coBüchi}\}$. *It is* PSpace-hard *for all objectives.*

Proof. The upper-bounds are consequence of Lemma 4, Lemma 5, and Proposition 6. Let us establish the lower bounds. Our proof is a reduction from QBF which uniformly works for all the types of objective. Let $\psi = \exists x_1 \forall x_2 ... \exists x_m \gamma(x_1, x_2, ..., x_m)$ be a QBF in 3CNF with k clauses $C_1, C_2, ..., C_k$. We build a $2m + 2$ players game $\mathcal{G}_\psi$, with players $\Omega = \{A, B, P_{10}, P_{11}, P_{20}, P_{21}, \ldots, P_{m0}, P_{m1}\}$, the system being played by player A, such that ψ is true if and only if $\mathcal{G}_\psi$ admits a solution to the NCRSP. The arena $\mathcal{A}_\psi$ of the game is depicted in Figure 3.

For each existential (resp. universal) variable x_i in ψ, the arena $\mathcal{A}_\psi$ contains a rounded (resp. diamond) state x_i controlled by Player A (resp. player B). For each node $x_i, 1 \leqslant i < m$, $\mathcal{A}_\psi$ contains the edges $(x_i, 0_{x_i}), (x_i, 1_{x_i})$, as well as the edges $(0_{x_i}, x_{i+1}), (1_{x_i}, x_{i+1})$, choices of edges in those states naturally encode valuations of the variables of the QBF formula.

For each $1 \leqslant i \leqslant m$, the rectangle state 1_{x_i} (resp. 0_{x_i}) is controlled by player P_{i1} (resp. P_{i0}) and has an additional edge leading to the self-loop over the state v_{2i-1} (resp. v_{2i}). The value nodes $1_{x_m}, 0_{x_m}$ (for the last variable x_m) are then connected to a vertex z controlled by Player B. From there Player B can choose a clause, i.e. an edge $(z, C_i), 1 \leqslant i \leqslant k$. Finally, each state associated to a clause C_i is controlled by Player A and has three outgoing edges toward the terminal nodes (with self-loops) l_{i1}, l_{i2}, l_{i3}, one for each literal in C_i.

Given the arena described above for the X-game $\mathcal{G}_\psi$, it is easy to phrase with the different types of objectives, the following winning conditions:

1. All paths that end in a state labeled with v_i $(1 \leqslant i \leqslant 2m)$ are winning for all the players.
2. All paths that end in a state labeled with l_{ij} $(1 \leqslant i \leqslant k, 1 \leqslant i \leqslant 3)$ are winning for all the players but Player A and Player P_{hb} such that $(l_{ij} = x_h \wedge b = 1) \vee (l_{ij} = \neg x_h \wedge b = 0)$.

Let us establish the correctness of our reduction. Assume that ψ is true. Then, the existential player has a winning strategy in the QBF game associated to ψ, i.e. he can choose the value of existentially quantified variables x_i as a function of the values given by the universal player to the universally quantified variables x_j, with $j \leqslant i$, so that ψ evaluates to true. We claim that, if Player A plays in the game arena according to such a winning strategy of the existential player, then Player A wins his objective whenever the other players play a NE. Indeed, there are two possibilities: either the game ends in a looping state labeled with v_i, and all players win (including Player A) or the game reaches z where Player B chooses a clause. As the strategy played by Player B encodes a winning strategy of the existential player in the QBF game, we know that each clause is satisfied, so Player A can choose a literal that evaluates to true in the clause. With such a choice, he makes sure that the player associated to this literal is losing while the play has visited a state in which this player has

decided to continue the game instead of going to a looping state labeled with v_i. This means that this player has a profitable deviation in the profile associated to the outcome of the game and so this outcome of the game is not a NE.

Otherwise, ψ is false. Then the universal player has a winning strategy in the QBF game associated to ψ. Consider a strategy profile where Player B plays according to this strategy and each Player P_{ib}, $1 \leqslant i \leqslant m, b \in \{0,1\}$, plays to continue the game and avoid looping states labelled by v_i. Then the game reaches state z, and Player B can choose a clause C_i that is false according to the instantiation of variables along the path followed so far. Therefore, for any choice of Player A from C_i, the play will be winning for all the players with the exception of Player A, and the player associated to the literal that has been chosen by Player A in the clause. This outcome is part of a NE as the later player cannot improve on his payoff by deviating as in the current profile, the play does not visit the rectangle state associated to the literal that evaluates to false, and so he has no available deviation at all. ◀

5 Fixed number of players

Several instances of the rational synthesis problems become tractable for fixed number of players. The number of players is a natural parameter to study: in practical applications, the number of components composing the environment may be limited to a few.

Cooperative Setting. We provide a generic reduction for the lower bounds of Table 1.

▶ **Lemma 8.** *Let $X \in \{\mathsf{Safety}, \mathsf{Reachability}, \mathsf{Büchi}, \mathsf{coBüchi}, \mathsf{Parity}, \mathsf{Streett}, \mathsf{Rabin}, \mathsf{Muller}\}$. Given a two-player zero-sum game between players A and B with an objective of type X for Player A, we can construct a multiplayer game with objective of type X with two players $\Omega = \{0,1\}$ such that Player A does not have a winning strategy in the zero-sum game if and only if the multiplayer game is a positive instance of the CRSP problem.*

Proof. Let $\mathcal{G}$ be a two-players zero-sum game where the protagonist (player A) has the objective ψ, and so Player B has objective $\neg\psi$. We construct the two-players CRSP $\mathcal{G}'$ by considering a copy of $\mathcal{G}$ and two fresh states v and w. The state v is the initial state of $\mathcal{G}'$ and has a transition to the initial state of $\mathcal{G}$ and a transition to w, which is equipped with a self-loop. The environment (Player 1) controls v, w and the states belonging to Player A in $\mathcal{G}$, while the system (Player 0) controls the states belonging to Player B in $\mathcal{G}$. For the winning conditions, Player 0 wins only if the play gets into w (and stays there forever), while the objective of the environment is ψ (i.e. the objective of Player A in $\mathcal{G}$).

$\mathcal{G}'$ is a positive instance of the CRSP problem iff Player 1 playing edge $v \to w$ is a NE. But clearly Player 1 does not have an incentive to deviate iff Player A does not have a winning strategy in $\mathcal{G}$ for forcing ψ. ◀

As an example, we get NP-hardness for the CRSP problem with Streett objectives because two-player zero-sum Streett games are coNP-hard.

The upper bounds on k-fixed CRSP for $X \in \{\mathsf{Reach}, \mathsf{Safe}, \mathsf{Büchi}, \mathsf{coBüchi}, \mathsf{Parity}, \mathsf{Rabin}\}$ listed in Table 1 can be obtained with the following procedure. First, compute the winning sets W_i for each of the $k+1$ players using the classical algorithms to solve two-player zero-sum games (w.r.t. the corresponding objective) and label the arena with the information $(W_i)_{i\in\Omega}$ (seen as atomic propositions). Then, check whether there is a path π such that $\pi \models \varphi_0 \wedge \phi^{\mathcal{G}}_{0\text{Nash}}$ (witnessing a solution to CRSP, by Proposition 2). The first step can be done in polynomial time for $X \in \{\mathsf{Reach}, \mathsf{Safe}, \mathsf{Büchi}, \mathsf{coBüchi}\}$, in UP$\cap$coUP for parity

conditions and in P^{NP} for Rabin conditions (as checking whether a state belongs to W_i is in NP). Due to the assumption that k is a fixed constant, the second step can be done in PTime for $X \in \{\mathsf{Reach}, \mathsf{Safe}, \mathsf{Büchi}, \mathsf{coBüchi}, \mathsf{Parity}\}$ and in NP for Rabin conditions. For Streett k-fixed CRSP, an NP upper bound was given in [25]. For Muller objectives, by Lemma 8, the problem is PSpace-hard and the PSpace upper bound follows from the unfixed case.

Non-Cooperative Setting. Results are summarized in the last column of Table 1. We start by the justification of the lower bounds. First, it should be clear that deciding the winner in a zero-sum two-player game with objective of type X is a special case of the NCRSP problem. Indeed, assume Player A has objective ψ in a game $\mathcal{G}$. Then, if we give objective ψ to Player 0 on the same game arena and declare all plays winning for Player 1, it is easy to see that this is a positive instance to the NCRSP problem iff Player A has a winning strategy in $\mathcal{G}$ for ψ. So this explains all the lower bounds but for $X \in \{\mathsf{Parity}, \mathsf{Streett}, \mathsf{Rabin}\}$. In the long version of this work [12], we provide a PSpace lower bound also to Streett and Rabin k-fixed NCRSP. This is done in two steps: first a reduction from QBF to zero-sum two-player Muller games is provided, similar to the one given in [16]. Then, the latter is reduced to a Street (resp. Rabin) NCRSP with two players. Finally, the lower bounds for parity k-fixed NCRSP reported in Table 1 have been obtained by reduction from the generalized parity games considered in [10], where the objective is a disjunction (dually, a conjunction) of parity conditions. In particular, we have proven that NCRSP on 3-players (resp. 4-players) parity game is NP (resp. coNP)-hard.

For $X \in \{\mathsf{Safety}, \mathsf{Reachability}, \mathsf{Büchi}, \mathsf{coBüchi}\}$, if the number of players in the given NCRSP is a fixed constant k, then the two-player game construction to test tree automata emptiness in Section 4 yields a polynomial size two-player zero-sum game $\mathcal{G}'$, where the formula characterizing the winning condition has constant size. The latter can be converted into an equivalent deterministic Büchi tree automaton of polynomial size, whose product with $\mathcal{G}'$ gives a Büchi game, solvable in PTime. Finally, if $X \in \{\mathsf{Parity}, \mathsf{Streett}, \mathsf{Rabin}, \mathsf{Muller}\}$, the corresponding k-fixed NCRSP can be reduced to a polynomial-size two-player game, with a 0-sum Muller acceptance condition defined by a boolean formula of polynomial size. Hence, the PSpace upper bound listed in Table 1 applies to k-fixed NCRSP with the above objectives.

References

1. Benjamin Aminof and Sasha Rubin. First cycle games. In *Proceedings 2nd International Workshop on Strategic Reasoning, SR 2014, Grenoble, France, April 5-6, 2014.*, pages 83–90, 2014. doi:10.4204/EPTCS.146.11.
2. Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.
3. Dietmar Berwanger. Admissibility in infinite games. In *STACS 2007, 24th Annual Symposium on Theoretical Aspects of Computer Science, Aachen, Germany, February 22-24, 2007, Proceedings*, volume 4393 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2007.
4. Romain Brenguier, Lorenzo Clemente, Paul Hunter, Guillermo A. Pérez, Mickael Randour, Jean-François Raskin, Ocan Sankur, and Mathieu Sassolas. Non-zero sum games for reactive synthesis. *CoRR*, abs/1512.05568, 2015. URL: http://arxiv.org/abs/1512.05568.
5. Romain Brenguier, Jean-François Raskin, and Ocan Sankur. Assume-admissible synthesis. In *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 100–113. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015. doi:10.4230/LIPIcs.CONCUR.2015.100.

6 Romain Brenguier, Jean-François Raskin, and Mathieu Sassolas. The complexity of admissibility in omega-regular games. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS'14, Vienna, Austria, July 14-18, 2014*, pages 23:1–23:10. ACM, 2014.

7 Krishnendu Chatterjee, Laurent Doyen, Emmanuel Filiot, and Jean-François Raskin. Doomsday equilibria for omega-regular games. In *Verification, Model Checking, and Abstract Interpretation – 15th International Conference, VMCAI 2014, San Diego, CA, USA, January 19-21, 2014, Proceedings*, volume 8318 of *Lecture Notes in Computer Science*, pages 78–97. Springer, 2014.

8 Krishnendu Chatterjee and Thomas A. Henzinger. Assume-guarantee synthesis. In *Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24 – April 1, 2007, Proceedings*, volume 4424 of *Lecture Notes in Computer Science*, pages 261–275. Springer, 2007.

9 Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdzinski. Games with secure equilibria. *Theor. Comput. Sci.*, 365(1-2):67–82, 2006. doi:10.1016/j.tcs.2006.07.032.

10 Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Generalized parity games. In *Proceedings of the 10th International Conference on Foundations of Software Science and Computational Structures*, FOSSACS'07, pages 153–167, Berlin, Heidelberg, 2007. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=1760037.1760051.

11 Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. Nusmv 2: An open-source tool for symbolic model checking. In *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364. Springer, 2002.

12 Rodica Condurache, Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. The complexity of rational synthesis – full version. Available at: http://lacl.fr/~rbozianu/papers/fullRationalSynthesis.pdf, 2016.

13 Marco Faella. Admissible strategies in infinite games over graphs. In *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings*, volume 5734 of *Lecture Notes in Computer Science*, pages 307–318. Springer, 2009.

14 Dana Fisman, Orna Kupferman, and Yoad Lustig. Rational synthesis. *CoRR*, abs/0907.3019, 2009. URL: http://arxiv.org/abs/0907.3019.

15 Yuri Gurevich and Leo Harrington. Trees, automata, and games. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC'82, pages 60–65, New York, NY, USA, 1982. ACM. doi:10.1145/800070.802177.

16 Paul Hunter and Anuj Dawar. Complexity bounds for regular games. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, Lecture Notes in Computer Science, pages 495–506, Berlin, Heidelberg, 2005. Springer-Verlag.

17 M. Jurdzinski. Algorithms for solving parity games. In *Lectures in Game Theory for Computer Scientists*, pages 74–98. Cambridge University Press, 2011.

18 Marcin Jurdzinski, Mike Paterson, and Uri Zwick. A deterministic subexponential algorithm for solving parity games. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 117–123, 2006. URL: http://dl.acm.org/citation.cfm?id=1109557.1109571.

19 Orna Kupferman, Giuseppe Perelli, and Moshe Y. Vardi. Synthesis with rational environments. In *Multi-Agent Systems – 12th European Conference, EUMAS 2014, Prague, Czech Republic, December 18-19, 2014, Revised Selected Papers*, pages 219–235, 2014. doi:10.1007/978-3-319-17130-2_15.

20 Fabio Mogavero, Aniello Murano, Giuseppe Perelli, and Moshe Y. Vardi. What makes atl* decidable? A decidable fragment of strategy logic. In *CONCUR 2012 – Concurrency Theory – 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 193–208. Springer, 2012.

21 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *POPL*, pages 179–190. ACM Press, 1989. doi:10.1145/75277.75293.

22 Sven Schewe. Solving parity games in big steps. In *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science, 27th International Conference, New Delhi, India, December 12-14, 2007, Proceedings*, volume 4855 of *Lecture Notes in Computer Science*, pages 449–460. Springer, 2007. doi:10.1007/978-3-540-77050-3_37.

23 Wolfgang Thomas. On the synthesis of strategies in infinite games. In *STACS*, pages 1–13, 1995. doi:10.1007/3-540-59042-0_57.

24 Michael Ummels. Rational behaviour and strategy construction in infinite multiplayer games. In *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science, 26th International Conference, Kolkata, India, December 13-15, 2006, Proceedings*, volume 4337 of *Lecture Notes in Computer Science*, pages 212–223. Springer, 2006.

25 Michael Ummels. The complexity of nash equilibria in infinite multiplayer games. In *Foundations of Software Science and Computational Structures, 11th International Conference, FOSSACS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29 – April 6, 2008. Proceedings*, volume 4962 of *Lecture Notes in Computer Science*, pages 20–34. Springer, 2008. doi:10.1007/978-3-540-78499-9_3.

26 Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theor. Comput. Sci.*, 200(1-2):135–183, 1998. doi:10.1016/S0304-3975(98)00009-7.

On the Complexity of Grammar-Based Compression over Fixed Alphabets

Katrin Casel*[1], Henning Fernau[2], Serge Gaspers†[3], Benjamin Gras[4], and Markus L. Schmid[5]

1 **Trier University, Fachbereich IV – Abteilung Informatikwissenschaften, Trier, Germany**
Casel@uni-trier.de
2 **Trier University, Fachbereich IV – Abteilung Informatikwissenschaften, Trier, Germany**
Fernau@uni-trier.de
3 **UNSW Australia, Sydney, Australia, and Data61 (formerly: NICTA), CSIRO, Sydney, Australia**
sergeg@cse.unsw.edu.au
4 **École Normale Superieure de Lyon, Département Informatique, Lyon, France**
benjamin.gras@ens-lyon.fr
5 **Trier University, Fachbereich IV – Abteilung Informatikwissenschaften, Trier, Germany**
MSchmid@uni-trier.de

Abstract

It is shown that the shortest-grammar problem remains NP-complete if the alphabet is fixed and has a size of at least 24 (which settles an open question). On the other hand, this problem can be solved in polynomial-time, if the number of nonterminals is bounded, which is shown by encoding the problem as a problem on graphs with interval structure. Furthermore, we present an $\mathcal{O}(3^n)$ exact exponential-time algorithm, based on dynamic programming. Similar results are also given for 1-level grammars, i. e., grammars for which only the start rule contains nonterminals on the right side (thus, investigating the impact of the "hierarchical depth" on the complexity of the shortest-grammar problem).

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, E.4 Coding and Information Theory

Keywords and phrases Grammar-Based Compression, Straight-Line Programs, NP-Completeness, Exact Exponential Time Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.122

1 Introduction

While in the early days of computer science, the most important requirements for compression schemes were fast (i. e., linear or near linear time) compression and decompression, nowadays their investigation regarding whether they are suitable for solving problems directly on

* Katrin Casel is supported by the Deutsche Forschungsgemeinschaft (FE 560/6-1).
† Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (DP150101134). NICTA is funded by the Australian Government through the Department of Communications and the ARC through the ICT Centre of Excellence Program.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 122; pp. 122:1–122:14

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

the compressed data without prior decompression forms a vibrant research area, usually subsumed under the term *algorithmics on compressed strings*.

Compressing a word by a context-free grammar, so-called *grammar-based compression*, is particularly well suited for this purpose.[1] Nevill-Manning and Witten [17, 16], and Kieffer et al. [10, 9, 23] are stated as the origins of this concept, but a closer look into the older literature reveals that the *external pointer macro* (*without overlapping and with pointer size* 1) defined by Storer and Szymanski [22, 21] is also equivalent to grammar-based compression.

The success of grammars with respect to algorithmics on compressed strings is due to the fact that they cover many compression schemes from practice (most notably, the family of Lempel-Ziv encodings) and that they are mathematically easy to handle (see Lohrey [11] for a survey on the role of grammar-based compression for algorithmics on compressed strings). Many basic problems on strings, e. g., comparison, pattern matching, membership in a regular language, retrieving subwords, etc. can all be solved in polynomial-time directly on the grammars [11]. In addition, grammar-based compression has been successfully applied in combinatorial group theory and to prove problems in computational topology to be polynomial-time solvable [11]. Grammars as compressions have also been extended to more complicated objects, e. g., trees (see [1, 12, 13, 14]) and two-dimensional words (see [3]).

On the other hand, work on the *shortest-grammar problem*, i. e., computing a minimal grammar for a given word w, is somewhat scarce; its NP-completeness, the major downside of grammar-based compression, has been used as a justification to focus on approximation algorithms. In this regard, the best achieved approximation ratio is $\mathcal{O}(\log(\frac{|w|}{m^*}))$ (see [18, 4]), where m^* is the size of a smallest grammar, and an approximation ratio better than $\frac{8569}{8568} \approx 1.0001$ is not possible (see [4]), assuming $\mathsf{P} \neq \mathsf{NP}$ (the research seems to have stagnated at this gap between lower and upper bound). However, the existing hardness reductions (also for the inapproximability result) have a serious deficiency: they assume the terminal alphabet to be unbounded. In [2], it is claimed that the hardness for alphabets of size 3 follows from [21], but a closer look into [21] does not confirm this. For string problems, where we typically deal with not only constant, but also very small alphabets, e. g., of size 2 or 4, this is rather unsatisfying. Another neglected aspect is the parameterised point of view, i. e., can minimal grammars be efficiently computed, if certain parameters (e. g., alphabet size, levels of the derivation tree, number of rules) are bounded? Furthermore, the fact that for grammar compressions basic problems can be solved without decompression motivates scenarios, where an extensive running time is invested only once, in order to obtain an optimal compression, which is then stored and worked with. This assumption naturally leads to exact exponential-time algorithms, which are not yet considered in the literature.

We investigate these aspects of the shortest-grammar problem. First, we close the gap with respect to fixed alphabets left open in the literature, by showing the NP-completeness for alphabets of size 24 in Section 3, which provides a more solid foundation for approximations (or heuristics), but leaves the cases of small alphabets open.[2] After this negative result, in Section 4, we show that minimal grammars can be computed in polynomial-time, provided that the size of the nonterminal alphabet is bounded. This is achieved by a reduction to a graph problem, which, since the graphs are structurally simple, can be efficiently solved (note that especially for string problems successful applications of this common algorithmic technique are rare). Additionally, we show that an FPT-algorithm with respect to this

[1] Such context-free grammars are also called *straight-line programs* in the literature.
[2] Note that this negative result also transfers to the shortest-grammar problem for trees.

parameter is unlikely (under complexity theoretical assumptions). Finally, we turn our attention to exact exponential-time algorithms in Section 5 and first observe that brute-force algorithms with running time $\mathcal{O}(c^{|w|})$, for a constant c, can be easily found, but we also present an $\mathcal{O}(3^{|w|})$ dynamic programming algorithm.

Moreover, these questions are also investigated for 1-*level grammars*, i. e., only the start rule contains nonterminals; thus, measuring the impact of the "hierarchical depth" of the grammars. Considering that the exploitation of hierarchical structure is one of the main features of grammars (allowing exponential compression rates, in contrast to 1-level grammars, where quadratic is the best), it is surprising that our results suggest that computing general grammars is, if at all, only insignificantly more difficult than computing 1-level grammars.

Due to space restriction, we only give proof sketches for the main results.

2 Preliminaries

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$. By $|A|$, we denote the cardinality of a set A. Let Σ be a finite alphabet of *symbols*. A *word* or *string* (over Σ) is a sequence of symbols from Σ. For any word w over Σ, $|w|$ denotes the length of w and ε denotes the *empty word*, i. e., $|\varepsilon| = 0$. The symbol Σ^+ denotes the set of all non-empty words over Σ and $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$. For the *concatenation* of two words w_1, w_2 we write $w_1 \cdot w_2$ or simply $w_1 w_2$. For every symbol $a \in \Sigma$, by $|w|_a$ we denote the number of occurrences of symbol a in w. We say that a word $v \in \Sigma^*$ is a *factor* of a word $w \in \Sigma^*$ if there are $u_1, u_2 \in \Sigma^*$ such that $w = u_1 v u_2$. If $u_1 = \varepsilon$ or $u_2 = \varepsilon$, then v is a *prefix* (or a *suffix*, respectively) of w. Furthermore, $\mathsf{F}(w) = \{u \colon u \text{ is a factor of } w\}$ and $\mathsf{F}_{\geq 2}(w) = \{u \colon u \in \mathsf{F}(w), |u| \geq 2\}$. For a position j, $1 \leq j \leq |w|$, we refer to the symbol at position j of w by the expression $w[j]$ and $w[j..j'] = w[j]w[j+1] \ldots w[j']$, $j \leq j' \leq |w|$. By w^R, we denote the *reversal* of w, i. e., $w^R = w[n]w[n-1] \ldots w[1]$, where $|w| = n$.

A *factorisation* of a word w is a tuple $(u_1, u_2, \ldots, u_k)$ with $u_i \neq \varepsilon$, $1 \leq i \leq k$ such that $w = u_1 u_2 \ldots u_k$. A factorisation $p = (u_1, u_2, \ldots, u_k)$ is a *refinement* of a factorisation $q = (v_1, v_2, \ldots, v_m)$, denoted by $p \preceq q$, if $(u_{j_{i-1}+1}, u_{j_{i-1}+2}, \ldots, u_{j_i})$ is a factorisation of v_i, $1 \leq i \leq m$ for some $\{j_i\}_{0 \leq i \leq m}$, with $0 = j_0 < j_1 < \ldots < j_m = k$.

Grammars: A *context-free grammar* is a tuple $G = (N, \Sigma, R, S)$, where N is the set of *nonterminals*, Σ is the *terminal alphabet*, $S \in N$ is the *start symbol* and $R \subseteq N \times (N \cup \Sigma)^+$ is the set of *rules* (as a convention, we write rules $(A, w) \in R$ also in the form $A \to w$). A context-free grammar $G = (N, \Sigma, R, S)$ is a *singleton grammar* if R is a total function $N \to (N \cup \Sigma)^+$ and the relation $\{(A, B) \colon (A, w) \in R, |\alpha|_B \geq 1\}$ is acyclic.

For a singleton grammar $G = (N, \Sigma, R, S)$, let $\mathsf{D}_G : (N \cup \Sigma) \to (N \cup \Sigma)^+$ be defined by $\mathsf{D}_G(A) = R(A)$, $A \in N$, and $\mathsf{D}_G(a) = a$, $a \in \Sigma$. We extend D_G to a morphism $(N \cup \Sigma)^+ \to (N \cup \Sigma)^+$ by setting $\mathsf{D}_G(\alpha_1 \alpha_2 \ldots \alpha_n) = \mathsf{D}_G(\alpha_1)\,\mathsf{D}_G(\alpha_2) \ldots \mathsf{D}_G(\alpha_n)$, for $\alpha_i \in (N \cup \Sigma)$, $1 \leq i \leq n$. Furthermore, for every $\alpha \in (N \cup \Sigma)^+$, we set $\mathsf{D}^1_G(\alpha) = \mathsf{D}_G(\alpha)$, $\mathsf{D}^k_G(\alpha) = \mathsf{D}(\mathsf{D}^{k-1}_G(\alpha))$, for every $k \geq 2$, and $\mathfrak{D}_G(\alpha) = \lim_{k \to \infty} \mathsf{D}^k_G(\alpha)$ is the *derivative* of α. By definition, $\mathfrak{D}_G(\alpha)$ exists for every $\alpha \in (N \cup \Sigma)^+$ and is an element from Σ^+. The size of the singleton grammar G is defined by $|G| = \sum_{A \in N} |\,\mathsf{D}_G(A)|$ and its number of *levels* is $\min\{k \colon \mathsf{D}^k_G(S) = \mathfrak{D}_G(S)\}$. In particular, a grammar with d levels is a *d-level grammar*.

From now on, we simply use the term *grammar* instead of singleton grammar and if the grammar under consideration is clear from the context, we also drop the subscript G. We set $\mathfrak{D}(G) = \mathfrak{D}(S)$ and say that G *is a grammar for* $\mathfrak{D}(G)$. In the tuple (N, Σ, R, S), we sometimes replace S directly by $\mathsf{D}(S)$, which we then call the *compressed string* (*of* G) and which we denote by cs.

Let $G = (N, \Sigma, R, cs)$ be a 1-level grammar. The *profit* of a rule $(A, \alpha) \in R$ is defined by $\mathsf{p}(A) = |cs|_A(|\alpha| - 1) - |\alpha|$. Intuitively speaking, if all occurrences of A in cs are replaced by α and the rule $A \to \alpha$ is deleted, then the size of the grammar increases by exactly $\mathsf{p}(A)$. Consequently, $|G| = |\mathfrak{D}(G)| - \sum_{A \in N} \mathsf{p}(A)$.

A grammar G is *minimal* if $|G| = \min\{|G'| \colon G' \text{ is a grammar for } \mathfrak{D}(G)\}$ and the problem of computing small grammars is defined as follows:

Shortest Grammar Problem (SGP)
Instance: A word w and a $k \in \mathbb{N}$.
Question: Does there exist a grammar G with $\mathfrak{D}(G) = w$ and $|G| \leq k$?

The Shortest 1-Level Grammar Problem (1-SGP) is defined analogously, with the only difference that we ask for a 1-level grammar of size at most k.

Examples: Even for small – say binary – alphabets and a fixed word with a simple structure, finding minimal grammars can be surprisingly difficult. In order to substantiate this claim, let $w = \prod_{i=1}^{n} 10^i$ be a word over $\Sigma = \{0, 1\}$, where $n = 2^k$, $k \in \mathbb{N}$. One way of compressing w that comes to mind is by the use of rules $A_1 \to 10$, $A_i \to A_{i-1}0$, $2 \leq i \leq n-1$, and a compressed string $A_1 A_2 \dots A_{n-1} A_{n-1} 0$, which leads to a grammar G_1 of size $3n-1$. However, it is also possible to construct the factors 0^i, $1 \leq i \leq n$, "from the middle" by rules $A_1 \to 010$, $A_i \to 0A_{i-1}0$, $2 \leq i \leq \frac{n}{2} - 1$, and a compressed string $1(A_1)^2(A_2)^2 \dots$. By using these ideas, we can construct a smaller grammar G_2 of size $\frac{5n}{2} + 2k - 3$. Both of these grammars achieve a compression rate of order $\mathcal{O}(\sqrt{|w|})$, but, generally, grammars are capable of exponential compression rates (see [4]). Aiming for such exponential compression, it seems worthwhile to represent every unary factor 0^{2^ℓ}, $1 \leq \ell \leq k$, by a nonterminal B_ℓ (obviously, this requires only k rules of size 2) and then represent all unary factors by sums of these powers (e. g., 0^{74} is compressed by $B_1 B_3 B_6$). However, this yields a grammar G_3 of size $\frac{k(n+3)}{2} - 2$, which, if k is sufficiently large, is worse than the previous grammars.

A smaller grammar can be obtained by combining the idea of G_2 with that of representing factors 0^{2^ℓ} by nonterminals B_ℓ. More precisely, for every ℓ, $1 \leq i \leq k-2$, we represent 0^{2^ℓ} by an individual nonterminal B_ℓ and, in addition, we use rules $A_1 \to 010$, $A_i \to 0A_{i-1}0$, $2 \leq i \leq \frac{n}{4}$. Then the left and right half of w can be compressed in the way of G_2, with the only difference that in the right part, for every unary factor, we also need an occurrence of B_{k-1}, i. e., the compressed string is $1(A_1)^2 \dots (A_{\frac{n}{4}})^2 B_{k-2} (A_1 B_{k-1})^2 \dots (A_{\frac{n}{4}-1} B_{k-1})^2 A_{\frac{n}{4}} (B_{k-2})^3$. The size of this grammar G_4 is only $\frac{9n}{4} + 2k - 2$.

3 NP-Hardness of Computing Minimal Grammars for Fixed Alphabets

In [4], Charikar et al. prove the shortest-grammar problem to be NP-complete by a reduction from the vertex cover problem (which is based on ideas used by Storer and Szymanski in [22]). A simple modification of this reduction yields the following.

▶ **Theorem 1.** 1-SGP *is* NP*-complete.*

In these reductions, we encode the different vertices of a graph by single symbols and also use individual separator symbols (i. e., symbols with only one occurrence in the word to be compressed). This makes it particularly easy to devise suitable gadgets, but, on the other hand, it assumes that we have an arbitrarily large alphabet at our disposal, which, for practical situations, is not justified. In the remainder of this section, we shall extend these hardness results to the more realistic case of fixed alphabets. The general structure of

our reductions is similar to the ones of [4, 21], but, due to the constraint of having a fixed alphabet, they substantially differ on a more detailed level.

Since fixed alphabets make it impossible to use single symbols (or even words of constant size) as separators or as representing vertices, we need to use special encodings for which we are able to determine how a smallest grammar will compress them (in this regard, recall our examples from page 4 demonstrating how difficult it can be to determine a smallest grammar even for a single simply structured word). This constitutes a substantial technical challenge, which complicates our reductions considerably.

3.1 The 1-Level Case

As a tool for proving the hardness of 1-SGP, but also as a result in its own right, we first show that the compression of any 1-level grammar is at best quadratic (in contrast to general grammars, which can achieve exponential compression (see [4])).[3]

▶ **Lemma 2.** *Let G be a 1-level grammar. Then $|G| \geq 2\left\lfloor\sqrt{|\mathfrak{D}(G)|}\right\rfloor$.*

In order to prove the NP-hardness of 1-SGP for constant alphabets, we devise a reduction from the vertex cover problem. To this end, let $\mathcal{G} = (V, E)$ be a graph with $V = \{v_1, \ldots, v_n\}$ and $E = \{(v_{j_{2i-1}}, v_{j_{2i}}) \colon 1 \leq i \leq m\}$. Without loss of generality, we assume $n \geq 40$. We define $\Sigma = \{\mathtt{a}, \mathtt{b}, \diamond, \star, \#\}$ and $[\diamond] = \diamond^{n^3}$. For each i, $1 \leq i \leq n$, we encode v_i by a word $\overline{v_i} \in \{\mathtt{a}, \mathtt{b}\}^{\lceil\log(n)\rceil}$ such that $\overline{v_i} \neq \overline{v_j}$ if and only if $i \neq j$ (e. g., by taking $\overline{v_i}$ to be the binary representation of i over symbols $\mathtt{a}$ and $\mathtt{b}$ with $\lceil\log(n)\rceil$ many digits). We now define the following word over Σ:

$$w = \prod_{i=1}^{n} (\#\overline{v_i}[\diamond]\overline{v_i}\#[\diamond])^{2\lceil\log(n)\rceil+3} \prod_{i=1}^{n} (\#\overline{v_i}\#[\diamond])^{\lceil\log(n)\rceil+1} \prod_{i=1}^{m} (\#\overline{v_{j_{2i-1}}}\#\overline{v_{j_{2i}}}\#[\diamond])^2 \star [\diamond]^{n^3}.$$

▶ **Theorem 3.** 1-SGP *is* NP*-hard, even for* $|\Sigma| = 5$.

Proof Sketch. A smallest grammar for w produces the two parts to the left and right of $\star$ independently, since $|w|_\star = 1$. According to Lemma 2, the right side $[\diamond]^{n^3}$ is best compressed by n^3 occurrences of a nonterminal D with derivative $[\diamond]$ and, by a slightly more involved argument, it can be shown that also for the whole word w, it is still best to compress all occurrences of $[\diamond]$ by D. Having established this basic property, it is then possible to show that the remaining rules have derivative $\#\overline{v_i}$, $\overline{v_i}\#$ or $\#\overline{v_j}\#$. The grammar is smallest, if every edge $\#\overline{v_{j_{2i-1}}}\#\overline{v_{j_{2i}}}\#$ is compressed by using a rule of the last type; thus, those rules translate into a vertex cover. Analogously, a vertex cover translates into a grammar. ◀

3.2 The Multi-Level Case

In the above reduction, the main difficulty is the use of unary factors as separators. However, once those separators are in place, we know the factors of w that are produced by nonterminals and, for a minimal 1-level grammar, this already fully determines the compressed string and, thus, the grammar itself. For the multi-level case, the situation is much more complicated. Even if we manage to force the compressed string to factorise w into parts that are either separators or codewords of vertices, this only determines the top-most level of the grammar and we do not necessarily know how these single factors are further hierarchically compressed

[3] The bound of Lemma 2 is tight, e. g., consider $\mathtt{a}^{n^2}$ and a grammar with rules $S \to A^n$ and $A \to \mathtt{a}^n$.

and, more importantly, the dependencies between these compressions (i. e., how they share the same rules).

To deal with these issues, we rely on a larger alphabet Σ and we use palindromic codewords $u \star u^R$, where $\star \in \Sigma$ and u is a word over an alphabet of size 7 representing a 7-ary number. The purpose of the palindromic structure is twofold. Firstly, it implies that codewords always start and end with the same symbol, which, in the construction of w, makes it easier to avoid the situation that an overlapping between neighbouring codewords is repeated elsewhere in w (see Lemma 4). Secondly, if all codewords are produced by individual nonterminals, then we can show that they are produced best "from the middle", similar as the rules of the example grammar G_2 from page 4. In addition to this, we also need a vertex colouring and an edge colouring of certain variants of the graph to be encoded.

In order to formally define the reduction, we first give some preparatory definitions. Let $\Sigma = \{x_1, \ldots, x_7, d_1, \ldots, d_7, \star, \#, \mathfrak{c}_1, \mathfrak{c}_2, \$_1, \ldots, \$_6\}$ be an alphabet of size 24. The function $M\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is defined by $M(q,k) := \min\{r > 0\colon \exists t \in \mathbb{N}\colon q = tk + r\}$.[4] Let the functions $f\colon \mathbb{N} \to \{x_1, \ldots, x_7\}^+$ and $g\colon \mathbb{N} \to \{d_1, \ldots, d_7\}^+$ be defined by $f(q) := x_{a_0} x_{a_1} \ldots x_{a_k}$ and $g(q) := d_{a_0} d_{a_1} \ldots d_{a_k}$, for every $q \in \mathbb{N}$, where $k \in \mathbb{N} \cup \{0\}$ and $a_i \in \{1, 2, \ldots, 7\}$, $0 \leq i \leq k$, are such that $q = \sum_{i=0}^{k} a_i 7^i$ is satisfied.[5] For every $i \in \mathbb{N}$, let $\langle i \rangle_v := f(i) \star f(i)^R$ and $\langle i \rangle_\diamond := g(i) \star g(i)^R$. The factors $\langle i \rangle_v$ and $\langle i \rangle_\diamond$ are called *codewords*; $\langle i \rangle_v$ represents a vertex v_i, while the $\langle i \rangle_\diamond$ are used as separators. The functions f and g are bijections and they are 7-ary representations of the integers $n > 0$ (least significant digit first). Thus, for every $n, n' \in \mathbb{N}$ with $M(n,7) \neq M(n',7)$, the words $\langle n \rangle_v$ and $\langle n' \rangle_v$ do not share any prefixes or suffixes (and the same holds for the words $\langle n \rangle_\diamond$).

Let $\mathcal{G} = (V, E)$ be a subcubic graph (i. e., a graph with maximum degree 3) with $V = \{v_1, \ldots, v_n\}$ and $E = \{\{v_{j_{2i-1}}, v_{j_{2i}}\}\colon 1 \leq i \leq m\}$ (note that the vertex cover problem remains NP-hard if restricted to subcubic graphs (see [7])). Let $\mathcal{G}' = (V, E')$ be the multi-graph defined by $E' := \big\{\{v_{j_{2i}}, v_{j_{2i+1}}\}\colon 1 \leq i \leq m-1\big\}$. By [19], it is possible to compute in polynomial-time a proper edge-colouring (meaning a colouring such that no two edges which share one or two vertices have the same colour) for a multi-graph with at most $\lfloor \frac{3}{2} m \rfloor$ colours, where m is the maximum degree of the multi-graph. Since $\mathcal{G}$ is subcubic, the maximum degree of $\mathcal{G}'$ is three and we can compute a proper edge-colouring $C_e\colon E' \to \{1,2,3,4\}$ for $\mathcal{G}'$ with colours {1,2,3,4}. Let $\mathcal{G}^2 = (V, E'')$ be the graph defined by $E'' = \{\{u,v\}\colon \{u,w\}, \{w,v\} \in E \text{ for some } w \in V \setminus \{u,v\}, u \neq v\}$. Since $\mathcal{G}$ is subcubic, $\mathcal{G}^2$ has maximum degree at most six. Let $C_v\colon \{1, \ldots, n\} \to \{1,2,3,4,5,6,7\}$ be a proper vertex-colouring (defined over the vertex-indices of $V = \{v_1, \ldots, v_n\}$) for $\mathcal{G}^2$ with colours $\{1,2,3,4,5,6,7\}$. Such a colouring can be computed by an algorithmic version of Brook's theorem [20].

Let $w_\mathcal{G} = uvw$ be the word representing $\mathcal{G}$, where $u, v, w \in \Sigma^+$ are defined as follows.[6]

$$u = \prod_{j=0}^{6} \left(\prod_{i=1}^{14n} (\langle i \rangle_\diamond \ \langle M(i+j, 14n) \rangle_v) \right) \ \$_1$$

$$v = \prod_{i=1}^{n} (\# \ \langle 7i + C_v(i) \rangle_v \ \mathfrak{c}_1 \ \langle 7i - 1 \rangle_\diamond) \ \$_2 \prod_{i=1}^{n} (\# \ \langle 7i + C_v(i) \rangle_v \ \mathfrak{c}_2 \ \langle 7i - 2 \rangle_\diamond) \ \$_3$$

[4] M is the positive modulo-function, i. e., $M(q,k) = q \% k$, if $q \% k \neq 0$ and $M(q,k) = k$, otherwise.

[5] Since, for every $q \in \mathbb{N}$, there are unique $k \in \mathbb{N}$ and $a_i \in \{1, 2, \ldots, 7\}$, $1 \leq i \leq k$, such that $q = \sum_{i \geq 0}^{k} a_i 7^i$, the functions f and g are well-defined.

[6] Note that $m \leq \frac{3n}{2}$, so $7m < 14n$ in the word w.

$$\prod_{i=1}^{n}(\langle 7i+C_v(i)\rangle_v \ \#\ \langle 7i-2\rangle_\diamond\ \mathfrak{c}_1)\ \ \$_4 \prod_{i=1}^{n}(\langle 7i+C_v(i)\rangle_v \ \#\ \langle 7i-1\rangle_\diamond\ \mathfrak{c}_2)\ \ \$_5$$
$$\prod_{i=1}^{n}(\#\ \langle 7i+C_v(i)\rangle_v \ \#\ \langle 7i\rangle_\diamond)\ \ \$_6$$
$$w = \prod_{i=1}^{m-1}(\#\ \langle 7j_{2i-1}+C_v(j_{2i-1})\rangle_v \ \#\ \langle 7j_{2i}+C_v(j_{2i})\rangle_v \ \#\ \langle 7i+C_e(v_{j_{2i}},v_{j_{2i+1}})\rangle_\diamond)$$
$$\#\ \langle 7j_{2m-1}+C_v(j_{2m-1})\rangle_v \ \#\ \langle 7j_{2m}+C_v(j_{2m})\rangle_v \#$$

The next lemma states that any factor of $w_\mathcal{G}$ is not repeated, if it spans over the $\star$ of some codeword $\langle i\rangle_v$ or $\langle i\rangle_\diamond$ and also reaches over the boundaries of this codeword into some other factor. This property, which can be proven by a straightforward, but rather cumbersome analysis, is crucial for the correctness of the reduction and also responsible for the complicated structure of $w_\mathcal{G}$. Here, we only wish to point out that it follows from the fact that all occurrences of the same codeword are delimited by distinct symbols. This is ensured by the symbols $\#, \mathfrak{c}_1, \mathfrak{c}_2, \$_1, \ldots, \$_6$, by the fact that codewords $\langle i\rangle_v$ and $\langle i\rangle_\diamond$ start and end with $x_{M(i,7)}$ and $d_{M(i,7)}$, respectively, and, for the part w, by the colourings C_e and C_v.

▶ **Lemma 4.** *There is a minimal grammar $G = (N, \Sigma, R, S)$ for $w_\mathcal{G}$ such that, for every $A \in N$, $|\mathfrak{D}(A)|_\star \geq 1$ implies that $\mathfrak{D}(A)$ is a factor of some $\#\,\langle 7i+C_v(i)\rangle_v\,\#$, $1 \leq i \leq n$, or a factor of some $\langle j\rangle_\diamond$, $1 \leq j \leq 14n$.*

▶ **Lemma 5.** *There is a minimal grammar G for $w_\mathcal{G}$ such that, for every i, $1 \leq i \leq 14n$, there is a nonterminal with derivative $\langle i\rangle_\diamond$ and a nonterminal with derivative $\langle i\rangle_v$, and, for every i, $1 \leq i \leq n$, there is a nonterminal with derivative $\#\,\langle 7i+C_v(i)\rangle_v$ and a nonterminal with derivative $\langle 7i+C_v(i)\rangle_v\#$.*

Proof Sketch. Let G be a minimal grammar. Since $|u|_\star = 196n$, Lemma 4 implies that $|\beta| \geq 196n$, where β is the prefix of the compressed string producing u. Also by Lemma 4, every $\langle i\rangle_\diamond$ or $\langle i\rangle_v$ that is not a derivative of some rule, is produced by at least two nonterminals (we assume that there are k many such *bad* codewords). This implies that β contains at least $7\lceil\frac{k}{2}\rceil$ additional nonterminals (each codeword has 7 occurrences in u and the nonterminal not producing $\star$ can be used in the production of at most 2 bad codewords). Hence, $|\beta| \geq 196n + 7\lceil\frac{k}{2}\rceil$. For every bad codeword $x \in \{\langle i\rangle_\diamond, \langle i\rangle_v\}$, we can add a new rule $A_x \to \alpha_x$ with $|\alpha_x| = 3$ and $\mathfrak{D}(A_x) = x$ (this can be done by right sides of the form $d_j A d_j$, where A derives another codeword). This increases the size of the grammar by $3k$, but we can now produce every codeword of u by one nonterminal, which shortens β by $7\lceil\frac{k}{2}\rceil > 3k$.

We now add rules $\overleftarrow{V_i} \to \#V_{7i+C_v(i)}$, $\overrightarrow{V_i} \to V_{7i+C_v(i)}\#$, where $\mathfrak{D}(V_{7i+C_v(i)}) = \langle 7i+C_v(i)\rangle_v$, and use them, in addition to the rules for the codewords $\langle i\rangle_\diamond$, to obtain a new compressed string from $w_\mathcal{G}$ (where also every factor $\#\langle 7i+C_v(i)\rangle_v\#$ that has been produced before by a single nonterminal is compressed by a rule $\overleftrightarrow{V_i} \to \overleftarrow{V_i}\#$). Then, we erase all old rules with derivatives $\#f(7i+C_v(i))\star r_i$, $r_i \star f(7i+C_v(i))^R\#$. The deletion of these rules and the size of the new compressed string compensates the size increase of adding the new rules. ◀

▶ **Lemma 6.** *There is a minimal grammar G for $w_\mathcal{G}$ with the rules $\{r_{\diamond,i}, r_{v,i} \colon 1 \leq i \leq 14n\}$, where $r_{\diamond,i} = D_i \to d_i \star d_i$ and $r_{v,i} = V_i \to x_i \star x_i$, $1 \leq i \leq 7$, $r_{\diamond,i} = D_i \to g(i)[1]D_{h(i)}g(i)[1]$ and $r_{v,i} = V_i \to f(i)[1]V_{h(i)}f(i)[1]$, $8 \leq i \leq 14n$ with $h(i) = \frac{1}{7}(i - M(i,7))$, $\{\overleftarrow{V_i} \to \#V_{7i+C_v(i)}, \overrightarrow{V_i} \to V_{7i+C_v(i)}\# \colon 1 \leq i \leq n\}$, $\{\overleftrightarrow{V_i} \to \#\overrightarrow{V_i} \colon i \in \mathfrak{I}\}$, for an $\mathfrak{I} \subseteq \{1, \ldots, n\}$, and with the compressed string*

$\prod_{j=0}^{6}\left(\prod_{i=1}^{14n}(D_i\, V_{M(i+j,14n)})\right)\prod_{i=1}^{n}\left(\overleftarrow{V_i}\ \mathfrak{c}_1\ D_{7i-1}\right)\prod_{i=1}^{n}\left(\overleftarrow{V_i}\ \mathfrak{c}_2\ D_{7i-2}\right)\prod_{i=1}^{n}\left(\vec{V_i}\ D_{7i-2}\ \mathfrak{c}_1\right)$
$\prod_{i=1}^{n}\left(\vec{V_i}\ D_{7i-1}\ \mathfrak{c}_2\right)\prod_{i=1}^{n}(y_i\ D_{7i})\prod_{i=1}^{m-1}(z_i D_{7i+C_e(v_{j_{2i}},v_{j_{2i+1}})})z_m$,
where for every i, $1\le i\le n$, $y_i=\overleftrightarrow{V_i}$, *if* $i\in\mathfrak{I}$ *and* $y_i=\overleftarrow{V_i}\#$, *otherwise, and, for every* k, $1\le k\le m$, $z_k\in\{\overleftrightarrow{V}_{j_{2k-1}}\vec{V}_{j_{2k}},\overleftarrow{V}_{j_{2k-1}}\overleftrightarrow{V}_{j_{2k}}\}$ *if* $\{j_{2k-1},j_{2k}\}\cap\mathfrak{I}\neq\emptyset$, $z_k=\overleftarrow{V}_{j_{2k-1}}\overleftarrow{V}_{j_{2k}}\#$, *otherwise.*

Proof Sketch. Lemma 5 ensures nonterminals $V_i\to\alpha_i$ with $\mathfrak{D}(\alpha_i)=\langle i\rangle_v$. We now replace it by a rule $V_i\to x_i\star x_i$ or $V_i\to f(i)[1]V_{h(i)}f(i)[1]$, as described in the statement of the lemma. If $|\alpha_i|\ge 3$, this is fine, but if $|\alpha_i|=2$, we have to argue more carefully: we remove $V_i\to AB$ for which i is maximal and observe that this implies that either A or B cannot occur on any right side of a rule; thus, can be removed. Repeating this argument turns all rules with derivative $\langle i\rangle_v$ in the right form and a similar argument applies to the rules with derivatives $\langle i\rangle_\diamond$. Now it only remains to prove that the compressed string can be assumed to have the desired form. If we replace all $\langle i\rangle_v$, $\langle i\rangle_\diamond$, $\#\langle i\rangle_v$ and $\langle i\rangle_v\#$ in $w_{\mathcal{G}}$ by the respective nonterminals, then this produces a compressed string whose size may increase compared to the original one, but only by the number of factors $\#\langle i\rangle_v\#$ that have been compressed by a single nonterminal and are now compressed by $\overleftarrow{V_i}\#$. This can be repaired by simply adding a rule $\overleftrightarrow{V_i}\to\#\vec{V_i}$, resulting in the set $\mathfrak{I}$ mentioned in the statement of the lemma. ◀

Lemma 6 allows us to argue similarly as for the reduction from [4]: $\Gamma=\{v_i\colon i\in\mathfrak{I}\}$ is a vertex cover (if $\{v_i,v_j\}\in E$ is not covered, then adding $\overleftrightarrow{V_i}\to\#\vec{V_i}$ does not increase the size of the grammar) and $|G|=f(m,n)+|\Gamma|$, where f is a polynomial. Furthermore, a vertex cover Γ translates into a grammar of size $f(m,n)+|\Gamma|$, by setting $\mathfrak{I}=\{i\colon v_i\in\Gamma\}$. Thus, $\mathcal{G}$ has a vertex cover Γ iff there is a grammar G for $w_{\mathcal{G}}$ with $|G|\le f(m,n)+|\Gamma|$.

▶ **Theorem 7.** SGP *is* NP*-complete, even for alphabets of size* 24.

4 Minimal Grammars with a Bounded Number of Nonterminals

A natural follow-up question to the hardness for fixed alphabets is whether polynomial-time solvability is possible if instead the cardinality of the nonterminal alphabet N is bounded. In this section, we answer this question in the affirmative by representing words $w\in\Sigma^*$ as graphs $\Phi_{\mathrm{m}}(w)$ and $\Phi_1(w)$, such that smallest independent dominating sets of these graphs correspond to a smallest grammar and a smallest 1-level grammar, respectively, for w.

We first define $\Phi_1(w)$ and then derive $\Phi_{\mathrm{m}}(w)$ from $\Phi_1(w)$. Let $\Phi_1(w)=(V,E)$ be defined by $V=V_1\cup V_2\cup V_3$ and $E=E_1\cup E_2\cup E_3$, where

$$V_1=\{(i,j)\colon 1\le i\le j\le|w|\},\qquad E_1=\{\{(i_1,j_1),(i_2,j_2)\}\colon i_1\le i_2\le j_1\},$$
$$V_2=\mathsf{F}_{\ge 2}(w),\qquad E_2=\{\{w[i..j],(i,j)\}\colon 1\le i<j\le|w|\},$$
$$V_3=\{(u,i)\colon u\in V_2,0\le i\le|u|\},\qquad E_3=\{\{u,(u,i)\}\colon u\in V_2,0\le i\le|u|\}.$$

Intuitively speaking, the vertices of V_1 represent every factor by its start and end position, whereas V_2 contains exactly one vertex per factor of length at least 2. Every $u\in V_2$ is connected to (i,j), if and only if $w[i..j]=u$. Vertices (i,j), (i',j') are connected if they refer to overlapping factors. For every $u\in V_2$, there are $|u|+1$ special vertices in V_3 that are only connected with u. Consequently, $\Phi_1(w)$ consists of $|w|$ layers, where the i^{th} layer contains the vertices $(j,j+(i-1))\in V_1$, $1\le j\le|w|-(i-1)$, the vertices $\{u\in V_2\colon|u|=i\}$ and the vertices $\{(u,j)\in V_3\colon|u|=i,0\le j\le|u|\}$ (see Figure 1 for an illustration).

▶ **Lemma 8.** *Let* $w\in\Sigma^*$, $k\ge 1$. *There is an independent dominating set* D *of cardinality* k *for* $\Phi_1(w)$ *if and only if there is a* 1*-level grammar* G *for* w *with* $|G|=k-|\,\mathsf{F}_{\ge 2}(w)|$.

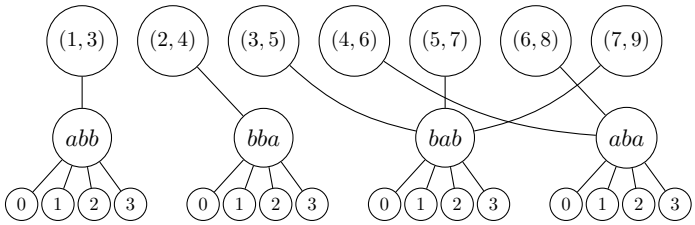


Figure 1 The third layer of $\Phi_1(abbababab)$ (edges E_1 omitted).

Proof Sketch. For an independent dominating set D of $\Phi_1(w)$, $V_1 \cap D$ induces a factorisation of w. For every $(i,j) \in D$, $w[i..j] \notin D$, which implies that all $|w[i..j]|+1$ many V_3-neighbours of $w[i..j]$ are in D. Now a 1-level grammar can be obtained by constructing rules for all $V_2 \setminus D$. Analogously, a 1-level grammar translates into an independent dominating set. ◀

In order to extend this idea to the multi-level case, what comes to mind is to somehow represent the vertices $u \in V_2$ again by graph structures of the type $\Phi_1(u)$ and repeating this step, which considerably increases the size of the graph. Fortunately, it turns out that a surprisingly simple modification of $\Phi_1(w)$ is sufficient. For a word $w \in \Sigma^*$, let $\Phi_{\mathrm{m}}(w) = (V, E)$ be defined as follows. Let $V = V_1 \cup V_2 \cup V_3 \cup V_4$, where V_1 and V_2 are defined as for $\Phi_1(w)$, $V_3 = \{(u,0) \colon u \in V_2\}$ and $V_4 = \bigcup_{u \in V_2} V_{4,u}$ with $V_{4,u} = \{(u,i,j) \colon 1 \le i \le j \le |u|, u[i..j] \neq u\}$ for every $u \in V_2$. Moreover, $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$, where E_1 and E_2 are defined as for $\Phi_1(w)$, $E_3 = \{\{u,(u,0)\} \colon u \in V_2\} \cup \{\{u,(u,i,j)\} \colon u \in V_2, (u,i,j) \in V_{4,u}\}$, $E_4 = \bigcup_{u \in V_2} E_{4,u}$, where, for every $u \in V_2$, $E_{4,u} = \{\{(u,i_1,j_1),(u,i_2,j_2)\} \subseteq V_{4,u} \colon i_1 \le i_2 \le j_1\}$ and $E_5 = \{\{u,(v,i,j)\} \colon u,v \in V_2, v[i..j] = u, u \neq v\}$.

Intuitively speaking, $\Phi_{\mathrm{m}}(w)$ differs from $\Phi_1(w)$ in the following way. We add to every vertex $u \in V_2$ a subgraph $(V_{4,u}, E_{4,u})$, which is completely connected to u and which represents u in the same way as the subgraph (V_1, E_1) of $\Phi_1(w)$ represents w, i. e., factors $u[i..j]$ are represented by (u,i,j) and edges represent overlappings. Moreover, if a $u \in V_2$ is a factor of some $v \in V_2$, then there is an edge from u to all the vertices $(v,i,j) \in V_{4,v}$ that satisfy $v[i..j] = u$. Finally, every $u \in V_2$ is also connected with an otherwise isolated vertex $(u,0) \in V_3$. See Figure 2 for a partial illustration of a $\Phi_{\mathrm{m}}(w)$.

▶ **Lemma 9.** *Let $w \in \Sigma^*$, $k \ge 1$. There is an independent dominating set D of cardinality k for $\Phi_{\mathrm{m}}(w)$ if and only if there is a grammar G for w with $|G| = k - |\,\mathsf{F}_{\ge 2}(w)|$.*

Proof Sketch. The correspondence of independent dominating sets D for $\Phi_{\mathrm{m}}(w)$ and grammars for w is similar as in the 1-level case. Again, $D \cap V_1$ induces a factorisation of w, and, in the same way, for every $u \in V_2 \setminus D$ (i. e., the factors for which rules will be constructed), $D \cap V_{4,u}$ induces a factorisation of u. Each $(v,i,j) \in D \cap V_{4,u}$ with $|v| \ge 2$ is connected to $v \in V_2$, which implies that $v \notin D$; thus, v will also be represented by a rule and so on. ◀

The proofs of Lemmas 8 and 9 also show how an independent dominating set D of $\Phi(w) \in \{\Phi_1(w), \Phi_{\mathrm{m}}(w)\}$ translates into a grammar for w, which, in the following, we will denote by $\mathsf{G}(D)$. Consequently, we can solve the smallest grammar problem by computing minimal independent dominating sets. Unfortunately, this is a hard problem, even for quite restricted graph classes [15, Theorem 13]. However, $\Phi(w)$ may have structural features that could be exploited in this regard, e. g., it is a 2-interval graph (see [8]).

Our algorithmic application is based on the following observation. If we are looking for a grammar $G = (N, \Sigma, R, cs)$ with $\{\mathfrak{D}(A) \colon A \in N\} = F$, for some set $F \subseteq \mathsf{F}_{\ge 2}(w)$, then we need an independent dominating set D with $(\mathsf{F}_{\ge 2}(w) \setminus F) \subseteq D$ and $F \cap D = \emptyset$. Obviously,

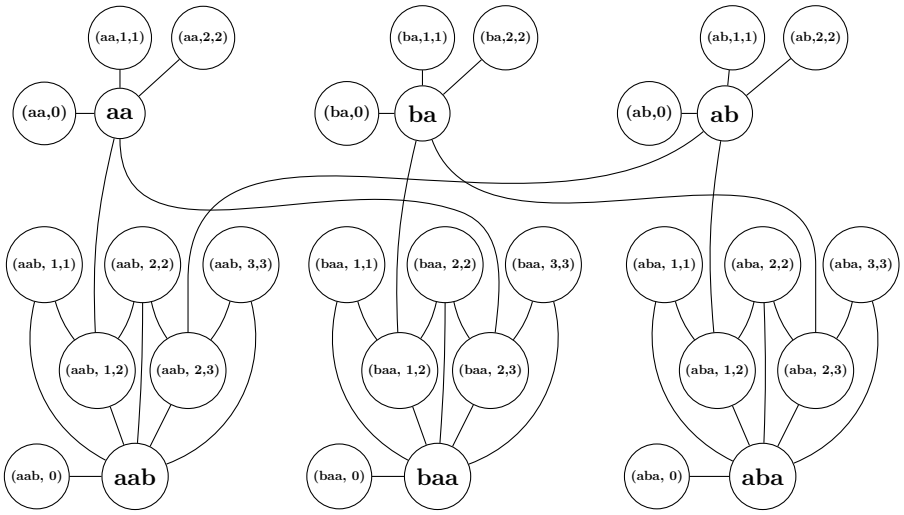


Figure 2 Second and third layer of $\Phi_{\mathrm{m}}(abaabaa)$ (vertices V_1 and edges $E_1 \cup E_2$ omitted).

D is the disjoint union of $(\mathsf{F}_{\geq 2}(w) \setminus F)$ and an independent dominating set D' for the graph $\mathcal{H} = \Phi(w) \setminus (N[\mathsf{F}_{\geq 2}(w) \setminus F] \cup F)$,[7] which is necessarily an interval graph; thus, a smallest independent dominating set for it can be efficiently computed (see [6]). For a word w and a set $F \subseteq \mathsf{F}_{\geq 2}(w)$, we define $\mathsf{MinIDS}(w, F) = D_{\mathcal{H}} \cup (\mathsf{F}_{\geq 2}(w) \setminus F)$, where $D_{\mathcal{H}}$ is a smallest independent dominating set for $\mathcal{H} = \Phi_{\mathrm{m}}(w) \setminus (N[\mathsf{F}_{\geq 2}(w) \setminus F] \cup F)$, and $\mathsf{1L\text{-}MinIDS}(w, F)$ is defined analogously by using $\Phi_1(w)$ instead of $\Phi_{\mathrm{m}}(w)$. In this way, for any set $F \subseteq \mathsf{F}_{\geq 2}$, we can compute a grammar that is minimal among all grammars that have rules for exactly the factors in F (this is also an interesting problem in its own right, e. g., if a compressed word is extended by a new part that should be compressed by already existing rules).

▶ **Lemma 10.** *Let $w \in \Sigma^+$ and $F \subseteq \mathsf{F}_{\geq 2}(w)$. $\mathsf{MinIDS}(w, F)$ and $\mathsf{1L\text{-}MinIDS}(w, F)$ can be computed in time $\mathcal{O}(|w|^6)$ and $\mathcal{O}(|w|^4)$, respectively. Furthermore, $\mathsf{G}(\mathsf{MinIDS}(w, F))$ is a minimal grammar for w and $\mathsf{G}(\mathsf{1L\text{-}MinIDS}(w, F))$ is a minimal 1-level grammar for w.*

If instead of a set F of factors, we are only given an upper bound k on $|N|$, then we can compute a minimal grammar by enumerating all $F \subseteq \mathsf{F}_{\geq 2}(w)$ with $|F| \leq k$ and computing $\mathsf{G}(\mathsf{MinIDS}(w, F))$. This shows that minimal grammars can be computed in polynomial time if the number of nonterminals is bounded.

▶ **Theorem 11.** *Let $w \in \Sigma^*$ and $k \in \mathbb{N}$. A grammar (1-level grammar, resp.) for w with at most k rules that is minimal among all grammars (1-level grammars, resp.) for w with at most k rules can be computed in time $\mathcal{O}(|w|^{2k+6})$ ($\mathcal{O}(|w|^{2k+4})$, resp.).*

The next question is whether these problems are also fixed-parameter tractable with respect to the number of nonterminals.[8] Unfortunately, this seems unlikely, since, as stated by the next result, these parameterisations of 1-SGP and SGP are $\mathsf{W}[1]$-hard.

▶ **Theorem 12.** 1-SGP *and* SGP *parameterised by $|N|$ are $\mathsf{W}[1]$-hard.*

[7] $N[v]$ is the closed neighbourhood of vertex v and for $C \subseteq V$, $N[C] = \bigcup_{v \in C} N[v]$.

[8] For unexplained concepts of parameterised complexity, we refer to Downey and Fellows [5].

This can be proven by reducing from independent set, i. e., a graph $\mathcal{G} = (\{v_1, v_2, \ldots, v_n\}, E)$ and $k \in \mathbb{N}$ is transformed into the word $w = \prod_{\{v_i,v_j\}\in E}(\#v_i\#v_j\#\diamond) \prod_{i=1}^{n}(\#v_i\#\diamond)^{n-|N(v_i)|}$, where $N(v_i)$ is the neighbourhood of v_i and every occurrence of $\diamond$ stands for a distinct symbol. To see the the correctness of this reduction, it is sufficient to observe that the vertices v_i of an independent set for G correspond to the rules of a grammar of form $A_i \to \#v_i\#$.

5 Exact Exponential Time Algorithms

Computing $\mathsf{G}(\mathsf{MinIDS}(w, F))$, for all $F \subseteq \mathsf{F}_{\geq 2}(w)$, yields a simple brute-force algorithm with a running time in $\mathcal{O}(2^{|w|^2})$. Another obvious approach is to enumerate all ordered trees with $|w|$ leaves (for each such tree T, an optimal grammar whose derivation tree has the structure T can be easily computed), which can be done in time $\mathcal{O}(8^{|w|})$.[9] In the following, we shall give more sophisticated exact exponential-time algorithms with running times in $\mathcal{O}^*(1.8392^{|w|})$, for the 1-level case, and $\mathcal{O}^*(3^{|w|})$, for the multi-level case.

Let $G = (N, \Sigma, R, cs)$ be a grammar for w and let $\alpha = A_1 \ldots A_k$, $A_i \in (\Sigma \cup N)$, $1 \leq i \leq k$. The *factorisation of* $\mathfrak{D}(\alpha)$ *induced by* α is the tuple $(\mathfrak{D}_G(A_1), \ldots, \mathfrak{D}_G(A_k))$. Furthermore, the factorisation of w induced by cs is called the *factorisation of* w *induced by* G.

5.1 The 1-Level Case

Let $q = (u_1, u_2, \ldots, u_k)$ be a factorisation for a word w and let $\Gamma_q = \{u_i \colon 1 \leq i \leq k, |u_i| \geq 2\}$ and let the 1-level grammar $G_q = (N_q, \Sigma, R_q, cs_q)$ be defined by $R_q = \{(A_u, u) \colon u \in \Gamma_q\}$, $N_q = \{A_u \colon u \in \Gamma_q\}$ and $cs_q = B_1 \ldots B_k$ with $B_j = A_{u_j}$, if $u_j \in \Gamma_q$ and $B_j = u_j$, otherwise.

▶ **Lemma 13.** *For any factorisation $q = (u_1, u_2, \ldots, u_k)$ for w, G_q is minimal among all 1-level grammars for w that induce the factorisation q.*

Choosing the smallest among all grammars $\{G_q \colon q \text{ is a factorisation of } w\}$ yields an $\mathcal{O}^*(2^n)$ algorithm for 1-SGP. However, it is not necessary to enumerate factorisations that contain at least two consecutive factors of length 1, which improves this result as follows.

▶ **Theorem 14.** 1-SGP *can be solved exactly in polynomial space and in time $\mathcal{O}^*(1.8392^{|w|})$.*

5.2 The Multi-Level Case

The obvious idea for a dynamic programming algorithm is to extend a smallest i-level grammar by a new level in order to obtain a smallest $(i+1)$-level grammar. However, this approach does not seem to work if we take the levels of a grammar to be $cs, \mathsf{D}(cs), \mathsf{D}(\mathsf{D}(cs)), \ldots, w$ (note that these are also the levels of the derivation tree). Intuitively speaking, the problem is that if we try to either add a new level on top (i. e., a new compressed string) of the grammar or at the bottom (by further compressing the terminal right sides of the last rules applied), then this decision is not local, since it is possible that rules to be added are already used somewhere else in the grammar. So we need to define levels in such a way that all occurrences of a nonterminal are on the same level.

For a d-level grammar $G = (N, \Sigma, R, cs)$, let $N_1, \ldots, N_d$ be the partition of N into $N_i = \{A \in N \colon (\mathsf{D}_G^i(A) \in \Sigma^+) \wedge (\mathsf{D}_G^{i-1}(A) \notin \Sigma^+)\}$ and let $\mathsf{L}_i \colon (N \cup \Sigma)^* \to (N \cup \Sigma)^*$, $1 \leq i \leq d$, be component-wise defined by $\mathsf{L}_i(x) = \mathsf{D}(x)$, if $x \in N_i$ and $\mathsf{L}_i(x) = x$, otherwise.

[9] There are $C_{|w|-1} \leq 4^{|w|-1}$ ordered binary trees with $|w|$ leaves, where $C_{|w|-1}$ is the $(|w|-1)^{\text{th}}$ Catalan number, and every ordered tree can be obtained from a binary one by contracting some of its edges.

Taking the strings $(\mathsf{L}_{i+1} \circ \mathsf{L}_{i+2} \circ \cdots \circ \mathsf{L}_d)(cs)$, which contain all occurrences of nonterminals N_i, as the levels of the grammar, we are able to define a dynamic programming algorithm.[10]

▶ **Theorem 15.** SGP *can be solved exactly in time and space* $\mathcal{O}^*(3^{|w|})$.

Proof Sketch. With the help of the mappings L_i, we can define the term *profit* for rules from a d-level grammar $G = (N, \Sigma, R, cs)$ as follows. The profit for a rule $A \to \alpha$ with $A \in N_d$ can be defined like in the 1-level case, i. e., $\mathsf{p}(A) = |cs|_A(|\alpha| - 1) - |\alpha|$, considering that removing this rule and replacing each occurrence of A in cs by α increases the size of the grammar by $|cs|_A(|\alpha| - 1) - |\alpha|$. Inductive use of this argument allows us to define the profit of any rule $A \to \alpha$ with $A \in N_i$ by $\mathsf{p}(A) := |(\mathsf{L}_{i+1} \circ \mathsf{L}_{i+2} \circ \cdots \circ \mathsf{L}_d)(cs)|_A(|\alpha| - 1) - |\alpha|$. This allows us to compute the size of a G by $|w| - \sum_{A \in N} \mathsf{p}(A)$. The dynamic programming algorithm runs trough steps $i = 1, 2, \ldots, \frac{w}{2}$ and in step i, it considers all possibilities for two factorisations q_{i-1} and q_i of w induced by $(\mathsf{L}_i \circ \mathsf{L}_{i+1} \circ \cdots \circ \mathsf{L}_d(cs)$ and $(\mathsf{L}_{i+1} \circ \cdots \circ \mathsf{L}_d)(cs)$, respectively (note that this implies $q_{i-1} \preceq q_i$). The differences between q_{i-1} and q_i implicitly define N_i. Let $q_i = (v_1, v_2, \ldots, v_k)$ and let $q_{i-1} = (u_1, u_2, \ldots, u_\ell)$, i. e., for some j_i, $0 \leq i \leq k$, with $1 = j_0 < j_1 < \ldots < j_k = \ell + 1$, $(u_{j_{i-1}}, u_{j_{i-1}+1}, \ldots, u_{j_i-1})$ is a factorisation of v_i, $1 \leq i \leq k$. If $j_s - j_{s-1} > 1$ for some $1 \leq s \leq k$, N_i contains a nonterminal A with $|\,\mathsf{D}(A)| = j_s - j_{s-1}$ and $\mathfrak{D}(A) = v_s$. The term $|\,\mathsf{L}_i \circ \mathsf{L}_{i+1} \circ \cdots \circ \mathsf{L}_d)(cs)|_A$ is also implicitly given by counting how often the sequence of factors $(u_{j_{s-1}+1}, \ldots, u_{j_s})$ independently occurs in q_{i-1} and is combined into one single factor in q_i, i.e.: $|\{t\colon (u_{j_{t-1}+1}, \ldots, u_{j_t}) = (u_{j_{s-1}+1}, \ldots, u_{j_s})\}|$. This allows to calculate the profit of the rule for A without knowing the exact structure of the rules for nonterminals in N_j with $j \neq i$. By Lemma 13, this choice of nonterminals for N_i is optimal for the fixed induced factorisations, which means that a search among all choices for q_{i-1} and q_i yields a minimal i-level grammar for w. The running time of this algorithm is dominated by enumerating all pairs q_{i-1} and q_i of factorisations of w. However, due to $q_{i-1} \preceq q_i$, these pairs can be compressed as vectors $\{0, 1, 2\}^{|w|-1}$ (the entries denote whether the corresponding position in w is factorised by both, only one or none of the factorisations). Hence, enumerating these pairs of vectors can be done in time $\mathcal{O}(3^{|w|})$. ◀

6 Conclusions

We conclude this work by deriving some parameterised complexity results.[11] The shortest-grammar problem (1-level and multi-level) is Para-NP-hard with respect to $|\Sigma|$, it is in XP with respect to $|N|$, but also W[1]-hard, so most likely not in FPT. Furthermore, the hardness of 1-SGP shows that bounding or parameterising by the number of levels does not help either. However, if we parameterise by both $|\Sigma|$ and $\ell = \max\{|\,\mathfrak{D}(A)|\colon A \in N\}$, then it is sufficient to compute $\mathsf{G}(\mathsf{MinIDS}(w, F))$ for every set $F \subseteq \{u\colon u \in \Sigma^+, |u| \leq \ell\}$, which, since the number of such sets is bounded by the parameters, yields an fpt-algorithm. A probably more interesting combination of parameters, for which the existence of an fpt-algorithm is still open, would be $|\Sigma|$ and $|N|$.

The most interesting question (also from a practical point of view) left open is whether it is possible to compute minimal grammars for small (especially binary) alphabets in polynomial-time. The substantial effort that was necessary to prove Theorem 7 suggests that answering this question in the negative might be difficult. On the other hand, it is not apparent how a small alphabet could help in order to efficiently compute smallest grammars and, if

[10] The composition $(f \circ g)$ of mapping $f : A \to A$, $g : A \to A$ is defined by $(f \circ g)(a) = f(g(a))$.
[11] For unexplained concepts of parameterised complexity, we refer to Downey and Fellows [5].

this is possible, it seems that deeper combinatorial insights with respect to grammar-based compression are necessary.

References

1 T. Akutsu. A bisection algorithm for grammar-based compression of ordered trees. *Information Processing Letters*, 110(18-19):815–820, 2010.

2 J. Arpe and R. Reischuk. On the complexity of optimal grammar-based compression. In *2006 Data Compression Conference (DCC)*, pages 173–182. IEEE Computer Society, 2006.

3 P. Berman, M. Karpinski, L. L. Larmore, W. Plandowski, and W. Rytter. On the complexity of pattern matching for highly compressed two-dimensional texts. *Journal of Computer and System Sciences*, 65(2):332–350, 2002.

4 M. Charikar, E. Lehman, D. Liu, R. Panigrahy, M. Prabhakaran, A. Sahai, and A. Shelat. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005.

5 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

6 M. Farber. Independent domination in chordal graphs. *Operations Research Letters*, 1(4):134–138, 1982.

7 M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.

8 M. Jiang and Y. Zhang. Parameterized complexity in multiple-interval graphs: Domination, partition, separation, irredundancy. *Theoretical Computer Science*, 461:27–44, 2012.

9 J. C. Kieffer and E.-H. Yang. Grammar-based codes: A new class of universal lossless source codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000.

10 J. C. Kieffer, E.-H. Yang, G. J. Nelson, and P. C. Cosman. Universal lossless compression via multilevel pattern matching. *IEEE Transactions on Information Theory*, 46(4):1227–1245, 2000.

11 M. Lohrey. Algorithmics on SLP-compressed strings: A survey. *Groups, Complexity, Cryptology*, 4:241–299, 2012.

12 M. Lohrey and S. Maneth. The complexity of tree automata and XPath on grammar-compressed trees. *Theoretical Computer Science*, 363(2):196–210, 2006.

13 M. Lohrey, S. Maneth, and R. Mennicke. XML tree structure compression using RePair. *Information Systems*, 38(8):1150–1167, 2013.

14 M. Lohrey, S. Maneth, and M. Schmidt-Schauß. Parameter reduction and automata evaluation for grammar-compressed trees. *Journal of Computer and System Sciences*, 78(5):1651–1669, 2012.

15 D. F. Manlove. On the algorithmic complexity of twelve covering and independence parameters of graphs. *Discrete Applied Mathematics*, 91:155–175, 1999.

16 C. G. Nevill-Manning. *Inferring Sequential Structure*. PhD thesis, University of Waikato, NZ, 1996.

17 C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67–82, 1997.

18 W. Rytter. Application of Lempel-Ziv factorization to the approximation of grammar-based compression. *Theoretical Computer Science*, 302:211–222, 2003.

19 C. E. Shannon. A theorem on coloring the lines of a network. *J. Math. Physics*, 28:148–151, 1949.

20 S. Skulrattanakulchai. Δ-list vertex coloring in linear time. *Information Processing Letters*, 98(3):101–106, 2006.

21 J. A. Storer. NP-completeness results concerning data compression. Technical Report 234, Dept. Electrical Engineering and Computer Science, Princeton University, USA, November 1977.

22 J. A. Storer and T. G. Szymanski. Data compression via textural substitution. *Journal of the ACM*, 29(4):928–951, 1982.

23 E.-H. Yang and J. C. Kieffer. Efficient universal lossless data compression algorithms based on a greedy sequential grammar transform - part one: Without context models. *IEEE Transactions on Information Theory*, 46(3):755–777, 2000.

The Complexity of Downward Closure Comparisons

Georg Zetzsche*

LSV, CNRS & ENS Cachan, Université Paris-Saclay, France
zetzsche@lsv.fr

Abstract

The downward closure of a language is the set of all (not necessarily contiguous) subwords of its members. It is well-known that the downward closure of every language is regular. Moreover, recent results show that downward closures are computable for quite powerful system models.

One advantage of abstracting a language by its downward closure is that then equivalence and inclusion become decidable. In this work, we study the complexity of these two problems. More precisely, we consider the following decision problems: Given languages K and L from classes $\mathcal{C}$ and $\mathcal{D}$, respectively, does the downward closure of K include (equal) that of L?

These problems are investigated for finite automata, one-counter automata, context-free grammars, and reversal-bounded counter automata. For each combination, we prove a completeness result either for fixed or for arbitrary alphabets. Moreover, for Petri net languages, we show that both problems are Ackermann-hard and for higher-order pushdown automata of order k, we prove hardness for complements of nondeterministic k-fold exponential time.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Downward closures, Complexity, Inclusion, Equivalence

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.123

1 Introduction

The downward closure of a language is the set of (not necessarily contiguous) subwords of its members. It is a well-known result of Haines [17] that the downward closure of *every* language is regular. Of course, it is not always possible to compute the downward closure of a given language, but oftentimes it is. For example, it has been shown to be computable for such powerful models as *Petri net languages* by Habermehl, Meyer, and Wimmel [14] and *higher-order pushdown automata* by Hague, Kochems, and Ong [15]. A sufficient condition for computability can be found in [34].

Moreover, not only are downward closures often computable, they are also a meaningful abstraction of infinite-state systems. In a complex system, one can abstract a component by the downward closure of the messages it sends to its environment. This corresponds to the assumption that messages can be dropped on the way. Furthermore, recent work of La Torre, Muscholl, and Walukiewicz [32] shows that among other mild conditions, computing downward closures is sufficient for verifying safety conditions of parametrized asynchronous shared-memory systems.

The advantage of having an abstraction of an infinite-state systems as regular languages is that the latter offer an abundance of methods for analysis. An important example is deciding

* This work is supported by a fellowship within the Postdoc-Program of the German Academic Exchange Service (DAAD).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 123; pp. 123:1–123:14

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

behavioral equivalence or inclusion. This is notoriously hard to do and for nondeterministic infinite-state systems, language equivalence and inclusion are usually undecidable. Using downward closures, such behavioral comparisons can be made in an approximative manner.

Despite these facts, results about the complexity of deciding whether the downward closure of one language includes or equals that of another mainly considered regular languages. Bachmeier, Luttenberger, and Schlund [4] have shown that the equivalence problem for downward closures of two given NFAs is coNP-complete. Karandikar, Niewerth, and Schnoebelen [22] strengthened coNP-hardness to the case of DFAs over binary alphabets and proved coNP-completeness for the inclusion variant. They also obtained NL-completeness of inclusion in the case of NFAs over a unary alphabet. Together with exponential-time downward closure constructions [4, 7, 11, 33, 27], these results imply that equivalence and inclusion are in coNEXP for context-free grammars. Rampersad, Shallit, and Xu [31] proved that one can decide in linear time whether the downward closure of a given NFA contains all words. Subsequently, Karandikar, Niewerth, and Schnoebelen [22] showed that this problem is NL-complete. Similar questions have been studied for upward closures [4, 22].

Previous work on downward closures of infinite-state systems has mainly focused on mere computability [1, 2, 7, 14, 15, 33, 34, 35] or on descriptional complexity [3, 10, 11, 27, 22]. This work studies the complexity of the inclusion and the equivalence problem of downward closures between some prominent types of system models—finite automata, one-counter automata, reversal-bounded counter automata [19], and context-free grammars. More precisely, we are interested in the following questions: For two system models $\mathcal{M}$ and $\mathcal{N}$ and languages L and K generated by some device in $\mathcal{M}$ and $\mathcal{N}$, respectively, what is the complexity of (i) deciding whether $K{\downarrow} \subseteq L{\downarrow}$ (*downward closure inclusion problem*) or (ii) deciding whether $K{\downarrow} = L{\downarrow}$ (*downward closure equivalence problem*)?

Contribution. We determine the complexity of the downward closure inclusion problem and the downward closure equivalence problem among finite automata, one-counter automata, reversal-bounded counter automata (either with a fixed number of counters and reversals or without), and context-free grammars.

For the inclusion problem, we prove completeness results in all cases except for two. The complexities range from coNP over Π_2^{P} to coNEXP (see Table 1). The two cases for which we provide no completeness compare context-free grammars or general reversal-bounded counter automata on the one side with reversal-bounded counter automata with a fixed number of counters and reversals on the other side. However, we prove that both of these problems are coNP-complete for each fixed input alphabet. For the equivalence problem, the situation is similar. We prove completeness for each of the cases except for the combination above. Again, fixing the alphabet leads to coNP-completeness.

The tools developed to achieve these results fall into three categories. First, there are several generic results guaranteeing small witnesses to yield upper bounds. Second, we prove model-specific results about downward closures that yield the upper bounds in each case. Third, we have a general method to prove lower bounds for downward closure comparisons. In fact, it applies to more models than the above: We prove that for Petri net languages, the two comparison problems are Ackermann-hard. For higher-order pushdown automata of order k, we show co-k-NEXP-hardness.

Related work. Another abstraction of formal languages is the well-known Parikh image [28]. The Parikh image of a language $L \subseteq X^*$ contains for each word $w \in L$ a vector in $\mathbb{N}^{|X|}$ that counts the number of occurrences of each letter. For some language classes, it is known that

Table 1 Complexity of the inclusion problem. The entry in row $\mathcal{M}$ and column $\mathcal{N}$ is the complexity of $\mathcal{M} \subseteq_{\downarrow} \mathcal{N}$. Except in the case Ideal $\subseteq_{\downarrow}$ Ideal, all entries indicate completeness. A † means that the entry refers to the fixed alphabet case (for at least two letters).

	Ideal	NFA	OCA	$\mathsf{RBC}_{k,r}$	CFG	RBC
Ideal	$\in$ L	NL	NL	NL	P	NP
NFA	NL	coNP [4, 22]	coNP [3, 4, 22]	coNP	coNP	Π_2^{P}
OCA	NL	coNP [3, 4, 22]	coNP [3, 4, 22]	coNP	coNP	Π_2^{P}
$\mathsf{RBC}_{k,r}$	NL	coNP	coNP	coNP	coNP	Π_2^{P}
CFG	P	coNP	coNP	coNP†	coNEXP	coNEXP
RBC	coNP	coNP	coNP	coNP†	coNEXP	coNEXP

their Parikh image is effectively semilinear, which implies decidability of the inclusion and equivalence problem for Parikh images. The investigation of these problems' complexity has been initiated by Huynh [18] in 1985, who showed that this problem is Π_2^{P}-hard and in coNEXP for regular and context-free languages. Kopczyński and To [23, 24] have then shown that these problems are Π_2^{P}-complete for fixed alphabets. Only very recently, Haase and Hofman [13] have shown that the case of general alphabets is coNEXP-complete.

Due to space restrictions, most proofs can only be found in the full version of this work [36].

2 Concepts and Results

If X is an alphabet, X^* ($X^{\leq n}$) denotes the set of all words (of length $\leq n$) over X. The empty word is denoted by $\varepsilon \in X^*$. For words $u, v \in X^*$, we write $u \preceq v$ if $u = u_1 \cdots u_n$ and $v = v_0 u_1 v_1 \cdots u_n v_n$ for some $u_1, \ldots, u_n, v_0, \ldots, v_n \in X^*$. It is well-known that $\preceq$ is a well-quasi-order on X^* and that therefore the *downward closure* $L{\downarrow} = \{u \in X^* \mid \exists v \in L \colon u \preceq v\}$ is regular for every $L \subseteq X^*$ [17]. An *ideal* is a set of the form $Y_0^* \{x_1, \varepsilon\} Y_1^* \cdots \{x_n, \varepsilon\} Y_n^*$, where $Y_0, \ldots, Y_n$ are alphabets and $x_1, \ldots, x_n$ are letters. We will make heavy use of the fact that every downward closed language can be written as a finite union of ideals, which was first discovered by Jullien [21]. By $\mathbb{P}(S)$, we denote the powerset of the set S.

A *finite automaton* is a tuple $\mathcal{A} = (Q, X, \Delta, q_0, Q_f)$, where Q is a finite set of *states*, X is its input alphabet, $\Delta \subseteq Q \times X^* \times Q$ is a finite set of *edges*, $q_0 \in Q$ is its *initial state*, and $Q_f \subseteq Q$ is the set of its *final states*. The language accepted by $\mathcal{A}$ is denoted $L(\mathcal{A})$. Sometimes, we write $|\mathcal{A}|$ for the number of states of $\mathcal{A}$.

A *context-free grammar* is a tuple $\mathcal{G} = (N, T, P, S)$ where N and T are pairwise disjoint alphabets, whose members are called the *nonterminals* and *terminals*, respectively. $S \in N$ is the *start symbol* and P is the finite set of *productions* of the form $A \to w$ with $A \in N$ and $w \in T^*$. The language generated by $\mathcal{G}$ is defined as usual.

One-counter Automata. A *one-counter automaton (OCA)* is a nondeterministic finite automaton that has access to one counter that assumes natural numbers as values. The possible operations are *increment*, *decrement*, and *test for zero*. We will not require a formal definition, since in fact, all we need is the well-known fact that membership and emptiness are NL-complete and the recent result that given an OCA $\mathcal{A}$, one can compute in polynomial time an NFA $\mathcal{B}$ with $L(\mathcal{B}) = L(\mathcal{A}){\downarrow}$ [3].

Reversal-bounded counter automata. Intuitively, an *r-reversal-bounded k-counter automaton* [19] (short (k,r)-RBCA) is a nondeterministic finite automaton with k counters that can store natural numbers. For each counter, it has operations *increment*, *decrement*, and *zero test*. Moreover, a computation is only valid if each counter *reverses* at most r times. Here, a computation *reverses* a counter c if on c, it first executes a sequence of increments and then a decrement command or vice versa. See [19] for details.

Instead of working directly with RBCA, we will work here with the model of *blind counter automata* [9]. It is not as well-known as RBCA, but simpler and directly amenable to linear algebraic methods. A *blind k-counter automaton* is a tuple $\mathcal{A} = (Q, X, q_0, \Delta, Q_f)$, where Q, X, q_0, and Q_f are defined as in NFAs, but Δ is a finite subset of $Q \times (X \cup \{\varepsilon\}) \times \{-1,0,1\}^k \times Q$. A *walk* is a word $\delta_1 \cdots \delta_m \in \Delta^*$ where $\delta_i = (p_i, x_i, d_i, p'_i)$ for $i \in [1,m]$ and $p'_j = p_{j+1}$ for $j \in [1, m-1]$. The *effect* of the walk is $d_1 + \cdots + d_m$. Its *input* is $x_1 \cdots x_m \in X^*$. If the walk has effect 0 and $p_0 = q_0$ and $p_m \in Q_f$, then the walk is *accepting*. The *language accepted by* $\mathcal{A}$ is the set of all inputs of accepting walks.

Using blind counter automata is justified because to each (k,r)-RBCA, one can construct in logarithmic space a language-equivalent $(kr,1)$-RBCA [5], which is essentially a blind kr-counter automaton. On the other hand, every blind k-counter automaton can be turned in logarithmic space into a $(k+1,1)$-RBCA [20]. Hence, decision problems about (k,r)-RBCA for fixed k and r correspond to problems about blind k-counter automata for fixed k.

In the following, by a *model*, we mean a way of specifying a language. In order to succinctly refer to the different decision problems, we use symbols for the models above. By Ideal, NFA, OCA, $\mathsf{RBC}_{k,r}$, RBC, CFG, we mean ideals, finite automata, OCA, RBCA with a fixed number of counters and reversals, general RBCA, and context-free grammars, respectively. Then, for $\mathcal{M}, \mathcal{N} \in \{\mathsf{Ideal}, \mathsf{NFA}, \mathsf{OCA}, \mathsf{RBC}_{k,r}, \mathsf{RBC}, \mathsf{CFG}\}$, we consider the following problems. In the *downward closure inclusion problem* $\mathcal{M} \subseteq_\downarrow \mathcal{N}$, we are given a language K in $\mathcal{M}$ and a language L in $\mathcal{N}$ and are asked whether $K{\downarrow} \subseteq L{\downarrow}$. For the *downward closure equivalence problem* $\mathcal{M} =_\downarrow \mathcal{N}$, the input is the same, but we are asked whether $K{\downarrow} = L{\downarrow}$.

Results. The complexity results for the inclusion problem are summarized in Table 1. For the equivalence problem, we will see that every hardness result for $\mathcal{M} \subseteq_\downarrow \mathcal{N}$ also holds for $\mathcal{M} =_\downarrow \mathcal{N}$. Since for non-ideal models, the appearing complexity classes are pairwise comparable, this implies that the complexity for $\mathcal{M} =_\downarrow \mathcal{N}$ is then the harder of the two classes for $\mathcal{M} \subseteq_\downarrow \mathcal{N}$ and $\mathcal{N} \subseteq_\downarrow \mathcal{M}$. For example, the problem $\mathsf{NFA} =_\downarrow \mathsf{RBC}$ is Π_2^P-complete and for fixed alphabets, $\mathsf{RBC}_{k,r} =_\downarrow \mathsf{CFG}$ is coNP-complete.

3 Ideals and Witnesses

Our algorithms for inclusion use three types of witnesses. The first type is a slight variation of a result of [4]. The latter authors were interested in equivalence problems, which caused their bound to depend on both input languages. The proof is essentially the same.

▶ **Proposition 1** (Short witness). *If $\mathcal{A}$ is an NFA and $K{\downarrow} \not\subseteq L(\mathcal{A}){\downarrow}$, then there exists a $w \in K{\downarrow} \setminus L(\mathcal{A}){\downarrow}$ with $|w| \leq |\mathcal{A}| + 1$.*

The other types of witnesses strongly rely on ideals, which requires some notation. An ideal is a product $I = Y_0^*\{x_1, \varepsilon\}Y_1^* \cdots \{x_n, \varepsilon\}Y_n^*$ where the Y_i are alphabets and the x_i are letters. Its *length* $|I|_\mathsf{I}$ is the smallest n such that I can be written in this form. Since every downward closed language can be written as a finite union of ideals, we can extend this definition to languages: $|L|_\mathsf{I}$ is the smallest n such that $L{\downarrow}$ is a union of ideals of length $\leq n$.

Sometimes, it will be convenient to work with a different length measure of ideals. An *ideal expression (of length n)* is a product $L_1 \cdots L_n$, where each L_i is of the form Y^* or $\{x, \varepsilon\}$, where Y is an alphabet and x is a letter. Note that $Y^* = Y^*\{x, \varepsilon\}$ if $x \in Y$ and $\{x, \varepsilon\} = \emptyset^*\{x, \varepsilon\}$. Therefore, an ideal expression of length n defines an ideal of length $\leq n$. In analogy to $|\cdot|_{\mathsf{I}}$, for a language L, we define its *expression length* $|L|_{\mathsf{E}}$ to be the smallest n such that $L{\downarrow}$ can be written as a finite union of ideal expressions of length $\leq n$. The expression length has the advantage of being subadditive: For languages K, L we have $|KL|_{\mathsf{E}} \leq |K|_{\mathsf{E}} + |L|_{\mathsf{E}}$. Moreover, we have $|L|_{\mathsf{I}} \leq |L|_{\mathsf{E}} \leq 2|L|_{\mathsf{I}} + 1$.

The measure $|\cdot|_{\mathsf{I}}$ turns out to be instrumental for the inclusion problem. Note that $K{\downarrow} \not\subseteq L{\downarrow}$ if and only if there is an ideal $I \subseteq K{\downarrow}$ of length $\leq |K|_{\mathsf{I}}$ with $I \not\subseteq L{\downarrow}$. We can therefore guess ideals and check inclusion for them. From now on, we assume alphabets to come linearly ordered. This means for every alphabet Y, there is a canonical word w_Y in which every letter from Y occurs exactly once.

▶ **Proposition 2** (Ideal witness). *Let $I = Y_0^*\{x_1, \varepsilon\}Y_1^* \cdots \{x_n, \varepsilon\}Y_n^*$. Then the following are equivalent:* (i) *$I \subseteq L{\downarrow}$.* (ii) *$w_{Y_0}^m x_1 w_{Y_1}^m \cdots x_n w_{Y_n}^m \in L{\downarrow}$ for every $m \geq |L|_{\mathsf{I}} + 1$.* (iii) *$w_{Y_0}^m x_1 w_{Y_1}^m \cdots x_n w_{Y_n}^m \in L{\downarrow}$ for some $m \geq |L|_{\mathsf{I}} + 1$.*

A word of the form $w_{Y_0}^m x_1 w_{Y_1}^m \cdots x_n w_{Y_n}^m \in L{\downarrow}$ with $m \geq |L|_{\mathsf{I}} + 1$ is therefore called an *ideal witness for I and L*. The proof of Proposition 2 is a simple pumping argument based on the fact that an ideal of length $\leq m$ admits an NFA with $\leq m + 1$ states. Ideal witnesses are useful when we have a small bound on $|K|_{\mathsf{I}}$ and $|L|_{\mathsf{I}}$ but only a large bound on the NFA size of $L{\downarrow}$. Observe that putting a bound on $|L|_{\mathsf{I}}$ amounts to proving a pumping lemma: We have $|L|_{\mathsf{I}} \leq n$ if and only if for every $w \in L$, there is an ideal I with $|I|_{\mathsf{I}} \leq n$ and $x \in I \subseteq L{\downarrow}$.

However even if, say, $|K|_{\mathsf{I}}$ is polynomial and $|L|_{\mathsf{I}}$ is exponential, ideal witnesses can be stored succinctly in polynomial space, by keeping a binary representation of the power m. For instance, this will be used in the case $\mathsf{NFA} \subseteq_{\downarrow} \mathsf{RBC}$.

Sometimes, we have a small bound on $|L|_{\mathsf{I}}$, but $|K|_{\mathsf{I}}$ may be large. Then, ideal witnesses are too large to achieve an optimal algorithm. In these situations, we can guarantee smaller witnesses if we fix the alphabet.

▶ **Proposition 3** (Small alphabet witness). *Let $K, L \subseteq X^*$. If $K{\downarrow} \not\subseteq L{\downarrow}$, then there exists a $w \in K{\downarrow} \setminus L{\downarrow}$ with $|w| \leq |X| \cdot (|L|_{\mathsf{I}} + 1)^{|X|}$.*

The proof of Proposition 3 is more involved than Propositions 2 and 1. Note that a naive bound can be obtained by intersecting exponentially (in $|L|_{\mathsf{I}}$) many automata for the ideals of $L{\downarrow}$ and complementing the result. This would yield a doubly exponential (in $|L|_{\mathsf{I}}$) bound, even considering the fact that ideals have linear-size DFAs. We can, however, use the latter fact in a different way.

A DFA is *ordered* if its states can be partially ordered so that for every transition $p \xrightarrow{x} q$, we have $p \leq q$. In other words, the automaton is acyclic except for loop transitions. The following lemma is easy to see: In order to check membership in an ideal, one just has to keep a pointer into the expression that never moves left.

▶ **Lemma 4.** *Given an ideal representation of length n, one can construct in logarithmic space an equivalent ordered DFA with $n + 2$ states.*

An ordered DFA *cycles* at a position of an input word if that position is read using a loop. The following lemma is the key idea behind Proposition 3. Together with Lemma 4, it clearly implies Proposition 3. For unary alphabets, it is easy to see. We use induction on $|X|$ and show, roughly speaking, that without such a position, no strict subalphabet can be

used for too long. Then, all letters have to appear often, meaning a state has to repeat after seeing the whole alphabet. Hence, the automaton stays in this state until the end.

▶ **Lemma 5.** *If $w \in X^*$ with $|w| > |X| \cdot (n-1)^{|X|}$, then w has a position at which every ordered n-state DFA cycles.*

4 Insertion trees

In Section 5, we will show upper bounds for the size of downward closure NFAs and for ideal lengths for counter automata. These results employ certain decompositions of NFA runs into trees, which we discuss here. Let $\mathcal{A} = (Q, X, \Delta, q_0, Q_f)$ be a finite automaton. A *walk* is a word $w = \delta_1 \cdots \delta_m \in \Delta^*$ where $\delta_i = (p_i, x_i, p'_i)$ for $i \in [1, m]$ and $p'_j = p_{j+1}$ for $j \in [1, m-1]$. The walk is a *(p_1-)cycle* if $p_1 = p'_m$. In this case, we define $\sigma(w) := p_1$. A cycle is *prime* if $p_i = p_1$ implies $i = 1$. A cycle is *simple* if $p_i = p_j$ implies $i = j$. A state q *occurs* on the cycle if $p_i = q$ for some i. If $i \neq 1$, then q occurs *properly*.

A common operation in automata theory is to take a run and delete cycles until the run has length at most $|Q|$. The idea behind an insertion tree is to record where we deleted which cycles. This naturally leads to a tree.

For our purposes, trees are finite, unranked and ordered. An *insertion tree* is a tree $t = (V, E)$ together with a map $\gamma\colon V \to \Delta^*$ that assigns to each vertex $v \in V$ a simple cycle $\gamma(v)$ such that if u is the parent of v, then $\sigma(\gamma(v))$ properly occurs in $\gamma(u)$. Note that we allow multiple children for a state that occurs in $\gamma(u)$.

Since t is ordered and in every simple cycle there is at most one proper occurrence of each state, an insertion tree defines a unique (typically not simple) cycle $\alpha(t)$. Formally, if t is a single vertex v, then $\alpha(t) := \gamma(v)$. If t consists of a root r and subtrees $t_1, \ldots, t_s$, then $\alpha(t)$ is obtained by inserting each $\alpha(t_i)$ in $\gamma(r)$ at the (unique) occurrence of $\sigma(\alpha(t_i))$. The *height* of an insertion tree is the height of its tree.

▶ **Lemma 6.** *Every prime cycle of $\mathcal{A}$ admits an insertion tree of height at most $|Q|$.*

The idea is to pick a cycle c strictly contained in the prime cycle, but of maximal length. Then, after removing c, no state occurs both before and after the old position of c. This forces any insertion tree t of the remainder to place this position in the root. We then apply induction to the subtrees of t and to c. The resulting trees can then all be attached to the root, increasing the height by at most one.

One application of Lemma 6 is to construct short ideals in a pumping lemma for counter automata. Part of this construction is independent from counters, so we stay with NFAs for a moment. Suppose we have an insertion tree $t = (V, E)$ with map $\gamma\colon V \to \Delta^*$ and a subset $F \subseteq V$, whose members we call *fixed vertices* or *fixed cycles*. Those in $V \setminus F$ are called *pumpable vertices/cycles*.

We use fixed and pumpable vertices to guide a pumping process as follows. A sequence $s = t_1 \cdots t_m$ of insertion trees is called *compatible* if $\sigma(\alpha(t_1)) = \cdots = \sigma(\alpha(t_m))$. We assume that we have a global set F of vertices that designates the fixed vertices for all these trees. Suppose v is a pumpable vertex. We obtain new compatible sequences in two ways:

- Let $v_1, \ldots, v_\ell$ be the children of v. We choose $i \in [0, \ell]$ and split up v at i, meaning that we create a new vertex v' with $\gamma(v') = \gamma(v)$ to the right of v and move $v_{i+1}, \ldots, v_\ell$ (and, of course, their subtrees) to v'.
- If the whole subtree under v is pumpable (we call such subtrees *pumpable*), then we can duplicate this subtree and attach its root somewhere as a sibling of v.

If v is a root, these operations mean that we introduce a new tree in the sequence. If a compatible sequence s' is obtained from s by repeatedly performing these operations, we say that s' is obtained by *pumping* s. This allows us to define the following language:

$$P(t_1 \cdots t_m, F) = \{\iota(\alpha(t'_1) \cdots \alpha(t'_k)) \mid t'_1 \cdots t'_k \text{ results from pumping } t_1 \cdots t_m\}.$$

Here, for a walk w, $\iota(w)$ denotes the input word read by w. The following lemma will yield the desired short ideals.

▶ **Lemma 7.** *Let $s = t_1 \cdots t_m$ be a compatible sequence of insertion trees of height $\leq h$ and let F be a set of fixed vertices. Then, the language $P(s, F)\downarrow$ is an ideal that satisfies $|P(s, F)\downarrow|_{\mathsf{E}} \leq h|F|(2|Q| + |F|)^2$.*

Roughly speaking, the pumping process is designed so that pumpable subtrees only cause alphabets Y in factors Y^* of the ideal to grow and thus do not affect the ideal length. Hence, the only vertices that contribute to the length are those that are ancestors of vertices in F. Since the trees have height $\leq h$, there are at most $h|F|$ such ancestors.

5 Counter Automata

In this section, we construct downward closure NFAs for counter automata and prove upper bounds for ideal lengths. Mere computability of downward closures of blind counter automata can be deduced from computability for Petri net languages [14]. However, that necessarily results in non-primitive recursive automata (see Section 8). As a special case of stacked counter automata, blind counter automata were provided with a new construction method in [35]. That algorithm, however, yields automata of non-elementary size. Here, we prove an exponential bound.

▶ **Theorem 8.** *For each n-state blind k-counter automaton $\mathcal{A}$, there is an NFA $\mathcal{B}$ with $L(\mathcal{B}) = L(\mathcal{A})\downarrow$ and $|\mathcal{B}| \leq (3n)^{5nk+7k^3}$. Moreover, $\mathcal{B}$ can be computed in exponential time.*

Linear Diophantine equations. In order to show correctness of our construction, we employ a result of Pottier [29], which bounds the norm of minimal non-negative solutions to a linear Diophantine equation. Let $A \in \mathbb{Z}^{k\times m}$ be an integer matrix. We write $\|A\|_{1,\infty}$ for $\sup_{i\in[1,k]}(\sum_{j\in[1,m]} |a_{ij}|)$, where a_{ij} is the entry of A at row i and column j. A solution $x \in \mathbb{N}^m$ to the equation $Ax = 0$ is *minimal* if there is no $y \in \mathbb{N}^m$ with $Ay = 0$ and $y \leq x$, $y \neq x$. The set of all solutions clearly forms a submonoid of $\mathbb{N}^m$, which is denoted M. The set of minimal solutions is denoted $\mathcal{H}(M)$ and called the *Hilbert basis* of M. Let r be the rank of A. Pottier showed the following.

▶ **Theorem 9** (Pottier [29])**.** *For each $x \in \mathcal{H}(M)$, $\|x\|_1 \leq (1 + \|A\|_{1,\infty})^r$.*

By applying Theorem 9 to the matrix $(A| - b)$, it is easy to deduce that for each $x \in \mathbb{N}^m$ with $Ax = b$, there is a $y \in \mathbb{N}^m$ with $Ay = b$, $y \leq x$, and $\|y\|_1 \leq (1 + \|(A| - b)\|_{1,\infty})^{r+1}$.

Automata for the downward closure. Let $\mathcal{A}$ be a blind k-counter automaton with n states. The idea of the construction of $\mathcal{B}$ is to traverse insertion trees of prime cycles of $\mathcal{A}$. Although insertion trees were introduced for finite automata, they also apply to blind counter automata if we regard the counter updates as input symbols. $\mathcal{B}$ keeps track of where it is in the tree using a stack of bounded height. The stack alphabet will be $\Gamma = Q \times [-n, n]^k$. We define $B = n + n \cdot (3n)^{(k+1)^2}$. The state set of our automaton $\mathcal{B}_1$ is the following:

$$Q_1 = Q \times \Gamma^{\leq n} \times [-B, B]^k \times \mathbb{P}([-n, n]^k) \times \mathbb{P}([-n, n]^k).$$

Here, the number of states is clearly doubly exponential, but we shall make the automaton smaller in two later steps. The idea behind $\mathcal{B}_1$ is that counter values in the interval $[-B, B]$ are simulated precisely (in the factor $[-B, B]^k$). Roughly speaking, whenever we encounter a cycle, we can decide whether to (i) add its effect to this precise counter or to (ii) remember the effect as "must be added at least once". We call the former *precise cycles*; the latter are dubbed *obligation cycles* and are stored in the first factor $\mathbb{P}([-n, n]^k)$. In either case, the effect of a cycle is kept as "repeatable" in the second factor $\mathbb{P}([-n, n]^k)$.

In order to be able to guess for each cycle whether it should be a precise cycle or an obligation cycle, we traverse an insertion tree of (the prime cycles on) a walk of $\mathcal{A}$. On the stack (the factor $\Gamma^{\leq n}$), we keep the cycles that we have started to traverse. Suppose we are executing a cycle in a vertex v and the path from the root to v consists of the vertices $v_1, \ldots, v_m$. Let $\gamma(v_i)$ be a q_i-cycle for $i \in [1, m]$. Then, the stack content is $(q_1, u_1) \cdots (q_m, u_m)$, where u_i is the effect of the part of $\gamma(v_i)$ that has already been traversed.

In the end, we verify that (i) the precise counter is zero and (ii) one can add up obligation cycles (each of them at least once) and repeatable cycles to zero. The latter condition is captured in the following notion. Let $S, T \subseteq \mathbb{Z}^k$ be finite sets with $S = \{u_1, \ldots, u_s\}$, $T = \{v_1, \ldots, v_t\}$. We call the pair (S, T) *cancellable* if there are $x_1, \ldots, x_s \in \mathbb{N} \setminus \{0\}$ and $y_1, \ldots, y_t \in \mathbb{N}$ with $\sum_{i=1}^{s} x_i u_i + \sum_{i=1}^{t} y_i v_i = 0$. In particular, $(\emptyset, T)$ is cancellable for any finite $T \subseteq \mathbb{Z}^k$. Together, (i) and (ii) guarantee that the accepted word is in the downward closure: They imply that we could have executed all of the obligation cycles and some others (again) to fulfill our obligation. Hence, there is a run of $\mathcal{A}$ accepting a superword.

The number of cycles we can use as precise cycles is limited by the capacity B of our precise counter. We shall apply Theorem 9 to show that there is always a choice of cycles to use as precise cycles so as to reach zero in the end and not exceed the capacity.

The first type of transition in $\mathcal{B}_1$ is the following. For each transition $(p, a, d, q) \in \Delta$ and state $(p, \varepsilon, v, S, T) \in Q_1$ such that $v + d \in [-B, B]^k$, we have a transition

$$(p, \varepsilon, v, S, T) \xrightarrow{a} (q, \varepsilon, v + d, S, T). \tag{1}$$

These allow us to simulate transitions in a walk of $\mathcal{A}$ that are not part of a cycle. We can guess that a cycle is starting. If we are in state p, then we push $(p, 0)$ onto the stack:

$$(p, w, v, S, T) \xrightarrow{\varepsilon} (p, w(p, 0), v, S, T). \tag{2}$$

While we are traversing a cycle, new counter effects are stored in the topmost stack entry. For each $(p, a, d, q) \in \Delta$ and $(p, w(r, u), v, S, T) \in Q_1$ with $u + d \in [-n, n]^k$, we have:

$$(p, w(r, u), v, S, T) \xrightarrow{a} (q, w(r, u + d), v, S, T). \tag{3}$$

When we are at the end of a cycle, we have to decide whether it should be a precise cycle or an obligation cycle. The following transition means it should be precise: The counter effect u of the cycle is added to the counter v, the stack is popped, and u is added to the set of repeatable effects T. For each $(p, w(p, u), v, S, T) \in Q_1$ with $v + u \in [-B, B]^k$, we have:

$$(p, w(p, u), v, S, T) \xrightarrow{\varepsilon} (p, w, v + u, S, T \cup \{u\}). \tag{4}$$

In order to designate the cycle as an obligation cycle, we have the following transition: The stack is popped and u is added to both S and T. For each state $(p, w(p, u), v, S, T) \in Q_1$, we include the transition

$$(p, w(p, u), v, S, T) \xrightarrow{\varepsilon} (p, w, v, S \cup \{u\}, T \cup \{u\}) \tag{5}$$

The initial state is $(q_0, \varepsilon, 0, \emptyset, \emptyset)$ and the final states are all those of the form $(q, \varepsilon, 0, S, T)$ where q is final in $\mathcal{A}$ and (S, T) is cancellable. Employing Lemma 6 and Theorem 9, one can now show that $L(\mathcal{A}) \subseteq L(\mathcal{B}_1) \subseteq L(\mathcal{A})\downarrow$.

State space reduction I. We have thus shown that $L(\mathcal{B}_1)\downarrow = L(\mathcal{A})\downarrow$. However, $\mathcal{B}_1$ has a doubly exponential number of states. Therefore, we now reduce the number of states in two steps. First, instead of remembering the set S of obligation effects, we only maintain a linearly independent set of vectors generating the same vector space. For a set $R \subseteq \mathbb{Q}^k$, let $\mathrm{span}(R)$ denote the $\mathbb{Q}$-vector space generated by R. Moreover, $\mathbb{I}(R)$ denotes the set of linearly independent subsets of R. Our new automaton $\mathcal{B}_2$ has states

$$Q_2 = Q \times \Gamma^{\leq n} \times [-B, B]^k \times \mathbb{I}([-n, n]^k) \times \mathbb{P}([-n, n]^k)$$

and a state in $\mathcal{B}_2$ is final if it is final in $\mathcal{B}_1$. $\mathcal{B}_2$ has the same transitions as $\mathcal{B}_1$, except that aside from those of type (5), it has

$$(p, w(p, u), v, S, T) \xrightarrow{\varepsilon} (p, w, v, S', T \cup \{u\}) \tag{6}$$

for each linearly independent subset $S' \subseteq S \cup \{u\}$ such that $\mathrm{span}(S') = \mathrm{span}(S \cup \{u\})$. Of course, such an S' exists for any S and u. This means, by induction on the length, for any walk of $\mathcal{B}_1$ from (p, w, v, S, T) to (q, w', v', S', T'), we can find a walk with the same input in $\mathcal{B}_2$ from (p, w, v, S, T) to (q, w', v', S'', T') with $S'' \subseteq S'$ and $\mathrm{span}(S'') = \mathrm{span}(S')$. Since (S', T') is cancellable and $S' \subseteq T'$, the pair (S'', T') is cancellable as well. This means, our walk in $\mathcal{B}_2$ is accepting and hence $L(\mathcal{B}_1) \subseteq L(\mathcal{B}_2)$. It remains to verify that $L(\mathcal{B}_2) \subseteq L(\mathcal{B}_1)$.

Observe that for any walk arriving in (q, w, v, S, T) in $\mathcal{B}_2$, there is a corresponding walk in $\mathcal{B}_1$ arriving in (q, w, v, S', T) for some $S' \supseteq S$ with $\mathrm{span}(S') = \mathrm{span}(S)$. The next lemma tells us that if (q, w, v, S, T) is a final state in $\mathcal{B}_2$, then (q, w, v, S', T) is final in $\mathcal{B}_1$. This implies that $L(\mathcal{B}_2) \subseteq L(\mathcal{B}_1)$ and hence $L(\mathcal{B}_2) = L(\mathcal{B}_1)$.

▶ **Lemma 10.** *Let $T \subseteq \mathbb{Z}^k$ and $S_1 \subseteq S_2 \subseteq \mathbb{Z}^k$ such that $\mathrm{span}(S_1) = \mathrm{span}(S_2)$. If (S_1, T) is cancellable, then so is (S_2, T).*

State space reduction II. We apply a similar transformation to the last factor of the state space. In $\mathcal{B}_3$, we have the state space

$$Q_3 = Q \times \Gamma^{\leq n} \times [-B, B]^k \times \mathbb{I}([-n, n]^k) \times \mathbb{I}([-n, n]^k).$$

and a state is final in $\mathcal{B}_3$ if and only if it is final in $\mathcal{B}_2$. Analogous to $\mathcal{B}_2$, we change the transitions so that instead of adding $u \in [-n, n]^k$ to T, we store an arbitrary $T' \in \mathbb{I}(T \cup \{u\})$.

This time, it is clear that $L(\mathcal{B}_3) \subseteq L(\mathcal{B}_2)$: For every walk in $\mathcal{B}_3$ arriving at (q, w, v, S, T), there is a corresponding walk in $\mathcal{B}_2$ arriving at (q, w, v, S, T') such that $T \subseteq T'$. Clearly, if (S, T) is cancellable, then (S, T') must be cancellable as well. The following lemma implies $L(\mathcal{B}_2) \subseteq L(\mathcal{B}_3)$: It says that for each walk in $\mathcal{B}_2$ arriving at (q, w, v, S, T), there is a corresponding walk in $\mathcal{B}_3$ arriving at (q, w, v, S, T') for some linearly independent $T' \subseteq T$ such that (S, T') is cancellable and hence (q, w, v, S, T') is final.

▶ **Lemma 11.** *Let $S, T \subseteq \mathbb{Z}^k$ such that (S, T) is cancellable. Then there is a linearly independent subset $T' \subseteq T$ such that (S, T') is cancellable.*

We have thus shown that $L(\mathcal{B}_3)\downarrow = L(\mathcal{A})\downarrow$. An estimation of the size of Q_3 now completes the proof of Theorem 8. We apply Theorem 8 to derive an algorithm for $\mathsf{Ideal} \subseteq_\downarrow \mathsf{RBC}$.

▶ **Corollary 12.** *The problem* Ideal $\subseteq_\downarrow$ RBC *is in* NP.

Since Theorem 8 provides an exponential bound on $|L(\mathcal{A})|_{\mathsf{I}}$, we can use an ideal witness $w = w_{Y_0}^m x_1 w_{Y_1}^m \cdots x_\ell w_{Y_\ell}^m$ (Proposition 2) for which we have to check membership in $L(\mathcal{A})$. Since ℓ is polynomial and m exponential, we can compute a compressed representation of w in form of a *straight-line program*, a context-free grammar that generates one word [25]. It follows easily from work of Hague and Lin [16] that membership of such compressed words in languages of blind (or reversal-bounded) counter automata is decidable in NP.

Fixed number of counters. Unfortunately, the size bound for the NFAs provided by Theorem 8 has the number of states in the exponent, meaning that if we fix the number k of counters, we still have an exponential bound. In fact, we leave open whether one can construct polynomial-size NFAs for fixed k. However, in many cases it suffices to have a polynomial bound on the length of ideals.

▶ **Theorem 13.** *If $\mathcal{A}$ is an n-state blind k-counter automaton, then $|L(\mathcal{A})|_{\mathsf{I}} \leq (5n)^{7(k+1)^2}$.*

Recall that an upper bound on $|L|_{\mathsf{I}}$ is essentially a pumping lemma (see Section 3). Here, the idea is to take a walk of $\mathcal{A}$ and delete cycles until the remaining walk u is at most n steps. For the deleted cycles, we take an insertion tree of height at most n (Lemma 6). Then, using Theorem 9, we pick a subset F (whose size is polynomial when fixing k) of cycles that can balance out the effect of u. We then employ Lemma 7 to the insertion trees to construct an ideal whose length is polynomial in $|F|$.

6 Context-Free Grammars

We turn to context-free grammars. First, we mention that given a context-free grammar $\mathcal{G}$, one can construct in exponential time an (exponential-size) NFA accepting $L(\mathcal{A})\downarrow$ [4, 7, 11, 33, 27]. Second, we provide an algorithm for the problem Ideal $\subseteq_\downarrow$ CFG.

▶ **Theorem 14.** *The problem* Ideal $\subseteq_\downarrow$ CFG *is in* P.

In [34], this problem has been reduced to the *simultaneous unboundedness problem (SUP)* for context-free languages. The latter asks, given a language $L \subseteq a_1^* \cdots a_n^*$, whether we have $L\downarrow = a_1^* \cdots a_n^*$. Moreover, this reduction is clearly polynomial. Hence, we assume that $L(\mathcal{G}) \subseteq a_1^* \cdots a_n^*$ and that the grammar $\mathcal{G} = (N, T, P, S)$ is productive and in Chomsky normal form, meaning that productions are of the form $A \to BC$, $A \to a_i$, or $A \to \varepsilon$ for $A, B, C \in N$. First, we add productions $A \to \varepsilon$ for all $A \in N$, so that the resulting grammar $\mathcal{G}'$ satisfies $L(\mathcal{G}') = L(\mathcal{G})\downarrow$. For each $A \in N$, we can in polynomial time construct a CFG for $\{w \in (N \cup T)^* \mid A \Rightarrow_{\mathcal{G}'}^* w\}$, so we can compute the sets $L_i = \{A \in N \mid A \Rightarrow_{\mathcal{G}'}^* a_i A\}$ and $R_i = \{A \in N \mid A \Rightarrow_{\mathcal{G}'}^* A a_i\}$ using membership queries. We can thus compute the grammar $\mathcal{G}^\omega$, which results from $\mathcal{G}'$ by (i) removing all productions $A \to a_i$, (ii) adding $A \to a_i^\omega A$ for each $A \in L_i$ and (iii) adding $A \to A a_i^\omega$ for each $A \in R_i$. Clearly, an occurrence of a_i^ω certifies the ability to generate an unbounded number of a_i's. Thus, if $a_1^\omega \cdots a_n^\omega \in L(\mathcal{G}^\omega)$, then $a_1^* \cdots a_n^* \subseteq L(\mathcal{G}') = L(\mathcal{G})\downarrow$. It is not hard to see that the converse is true as well. We have thus reduced the SUP to the membership problem.

7 Algorithms

Algorithms for $\mathcal{M} \subseteq_\downarrow$ Ideal. Suppose $\mathcal{M} =$ Ideal and we want to decide whether $I \subseteq J$ for ideals $I, J \subseteq X^*$. In logspace, we construct an ideal witness w for I and J (Proposition 2)

and a DFA $\mathcal{A}$ for $X^* \setminus J$ (Proposition 4) and check whether $w \in L(\mathcal{A})$. In all other cases, to decide $L{\downarrow} \subseteq I$, we construct a DFA $\mathcal{A}$ for $X^* \setminus I$ and check whether $L{\downarrow} \cap L(\mathcal{A}) = \emptyset$.

Algorithms for $\mathcal{M} \subseteq_{\downarrow}$ NFA. Suppose $\mathcal{M} = \mathsf{Ideal}$ and we want to decide whether $I \subseteq L(\mathcal{A}){\downarrow}$ for an NFA $\mathcal{A}$. Since $|L(\mathcal{A})|_{\mathsf{I}} \leq |\mathcal{A}|$, we can construct in logspace an ideal witness w for I and $L(\mathcal{A}){\downarrow}$ and verify $w \in L(\mathcal{A}){\downarrow}$. In all other cases, we use a short witness for coNP-membership.

Algorithms for $\mathcal{M} \subseteq_{\downarrow}$ OCA. Suppose $\mathcal{M} = \mathsf{Ideal}$ and we want to decide whether $I \subseteq L(\mathcal{A}){\downarrow}$ for an OCA $\mathcal{A}$. We have a polynomial bound on $|L(\mathcal{A})|_{\mathsf{I}}$ (see Section 2). Hence, we construct in logspace an ideal witness w for I and $L(\mathcal{A}){\downarrow}$. We can also construct in logspace an OCA $\mathcal{A}'$ with $L(\mathcal{A}') = L(\mathcal{A}){\downarrow}$. Membership for OCA is in $\mathsf{NL} = \mathsf{coNL}$, so we can verify $w \in I$ and $w \notin L(\mathcal{A}') = L(\mathcal{A}){\downarrow}$. In all other cases, we convert the OCA to an NFA (see Section 2).

Algorithms for $\mathcal{M} \subseteq_{\downarrow} \mathsf{RBC}_{k,r}$. Let $\mathcal{A}$ be drawn from $\mathsf{RBC}_{k,r}$. First, suppose $\mathcal{M} = \mathsf{Ideal}$ and we want to decide whether $I \subseteq L(\mathcal{A})$. By Theorem 13, we have a polynomial bound on $|L(\mathcal{A})|_{\mathsf{I}}$ and can construct in logspace an ideal witness w for I and $L(\mathcal{A})$. We can also construct in logspace an RBCA $\mathcal{A}'$ with $L(\mathcal{A}') = L(\mathcal{A}){\downarrow}$. Since membership for $\mathsf{RBC}_{k,r}$ is in NL [12], we can check whether $w \in L(\mathcal{A}')$. Now let $\mathcal{M} \in \{\mathsf{NFA}, \mathsf{OCA}, \mathsf{RBC}_{k,r}\}$ and we are given L in $\mathcal{M}$ and an automaton $\mathcal{A}$ from $\mathsf{RBC}_{k,r}$. For NFA, OCA, and $\mathsf{RBC}_{k,r}$, we have a polynomial bound on $|L|_{\mathsf{I}}$ (see Section 2 and Theorem 13). Thus, we guess an ideal I of polynomial length and then verify that $I \subseteq L{\downarrow}$ but $I \not\subseteq L(\mathcal{A}){\downarrow}$. Since $\mathsf{Ideal} \subseteq_{\downarrow} \mathcal{M}$ and $\mathsf{Ideal} \subseteq_{\downarrow} \mathsf{RBC}_{k,r}$ are in NL, the verification is done in NL. Hence, non-inclusion is in NP. For $\mathcal{M} \in \{\mathsf{CFG}, \mathsf{RBC}\}$, we assume a fixed alphabet. Let L be in $\mathcal{M}$. Then Proposition 3 and Theorem 13 provide us with a witness of polynomial length. Since (non-)membership in $L{\downarrow}$ and in $L(\mathcal{A}){\downarrow}$ can be decided in NP, non-inclusion is in NP.

Algorithms for $\mathcal{M} \subseteq_{\downarrow}$ CFG. The case $\mathsf{Ideal} \subseteq_{\downarrow} \mathsf{CFG}$ is shown in Theorem 14. Suppose $\mathcal{M} \in \{\mathsf{NFA}, \mathsf{OCA}, \mathsf{RBC}_{k,r}\}$ and we are given L in $\mathcal{M}$ and a CFG $\mathcal{G}$. We have a polynomial bound on $|L|_{\mathsf{I}}$ (see Section 2 and Theorem 13), so that we can guess a polynomial-length ideal I. Since $\mathsf{Ideal} \subseteq_{\downarrow} \mathcal{M}$ is in NL in every case and $\mathsf{Ideal} \subseteq_{\downarrow} \mathsf{CFG}$ is in P, we can verify in polynomial time that $I \subseteq L{\downarrow}$ and $I \not\subseteq L(\mathcal{G}){\downarrow}$. Thus, non-inclusion is in NP. In the case $\mathcal{M} \in \{\mathsf{RBC}, \mathsf{CFG}\}$, we construct exponential-size downward closure NFAs and check inclusion for them (and the latter problem is in coNP). This yields a coNEXP algorithm.

Algorithms for $\mathcal{M} \subseteq_{\downarrow}$ RBC. Let $\mathcal{A}$ be from RBC. The ideal case is treated in Corollary 12. When given L in $\mathcal{M} \in \{\mathsf{NFA}, \mathsf{OCA}, \mathsf{RBC}_{k,r}\}$, we guess a polynomial length ideal I and verify that $I \subseteq L{\downarrow}$ in NL. Since $\mathsf{Ideal} \subseteq_{\downarrow} \mathsf{RBC}$ is in NP, we can also check in coNP that $I \not\subseteq L(\mathcal{A}){\downarrow}$. Hence, non-inclusion is in Σ_2^{P}. For $\mathcal{M} \in \{\mathsf{CFG}, \mathsf{RBC}\}$, we proceed as for $\mathcal{M} \subseteq_{\downarrow} \mathsf{CFG}$.

8 Hardness

In this section, we prove hardness results. Most of them are deduced from a generic hardness theorem that, under mild assumptions, derives hardness from the ability to generate finite sets with long words. We will work with bounds that exhibit the following useful property. A monotone function $f\colon \mathbb{N} \to \mathbb{N}$ will be called *amplifying* if $f(n) \geq n$ for $n \geq 0$ and there is a polynomial p such that $f(p(n)) \geq f(n)^2$ for large enough $n \in \mathbb{N}$. We say that a model *has property* $\Delta(f)$ (or short: *is* $\Delta(f)$) if for each given $n \in \mathbb{N}$, one can construct in polynomial time a description of a finite language whose longest word has length $f(n)$. For the sake of simplicity, we will abuse notation slightly and write $\Delta(f(n))$ instead of $\Delta(f)$. For a function $t\colon \mathbb{N} \to \mathbb{N}$, we use $\mathsf{coNTIME}(t)$ to denote the complements of languages accepted by nondeterministic Turing machines that are time bounded by $O(t(n^c))$ for some constant c.

We also need two mild language theoretic properties. A *transducer* is a finite automaton where every edge reads input and produces output. For a transducer $\mathcal{T}$ and a language L, the language $\mathcal{T}L$ consists of all words output by the transducer while reading a word from L. We call a model $\mathcal{M}$ a *full trio model* if given a transducer $\mathcal{T}$ and a language L described with $\mathcal{M}$, one can compute in polynomial time a description of $\mathcal{T}L$. A *substitution* is a map $\sigma\colon X \to \mathbb{P}(Y^*)$ that replaces each letter by a language. For languages L, we define $\sigma(L)$ in the obvious way. We call σ *simple* if $X \subseteq Y$ and there is some $x \in X$ such that for all $x' \in X \setminus \{x\}$, we have $\sigma(x') = \{x'\}$ and x occurs in each word from L at most once. We say that $\mathcal{M}$ has *closure under simple substitutions* if given a description of L and of $\sigma(x)$ in $\mathcal{M}$, we can compute in polynomial time a description of $\sigma(L)$.

▶ **Theorem 15.** *Let $t\colon \mathbb{N} \to \mathbb{N}$ be amplifying and let $\mathcal{M}$ and $\mathcal{N}$ be full trio models that are $\Delta(t)$ and have closure under simple substitutions. Then both $\mathcal{M} \subseteq_\downarrow \mathcal{N}$ and $\mathcal{M} =_\downarrow \mathcal{N}$ are hard for* coNTIME(t)*. Moreover, this hardness already holds for binary alphabets.*

Since NFAs are $\Delta(n)$, Theorem 15 yields coNP-hardness for inclusion and equivalence. In [4], hardness of equivalence was shown directly. RBCA and CFG clearly exhibit closure under simple substitutions and can generate exponentially long words. This yields:

▶ **Corollary 16.** *For $\mathcal{M}, \mathcal{N} \in \{CFG, RBC\}$, $\mathcal{M} \subseteq_\downarrow \mathcal{N}$ and $\mathcal{M} =_\downarrow \mathcal{N}$ are* coNEXP*-hard.*

From Theorem 15, we can also deduce hardness for other models. It was shown by Habermehl, Meyer, and Wimmel [14] that downward closures or Petri net languages are computable, which implies decidability of our problems. We use Theorem 15 to prove an Ackermann lower bound. Let $A_n\colon \mathbb{N} \to \mathbb{N}$ be defined as $A_0(x) = x + 1$, $A_{n+1}(0) = A_n(1)$, and $A_{n+1}(x+1) = A_n(A_{n+1}(x))$. Then, the function $A\colon \mathbb{N} \to \mathbb{N}$ with $A(n) = A_n(n)$ is the *Ackermann function*. Of course, for large enough n, we have $A_n(x) \geq x^2$. For such n, we have $A(n+1) = A_n(A_{n+1}(n)) \geq A_{n+1}(n)^2 \geq A(n)^2$, so A is amplifying. A result of Mayr and Meyer [26] (see also [30]) states that given $n \in \mathbb{N}$, one can construct in polynomial time a Petri net that, from its initial marking, can produce up to $A(n)$ tokens in an output place. Hence, Petri nets are $\Delta(A)$ and they clearly satisfy the language-theoretic conditions.

▶ **Corollary 17.** *For Petri net languages, inclusion and equivalence of downward closures is Ackermann-hard.*

Building on the sufficient condition of [34], Hague, Kochems, and Ong [15] have shown that downward closures are computable for higher-order pushdown automata. However, the method of [34] does not yield any information about the complexity of this computation. For $k \in \mathbb{N}$, we denote by $\exp_k$ the function with $\exp_0(n) = n$ and $\exp_{k+1}(n) = 2^{\exp_k(n)}$. It is easy to see that order-k pushdown automata are $\Delta(\exp_k)$ (for instance, one can adapt Example 2.5 of [8]). By co-k-NEXP, we denote the complements of languages accepted by nondeterministic Turing machines in time $O(\exp_k(n^c))$ for some constant c.

▶ **Corollary 18.** *For higher-order pushdown automata of order k, inclusion and equivalence of downward closures is hard for* co-k-NEXP*.*

Our last hardness result could also be shown using the method of Theorem 15. However, it is simpler to reduce a variant of the subset sum problem [6].

▶ **Proposition 19.** NFA $\subseteq_\downarrow$ RBC *and* NFA $=_\downarrow$ RBC *are Π_2^{P}-hard, even for binary alphabets.*

We have thus shown hardness for all inclusion problems that do not involve ideals. The remaining cases inherit hardness from the emptiness problem (for $\mathcal{M} \subseteq_\downarrow$ Ideal) or the non-emptiness problem (Ideal $\subseteq_\downarrow \mathcal{M}$).

References

1. P. A. Abdulla, L. Boasson, and A. Bouajjani. Effective lossy queue languages. In *ICALP 2001*, 2001.
2. P. A. Abdulla, A. Collomb-Annichini, A. Bouajjani, and B. Jonsson. Using forward reachability analysis for verification of lossy channel systems. *Formal Methods in System Design*, 25(1):39–65, 2004.
3. M. F. Atig, D. Chistikov, P. Hofman, K. N. Kumar, P. Saivasan, and G. Zetzsche. The complexity of regular abstractions of one-counter languages. To appear in LICS 2016.
4. G. Bachmeier, M. Luttenberger, and M. Schlund. Finite automata for the sub- and superword closure of cfls: Descriptional and computational complexity. In *LATA 2015*, 2015.
5. B. S. Baker and R. V. Book. Reversal-bounded multipushdown machines. *Journal of Computer and System Sciences*, 8(3):315–332, 1974.
6. P. Berman, M. Karpinski, L. L. Larmore, W. Plandowski, and W. Rytter. On the complexity of pattern matching for highly compressed two-dimensional texts. In *CPM 1997*, 1997.
7. B. Courcelle. On constructing obstruction sets of words. *Bulletin of the EATCS*, 44:178–186, 1991.
8. W. Damm and A. Goerdt. An automata-theoretic characterization of the OI-hierarchy. In *ICALP 1982*, 1982.
9. S. A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7(3):311–324, 1978.
10. H. Gruber, M. Holzer, and M. Kutrib. The size of Higman-Haines sets. *Theoretical Computer Science*, 387(2):167–176, 2007.
11. H. Gruber, M. Holzer, and M. Kutrib. More on the size of higman-haines sets: effective constructions. *Fundamenta Informaticae*, 91(1):105–121, 2009.
12. E. M. Gurari and O. H. Ibarra. The complexity of decision problems for finite-turn multicounter machines. *Journal of Computer and System Sciences*, 22(2):220–229, 1981.
13. C. Haase and P. Hofman. Tightening the complexity of equivalence problems for commutative grammars. In *STACS 2016*, 2016.
14. P. Habermehl, R. Meyer, and H. Wimmel. The downward-closure of Petri net languages. In *ICALP 2010*, 2010.
15. M. Hague, J. Kochems, and C.-H. L. Ong. Unboundedness and downward closures of higher-order pushdown automata. In *POPL 2016*, 2016.
16. M. Hague and A. W. Lin. Model checking recursive programs with numeric data types. In *CAV 2011*, 2011.
17. L. H. Haines. On free monoids partially ordered by embedding. *Journal of Combinatorial Theory*, 6(1):94–98, 1969.
18. D. T. Huynh. The complexity of equivalence problems for commutative grammars. *Information and Control*, 66(1):103–121, 1985.
19. O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM (JACM)*, 25(1):116–133, 1978.
20. M. Jantzen and A. Kurganskyy. Refining the hierarchy of blind multicounter languages and twist-closed trios. *Information and Computation*, 185(2):159–181, 2003.
21. P. Jullien. *Contribution à létude des types d'ordres dispersés*. PhD thesis, Université de Marseille, 1969.
22. P. Karandikar, M. Niewerth, and Ph. Schnoebelen. On the state complexity of closures and interiors of regular languages with subwords and superwords. *Theoretical Computer Science*, 610, Part A:91–107, 2016.
23. E. Kopczyński. Complexity of problems of commutative grammars. *Logical Methods in Computer Science*, 11(1), 2015.

24 E. Kopczyński and A. W. To. Parikh images of grammars: Complexity and applications. In *LICS 2010*, 2010.

25 M. Lohrey. Algorithmics on SLP-compressed strings: a survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.

26 E. W. Mayr and A. R. Meyer. The complexity of the finite containment problem for petri nets. *Journal of the ACM*, 28(3):561–576, 1981.

27 A. Okhotin. On the state complexity of scattered substrings and superstrings. *Fundamenta Informaticae*, 99(3):325–338, 2010.

28 R. J. Parikh. On context-free languages. *Journal of the ACM*, 13(4):570–581, 1966.

29 L. Pottier. Minimal solutions of linear diophantine systems : bounds and algorithms. In *RTA 1991*, 1991.

30 L. Priese and H. Wimmel. *Petri-Netze*. Springer-Verlag, 2003.

31 N. Rampersad, J. Shallit, and Z. Xu. The computational complexity of universality problems for prefixes, suffixes, factors, and subwords of regular languages. *Fundamenta Informaticae*, 116(1-4):223–236, 2012.

32 S. La Torre, A. Muscholl, and I. Walukiewicz. Safety of Parametrized Asynchronous Shared-Memory Systems is Almost Always Decidable. In *CONCUR 2015*, 2015.

33 J. van Leeuwen. Effective constructions in well-partially-ordered free monoids. *Discrete Mathematics*, 21(3):237–252, 1978.

34 G. Zetzsche. An approach to computing downward closures. In *ICALP 2015*, 2015.

35 G. Zetzsche. Computing downward closures for stacked counter automata. In *STACS 2015*, 2015.

36 G. Zetzsche. The complexity of downward closure comparisons, 2016. URL: `http://arxiv.org/abs/1605.03149`.

Anti-Powers in Infinite Words

Gabriele Fici[1], Antonio Restivo[2], Manuel Silva[3], and Luca Q. Zamboni[4]

1 **Dipartimento di Matematica e Informatica, Università di Palermo, Palermo, Italy**
gabriele.fici@unipa.it
2 **Dipartimento di Matematica e Informatica, Università di Palermo, Palermo, Italy**
antonio.restivo@unipa.it
3 **Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Lisbon, Portugal**
mnasilva@gmail.com
4 **Institut Camille Jordan, Université Claude Bernard Lyon 1, Lyon, France**
zamboni@math.univ-lyon1.fr

Abstract

In combinatorics of words, a concatenation of k consecutive equal blocks is called a power of order k. In this paper we take a different point of view and define an anti-power of order k as a concatenation of k consecutive pairwise distinct blocks of the same length. As a main result, we show that every infinite word contains powers of any order or anti-powers of any order. That is, the existence of powers or anti-powers is an unavoidable regularity. Indeed, we prove a stronger result, which relates the density of anti-powers to the existence of a factor that occurs with arbitrary exponent. From these results, we derive that at every position of an aperiodic uniformly recurrent word start anti-powers of any order. We further show that any infinite word avoiding anti-powers of order 3 is ultimately periodic, and that there exist aperiodic words avoiding anti-powers of order 4. We also show that there exist aperiodic recurrent words avoiding anti-powers of order 6, and leave open the question whether there exist aperiodic recurrent words avoiding anti-powers of order k for $k = 4, 5$.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases infinite word, anti-power, unavoidable regularity, avoidability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.124

1 Introduction

Suppose you are in a room with a hundred people and somebody tells you that by an incredible coincidence the people in the room have *all different* birthdays. Of course this is much less surprising than if all the people had *the same* birthday; still you remember from your first course in combinatorics (or even before) that already in a class of fifty people the probability that no two people have the same birthday is less than 3%. So actually you are in a very *special* situation![1]

For a number of instances of objects taken from a fixed class using some rule, being all distinct can be viewed as a kind of regularity. This has been already considered extensively

[1] To be more precise, the probability that at least two people have the same birthday in a room of 100 people is about 0.9999996928.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 124; pp. 124:1–124:9

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

in computer science. For example, the probability of collisions must be properly quantified when designing hashing functions. In the context of string processing, one usually deals with factors (substrings) of a word, and it is sometimes useful to factorize a word in blocks that are all distinct — a widely known example is the Lempel-Ziv factorization, at the basis of the eponymous compression algorithm. The problem of factoring a string in blocks that are all distinct (sometimes called equality-free factorization [7]) has practical applications also in bio-informatics, since it appears to be connected with gene synthesis [2]. Equality-free factorizations have been further considered by Fernau et al. [4], motivated by injective pattern matching with variables, which is identical to the special case of solving word equations where the left side of the equation does not contain variables, and different variables must be replaced by different words. In particular, in [4] it is proved that given a finite word w and a number k, the problem whether it is possible to factorize w into at least k distinct factors is $\mathcal{NP}$-complete.

In the context of infinite words, the complexity of sequences is often described by means of parameters that capture some kind of repetitiveness. To this end, one often considers as a degree of repetitiveness the maximal number of consecutive all-equal blocks occurring in the sequence, regardless of the length of the single blocks. A concatenation of k consecutive equal blocks is called *a power of order k*, or simply *a k-power*. E.g., *aabaabaabaab* is a 4-power. A first classification of infinite words consists in identifying those that are *k-power-free* for some $k \geq 2$, meaning that they do not contain any factor that is a k-power. Words avoiding k-powers have been the object of study of combinatorics on words since the very beginning of the theory [8] (cf. also [5]).

In this paper we adopt a different point of view based on the difference rather than on the equality. We consider the problem of finding in infinite words consecutive blocks of the same length that are all distinct. Of course, in the context of infinite words it is the requirement that the blocks all have the same length that makes the problem non-trivial, since otherwise one can always take arbitrarily long concatenations of blocks of increasing length to guarantee that they are all distinct.

We define an *anti-power of order k*, or simply a *k-anti-power*, as a concatenation of k consecutive pairwise distinct blocks of the same length. E.g., *aabaaabbbaba* is a 4-anti-power. A simple computation shows that there are in general much more anti-powers than powers for a fixed length and a fixed order; yet there are much less anti-powers than possible factors of the same given length.

Let us focus on an example. The Thue-Morse word

$$t = 01101001100101101001011001101001100101100110 \cdots$$

is perhaps the most prominent example in combinatorics of words [1]. It is defined as the word whose n-th digit is the parity of the number of 1s in the binary expansion of $n-1$ (so the first digit is the parity of 1s in 0, the second digit is the parity of 1s in 1, the third digit is the parity of 1s in 10, etc.). The Thue-Morse word does not contain *overlaps*, i.e., factors of the form $awawa$ for a letter a and a word w. In particular, the Thue-Morse word does not contain 3-powers (note that, on the other hand, every infinite binary word must contain 2-powers).

The shortest prefix of the Thue-Morse word that is a 2-antipower is 01. The shortest prefix that is a 3-anti-power is $01101 \cdot 00110 \cdot 01011$, of length 15. One can verify that the shortest 4-anti-power prefix has length 20. The first few lengths of the shortest prefixes of t that are k-anti-powers for different values of k are displayed in Table 1. A natural question is therefore the following: Given an integer $k > 1$, is it always possible to find a prefix of t

Table 1 The first few values of the sequence of lengths of the shortest prefixes of the Thue-Morse word that are k-anti-powers.

k	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	30	50	100
length	15	20	25	30	77	88	99	110	121	132	143	154	195	208	221	234	247	260	870	2450	9700
ratio	5	5	5	5	11	11	11	11	11	11	11	11	13	13	13	13	13	13	29	49	97

that is a k-anti-power? In this paper we answer this question in the affirmative. Actually, we prove a much stronger result. Indeed, we prove that the existence of powers of any order or anti-powers of any order is an unavoidable regularity:

▶ **Theorem 1.** *Every infinite word contains powers of any order or anti-powers of any order.*

Formally, an *unavoidable regularity* is a property P such that it is not possible to construct arbitrarily long words not satisfying P (cf. [3]). Some important theorems in combinatorics on words concern unavoidable regularities. Most of them follow from results originally stated in other areas of combinatorics, e.g., the Ramsey's, van der Waerden's and Shirshov's theorems (see [3, 5, 6] for further details).

Actually, we prove a stronger result, from which Theorem 1 follows. Given an infinite word x, we prove that if for some integer k the lower density of the set of lengths n for which the prefix of x of length kn is a k-anti-power is smaller than one, then there exists a word (whose length depends on k) that occurs in x with arbitrary exponent (Theorem 4). This implies that if an infinite word x has the property that each of its factors appears with bounded exponent (in the terminology of combinatorics on words we say that x is *ω-power-free*), then in x must start anti-powers of any order at every position. In particular, since a uniformly recurrent word is either purely periodic or ω-power-free, this property holds for every aperiodic uniformly recurrent word, as for example the Thue-Morse word or any Sturmian word[2].

In the second part of the paper, we focus on the avoidability of anti-powers. We show that any infinite word avoiding 3-anti-powers is ultimately periodic, and that there exist aperiodic words avoiding 4-anti-powers. We also show that there exist aperiodic *recurrent* words avoiding 6-anti-powers. We leave it as an open question to determine whether there exist aperiodic recurrent words avoiding 4-anti-powers or 5-anti-powers.

We conclude with final considerations and discuss open problems and further possible directions of investigation.

2 Preliminaries

Let $\mathbb{N} = \{1, 2, 3, \ldots\}$. Let $\mathbb{A}$ be a (possibly infinite) non-empty set, called the *alphabet*, whose elements are called *letters*. A *word* over $\mathbb{A}$ is a finite or infinite sequence of letters from $\mathbb{A}$. The *length* $|u|$ of a finite word u is the number of its letters. We let $\mathbb{A}^+$ denote the set of all finite words of positive length over $\mathbb{A}$, and $\mathbb{A}^{\mathbb{N}}$ the set of all infinite words over $\mathbb{A}$, that is, the

[2] Sturmian words are aperiodic words of minimal factor complexity. They are very well studied objects in combinatorics on words (see for instance [6]).

set of all maps from $\mathbb{N}$ to $\mathbb{A}$. Given a finite word u, we write u^ω the infinite word $uuu\cdots$ obtained by concatenating an infinite number of copies of u.

Given a finite or infinite word x, we say that a word u is a *factor* of x if $x = vuy$ for some words v and y. We say that u is a *prefix* (resp. *suffix*) of x if $x = uy$ (resp. $x = yu$) for some word y. We say that a word $u \neq x$ is a *border* of x if u is both a prefix and a suffix of x.

An infinite word x is *purely periodic* if there exists a positive integer p such that the letters occurring at positions i and j coincide whenever $i = j \mod p$. Equivalently, x is purely periodic if and only if $x = u^\omega$ for some word u of length p. An infinite word x is *ultimately periodic* if $x = uy$ for a finite word u and a purely periodic word y. An infinite word is *aperiodic* if it is not ultimately periodic.

An infinite word x is said to be *recurrent* if every finite factor of x occurs in x infinitely often. Equivalently, x is recurrent if and only if every finite prefix of x has a second occurrence as a factor. An infinite word x is said to be *uniformly recurrent* if every finite factor of x occurs syndetically (that is, it occurs infinitely often and with bounded gaps). Equivalently, x is uniformly recurrent if and only if for every finite factor u of x there exits an integer m such that u occurs in every factor of x of length m.

An infinite word x is said to be *k-power-free* for some integer $k > 1$ if for every finite factor u of x, one has that u^k is not a factor of x. An infinite word x is said to be *ω-power-free* if for every finite factor u of x there exists a positive integer l such that u^l is not a factor of x. Of course, if a word is k-power-free for some integer k, then it is ω-power-free, but the converse is not always true.

An important relationship between uniformly recurrent and ω-power-free words is the following (see for instance [3]):

▶ **Theorem 2.** *Every uniformly recurrent word is either purely periodic or ω-power-free.*

3 Unavoidability of powers or anti-powers

In order to state our main result, we need to introduce some definitions.

Let x be an infinite word and $k \in \mathbb{N}$. We set

$$P(x,k) = \{m \in \mathbb{N} \mid \text{ the prefix of } x \text{ of length } km \text{ is a } k\text{-power}\}.$$

Analogously, we set

$$AP(x,k) = \{m \in \mathbb{N} \mid \text{ the prefix of } x \text{ of length } km \text{ is a } k\text{-anti-power}\}.$$

Note that $P(x,1) = AP(x,1) = \mathbb{N}$ and that $P(x,k) \cap AP(x,k) = \emptyset$ for every $k \geq 2$. For example, if $x = 01^\omega$, we have $P(x,k) = AP(x,k) = \emptyset$ for every $k \geq 3$.

Recall that for any subset $X \subseteq \mathbb{N}$, the *lower density* of X is defined by

$$\underline{d}(X) = \liminf_{n\to\infty} \frac{|X \cap \{1,2,\ldots,n\}|}{n}.$$

Note that if X is finite, then $\underline{d}(X) = 0$. Moreover, if $\underline{d}(X) =< 1/t$ for some integer $t > 0$, then that there exist infinitely many integers m such that $\{m, m+1, \ldots, m+t-1\} \subset \mathbb{N} \setminus X$.

We are now going to prove our main result (Theorem 4). We premise a technical lemma.

▶ **Lemma 3.** *Let v be a border of a word w and u the word such that $w = uv$. If l is an integer such that $|w| > l|u|$, then u^l is a prefix of w.*

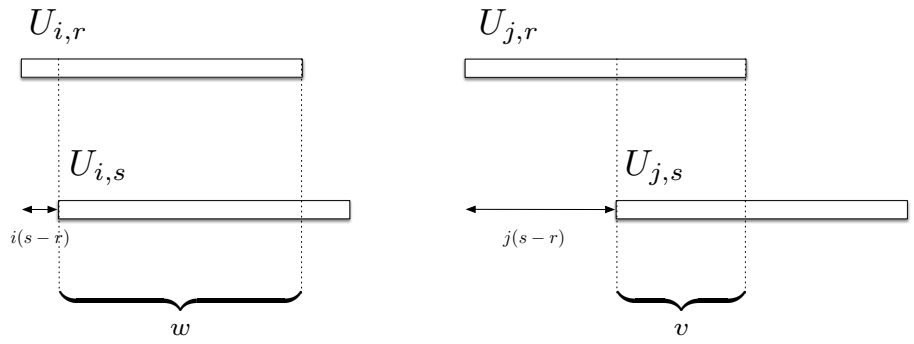


Figure 1 The proof of Theorem 4.

Proof. By induction on l. For $l = 1$ the statement trivially holds. Suppose $l > 1$. Since u is shorter than v and both are prefixes of w, we have that u is a prefix of v. Let us write $v = uv'$. Then $w = uuv'$ and v' is a border of v. Since $|v| = |w| - |u| > (l-1)|u|$, we can apply the induction hypothesis and derive that u^{l-1} is a prefix of v, whence u^l is a prefix of w. ◀

▶ **Theorem 4.** *Let x be an infinite word. Suppose that*

$$\underline{d}(AP(x,k)) < \left(1 + \binom{k}{2}\right)^{-1} = \frac{2}{2 + k(k-1)}$$

for some $k \in \mathbb{N}$. Then there exists $u \in \mathbb{A}^+$ with $|u| \leq (k-1)\binom{k}{2}$ such that u^l is a factor of x for every $l \geq 1$.

Proof. Fix k such that $\underline{d}(AP(x,k)) < (1+\binom{k}{2})^{-1}$. Since $AP(x,1) = \mathbb{N}$, and the lower density of $\mathbb{N}$ is 1, we have $k \geq 2$. We set $M = (k-1)\binom{k}{2}$. We have to show there exists $u \in \mathbb{A}^+$ with $|u| \leq M$ such that u^l is a factor of x for every $l \geq 1$. By the pigeonhole principle, it suffices to show that for every $l \in \mathbb{N}$ there exists $u \in \mathbb{A}^+$ with $|u| \leq M$ such that u^l is a factor of x.

So, let us fix $l \in \mathbb{N}$, and set $N = (l+1)M$. Since $\underline{d}(AP(x,k)) < (1 + \binom{k}{2})^{-1}$, there exists an integer $m > N$ such that $\{m, m+1, \ldots, m + \binom{k}{2}\} \subset \mathbb{N} \setminus AP(x,k)$.

For every j and r such that $0 \leq j \leq k-1$ and $m \leq r \leq m + \binom{k}{2}$, set

$$U_{j,r} = x_{jr+1}x_{jr+2}\cdots x_{(j+1)r},$$

so that $|U_{j,r}| = r$ and $U_{0,r}, U_{1,r}, \ldots, U_{k-1,r}$ are the first k consecutive blocks of x of length r. Thus for each $m \leq r \leq m + \binom{k}{2}$ there exist i and j, with $0 \leq i < j \leq k-1$, such that $U_{i,r} = U_{j,r}$. By the pigeonhole principle, there exist r and s, with $m \leq r < s \leq m + \binom{k}{2}$, and i and j, with $0 \leq i < j \leq k-1$, such that $U_{i,r} = U_{j,r}$ and $U_{i,s} = U_{j,s}$.

Notice that $(i+1)r > is + 1$ and $(j+1)r > js + 1$.

Let us now set $w = x_{is+1}x_{is+2}\cdots x_{(i+1)r}$ and $v = x_{js+1}x_{js+2}\cdots x_{(j+1)r}$ (see Figure 1). We have

$$|v| = (j+1)r - js < (i+1)r - is = |w|,$$

whence v is a border of w. Writing $w = uv$, we have

$$1 \leq |u| = |w| - |v| = (j-i)(s-r) \leq (k-1)\binom{k}{2} = M,$$

and

$$|w| > |v| = r - j(s-r) \geq m - (k-1)\binom{k}{2} = m - M > N - M = lM.$$

Thus, $|w| > l|u|$ and, by Lemma 3, u^l is a prefix of w, and therefore u^l is a factor of x. ◀

A more precise consequence of Theorem 4 is the following.

▶ **Corollary 5.** *Let x be a uniformly recurrent word. Then either*

$$\underline{d}(AP(x,k)) \geq \frac{2}{2+k(k-1)} \tag{1}$$

for every $k \in \mathbb{N}$ or there exists $r > 0$, such that

$$\underline{d}(P(x,k)) \geq r \tag{2}$$

for every $k \in \mathbb{N}$.

Proof. According to Theorem 4, if (1) does not hold for some $k' \in \mathbb{N}$, then $x = u^\omega$ for some u with $1 \leq |u| \leq (k'-1)\binom{k'}{2}$. Whence $n|u| \in P(x,k)$ for each $n, k \in \mathbb{N}$. The result now follows by setting $r = 1/|u|$. ◀

Note that the $\underline{d}(P(x,k)) \geq r$ for every k is stronger than just $\underline{d}(P(x,k)) > 0$. Conversely, if $\underline{d}(P(x,k)) > 0$ for some $k \geq 2$, then $\underline{d}(P(x,2)) > 0$, and from this it is immediate to see that x is periodic. In fact, something stronger is true: following the notation in the proof of Theorem 4, if there exists $j \geq 1$ such that $\underline{d}\{r \mid U_{0,r} = U_{j,r}\} > 0$, then x is periodic. And $\underline{d}(P(x,2)) > 0$ is a special case of this assumption (when $j = 1$).

▶ **Corollary 6.** *Let x be a uniformly recurrent word. If*

$$\underline{d}(AP(x,k)) < \frac{2}{2+k(k-1)}$$

for some $k \in \mathbb{N}$, then x is purely periodic.

Another direct consequence of Theorem 4 is the following.

▶ **Theorem 7.** *Let x be an infinite word. If x is ω-power-free, then at every position of x start anti-powers of any order.*

Proof. Suppose that there exists a positive integer k and a suffix x' of x such that no prefix of x' is a k-anti-power. Then $AP(x',k) = \emptyset$, whence $\underline{d}(AP(x',k)) = 0$. By Theorem 4, there exists a factor u of x' such that u^l is a factor of x' for every $l \geq 1$, hence x is not ω-power-free. ◀

From Theorems 2 and 7, we derive the following corollary.

▶ **Corollary 8.** *Let x be a uniformly recurrent aperiodic word. Then at every position of x start anti-powers of any order.*

4 Avoiding anti-powers

In this section we deal with avoidability of anti-powers.

▶ **Definition 9.** Given $k > 1$, we say that an infinite word x avoids k-anti-powers if no factor of x is a k-anti-power. That is, among any k consecutive blocks of the same length in x, at least two of them are equal. We say that an infinite word x avoids anti-powers if x avoids k-anti-powers for some k.

Periodic words avoid anti-powers, the period length being an upper bound for the maximal number of distinct consecutive blocks of the same length. In the following, we discuss the avoidability of anti-powers for aperiodic words. By Corollary 8, if an aperiodic word avoids anti-powers, then it cannot be uniformly recurrent.

Of course, any word containing at least two different letters cannot avoid 2-anti-powers. For 3-anti-powers, we have the following result.

▶ **Lemma 10.** *Let x be an infinite word. If x avoids 3-anti-powers, then x is a binary word.*

Proof. Suppose x avoids 3-anti-powers and contains three different letters. Then there is a factor of x of the form $u = ab^nc$ with $n \geq 1$ and a, b, c distinct letters. We will extend this factor to the right and force a 3-anti-power for every n. For $n = 1$, the word abc is already an anti-power. Take now $n = 2$. To avoid 3-anti-powers, $abbc$ can only be extended to $abbcb$. In the next step, the only option is $abbcbc$, and after that $abbcbcb$. But now, the word $abbcbcbyy'$ contains a 3-anti-power for every letters y, y'. Suppose now $u = ab^nc$ with $n \geq 3$. If n is odd, let $m = (n-1)/2$ and note that u can be factored as $ab^m \cdot b^{m+1} \cdot c$, so that u will be extended to the right to a 3-anti-power of length $3(m+1)$. If n is even, u can be factored as $u = ab^m \cdot b^{m+1} \cdot bc$, so that again u will be extended to the right to a 3-anti-power of length $3(m+1)$. ◀

Hence, in what follows we will suppose that x is an infinite word over the binary alphabet $\mathbb{A} = \{0, 1\}$.

▶ **Proposition 11.** *Let x be an infinite word. If x avoids 3-anti-powers, then it cannot contain a factor of the form 10^n1 or 01^n0 with $n > 1$.*

Proof. Suppose that x contains a factor of the form $u = 10^n1$ with $n > 1$ (the other situation is symmetric). The cases $n = 2, 3, 4, 5$ can be checked by computer, so let us suppose $n \geq 6$.

Suppose first n even, and write $n = 2m$. Since $u = 10^{m-1} \cdot 0^m \cdot 01$, any extension of u to the right will produce a 3-anti-power of length $3m$. If n is odd, $n = 2m + 1$, then we can write $u = 10^{m-1} \cdot 0^m \cdot 001$, so that any extension of u to the right will produce a 3-anti-power of length $3m$. ◀

▶ **Corollary 12.** *Let x be an infinite word avoiding 3-anti-powers. Then x is ultimately periodic.*

Actually, from Proposition 11, we have that an infinite word avoiding 3-anti-powers can only be of the form $x = (01)^\omega$, $x = 01^\omega$, or $x = 0^n10^\omega$ for some $n > 0$, up to exchanging letters.

▶ **Proposition 13.** *There exist aperiodic words avoiding 4-anti-powers.*

Proof. We exhibit an example of an aperiodic word avoiding 4-anti-powers. Let $(\alpha_i)_{i\geq 1}$ be an increasing sequence of positive integers with $\alpha_{i+1} \geq 5\alpha_i$ for each $i \geq 1$. Let $x \in \{0, 1\}^{\mathbb{N}}$

be defined by $x_n = 1$ if $n = \alpha_i$ for some $i \geq 1$, and $x_n = 0$ otherwise. Clearly x is aperiodic. Moreover, given $m \geq 0$ and $n \in \mathbb{N}$, if $|x_{m+1}x_{m+2}\cdots x_{m+n}|_1 \geq 2$, then for some $i \geq 1$

$$m + 1 \leq \alpha_i < 5\alpha_i \leq \alpha_{i+1} \leq m + n$$

and hence $n > 4\alpha_i \geq 4(m + 1)$ whence $m + 1 < n/4$. We claim that x avoids 4-anti-powers. In fact, suppose to the contrary that for some $m \geq 0$ and $n \in \mathbb{N}$ we have $x_{m+1}\cdots x_{m+n}$, $x_{m+n+1}\cdots x_{m+2n}$, $x_{m+2n+1}\cdots x_{m+3n}$, and $x_{m+3n+1}\cdots x_{m+4n}$ are pairwise distinct. Then at least three of the four blocks must contain an occurrence of 1. Thus $|x_{m+n+1}\cdots x_{m+4n}|_1 \geq 2$ from which it follows that $m + n + 1 < 3n/4$ and hence $m + 1 < 0$, a contradiction. ◀

The word in the previous proposition is not recurrent. It is natural to ask whether there exist recurrent words avoiding 4-anti-powers. We do not know the answer. However, we can state the following result.

▶ **Proposition 14.** *There exist aperiodic recurrent words avoiding* 6*-anti-powers.*

Proof. We exhibit an example of an aperiodic recurrent word avoiding 6-anti-powers. Let $w_0 = 0$ and $w_n = w_{n-1}1^{3|w_{n-1}|}w_{n-1}$ for every $n > 0$. Let w be the infinite word obtained as the limit of the sequence of words $(w_n)_{n\geq 1}$. Then clearly w is recurrent. Without loss of generality, we can assume that each occurrence of w_n in w is preceded and followed by $1^{3|w_n|}$, since w avoids 6-anti-powers if $1^\infty w$ does.

Let $v = v_1v_2\cdots v_6$ be a non-empty factor of w of length $6k$. Let n be the largest integer such that $|w_n| = 5^n < 2k$. By the hypothesis on n, no v_i can intersect two occurrences of w_n.

Suppose first that for some i, v_i is contained as a factor in w_n. By the hypothesis on n, neither $v_{i-1}v_i$ nor v_iv_{i+1} is contained in w_n. Since w_n is preceded and followed by $1^{3|w_n|}$, either v_{i-3} and v_{i-2} (if $i \geq 4$) or v_{i+2} and v_{i+3} (if $i < 4$) are both equal to 1^k, so that v cannot be an anti-power.

If instead no v_i is contained as a factor in w_n, then one of the following cases must hold:

(i) There is an occurrence of w_n intersecting v_i and the next occurrence of w_n intersects v_{i+1}. In this case, either v_{i-3} and v_{i-2} or v_{i+3} and v_{i+4} are both equal to 1^k.

(ii) There is an occurrence of w_n intersecting v_i and the next occurrence of w_n intersects v_{i+2}, so that $v_{i+1} = 1^k$. In this case, either v_{i-2} or v_{i+4} must be equal to 1^k.

(iii) There are two consecutive blocks v_i, v_{i+1} both equal to 1^k.

In all cases, v cannot be an anti-power. ◀

5 Conclusions and open problems

We proved that every infinite word contains powers of any order or anti-powers of any order, that is, the existence of powers or anti-powers is an unavoidable regularity. This result can also be stated in the following finite version.

▶ **Theorem 15.** *For every integers $k > 1$ and $r > 1$ there exists $N = N(k, r)$ such that every word of length N contains a k-power or an r-anti-power. Furthermore, for $k > 2$, one has $k^2 - 2 \leq N(k, k) \leq k^5 + k^3$.*

The upper bound follows from the proof of Theorem 4. For the lower bound, it is easy to prove that the word $(0^{k-1}1)^{k-2}0^{k-2}10^{k-2}$ of length $k^2 - 3$ avoids both k-powers and k-anti-powers, for any $k > 2$.

In the case of binary alphabet, it can be verified that $N(2, 2) = 2$, $N(3, 2) = 3$, $N(2, 3) = 4$, $N(3, 3) = 9$, $N(4, 3) = 12$, $N(3, 4) > 16$, $N(4, 4) > 16$. We do not know how these numbers

grow. Moreover, the bounds on $N(k, r)$ given in Theorem 15 can probably be improved by a deeper analysis of the function $N(k, r)$.

Concerning the avoidability of anti-powers, we proved that there exist words avoiding 4 anti-powers and that there exist recurrent words avoiding 6-anti-powers. A natural problem is therefore that of determining what is the least k such that there exists a recurrent word avoiding k-anti-powers.

Another possible direction of investigation is related to the possible lengths of anti-powers appearing in a word. For an aperiodic uniformly recurrent word x, define $ap(x, k) = \min(AP(x, k))$, i.e., the minimum length m for which the prefix of x of length km is a k-anti-power. The first values of this function for the Thue-Morse word are displayed in Table 1 (where the value of $ap(x, k)$ is the ratio between the length of the k-anti-power prefix and the order k). We wonder whether it is possible to link the behavior of the function $ap(x, k)$ to the combinatorics of the word x, at least for special classes of words. For example, the values reported in Table 1 suggest that for the Thue-Morse word the function $ap(x, k)$ grows linearly in k.

Acknowledgements. We thank the anonymous referees for their careful reading of the paper and Filippo Mignosi for useful discussions.

References

1 J.-P. Allouche and J. Shallit. The ubiquitous prouhet-thue-morse sequence. In *Sequences and their Applications: Proceedings of SETA'98*, pages 1–16. Springer, 1999.

2 A. Condon, J. Maňuch, and C. Thachuk. The complexity of string partitioning. *J. Discrete Algorithms*, 32(C):24–43, May 2015. doi:10.1016/j.jda.2014.11.002.

3 A. de Luca and S. Varricchio. *Finiteness and Regularity in Semigroups and Formal Languages*. Monographs in Theoretical Computer Science (An EATCS Series). Springer, 1st edition, 1999.

4 H. Fernau, F. Manea, R. Mercaş, and M. L. Schmid. Pattern Matching with Variables: Fast Algorithms and New Hardness Results. In *32nd International Symposium on Theoretical Aspects of Computer Science (STACS 2015)*, volume 30 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 302–315, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.STACS.2015.302.

5 M. Lothaire. *Combinatorics on Words, 2nd edition*. Cambridge Mathematical Library, Cambridge University Press, 1997.

6 M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, 2002.

7 M. L. Schmid. Computing equality-free and repetitive string factorisations. *Theoret. Comput. Sci.*, 618:42–51, March 2016. doi:10.1016/j.tcs.2016.01.006.

8 A. Thue. Uber unendliche zeichenreihen. *Kra. Videnskab. Selsk. Skrifter. I. Mat.-Nat. Kl.*, pages 1–22, 1906.

On Equivalence and Uniformisation Problems for Finite Transducers*

Emmanuel Filiot[†1], Ismaël Jecker[2], Christof Löding[3], and Sarah Winter[4]

1 **Université Libre de Bruxelles – F.R.S.-FNRS, Brussels, Belgium**
2 **Université Libre de Bruxelles – F.R.S.-FNRS, Brussels, Belgium**
3 **RWTH Aachen, Aachen, Germany**
4 **RWTH Aachen, Aachen, Germany**

Abstract

Transductions are binary relations of finite words. For rational transductions, i.e., transductions defined by finite transducers, the inclusion, equivalence and sequential uniformisation problems are known to be undecidable. In this paper, we investigate stronger variants of inclusion, equivalence and sequential uniformisation, based on a general notion of transducer resynchronisation, and show their decidability. We also investigate the classes of finite-valued rational transductions and deterministic rational transductions, which are known to have a decidable equivalence problem. We show that sequential uniformisation is also decidable for them.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases Transducers, Equivalence, Uniformisation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.125

1 Introduction

Transductions generalise finite word languages to binary relations of finite words. The notion of rationality for languages, and its correspondence with finite automata, has been extended to transductions and finite automata over pairs of words, called *finite transducers* [2]. In this paper, we study decision problems for finite transducers and prove new decidability results.

Finite transducers. (Finite) transducers are nondeterministic finite automata whose transitions are labelled by pairs of words. The (rational) transduction $\mathcal{R}_T$ defined by a transducer T consists of all the pairs of words (u, v) obtained by concatenating the pairs occurring on transitions of its successful computations. In this paper, we follow a dynamic vision of transducers, as a machine that processes input words u and produces output words v. Therefore, we may speak of the domain of a transduction, as the language of input words that admit at least one output word.

Equivalence problem. Unlike finite automata, finite transducers have undecidable inclusion and equivalence problems [17, 14], even when restricted to unary alphabets [19]. The largest known classes with decidable equivalence problem are those of finite-valued transducers and

* A full version of this paper can be found in [12].

† Emmanuel Filiot is research associate at FNRS. This work is supported by the ARC project *Transform* (French speaking community of Belgium), the FNRS PDR project *Flare*, and the French ANR project ExtStream.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 125; pp. 125:1–125:14

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

deterministic transducers. A transducer is finite-valued if it produces at most k outputs per input, for a bound k that only depends on the transducer. It is decidable whether a transducer is k-valued for a given k [18], and whether there exists k such that it is k-valued [33]. Any finite-valued transducer is known to be (effectively) equivalent to a finite union of unambiguous transducers [32], and thus to a finitely ambiguous transducer. Equivalence of k-ambiguous transducers was shown to be decidable in [18], and equivalence of k-valued transducers was first shown to be decidable in [20, 32]. Other algorithms with better complexities appeared later, and the best known algorithm runs in exponential time, for a fixed k [8].

A transducer is deterministic if the transitions are deterministic in the classical sense, and furthermore each state processes either only input symbols or only output symbols. The class of deterministic rational transductions is also referred to as DRat, and it strictly extends the class of synchronous rational transductions (also called automatic relations), see e.g. [7] for an overview of these sub-classes of rational transductions. As opposed to the class of finite-valued transducers, it is undecidable whether a transduction is equivalent to a deterministic transduction [14]. However, the equivalence problem for DRat is known to be decidable [3], even in polynomial time [15]. This makes this class an interesting candidate for further investigations of decision problems.

Uniformisation problem. Two classes of interest are the rational and sequential functions, which are respectively defined by 1-valued transducers and sequential transducers. The latter read input words in a deterministic manner, and therefore produce a unique output word for each input. There are rational functions that are not sequential, but it is decidable in PTime whether a transducer defines a sequential function [34]. Since rational transductions do not define, in general, functions, an interesting question is whether a unique output word can be picked for each input word of a rational transduction R, in a regular way, thus defining a function $f \subseteq R$ with the same domain as R. Such a function f is called a *uniformiser* of R. It is known that any rational transduction admits a rational uniformiser [23, 10] and, in the case of DRat, even a *lexicographic* uniformiser that picks the smallest output words according to a lexicographic order, making the uniformiser only depend on the transduction [21, 27]. In this paper, we are interested in sequential uniformisers. Even rational functions do not admit sequential uniformisers in general, and therefore this gives rise to a decision problem: Given a finite transducer, does it admit a sequential uniformiser? It is worth noting that even if any rational transduction R can be uniformised by a rational uniformiser U, the sequential uniformisability of R does not imply, in general, that *any* of the uniformisers U is equivalent to a sequential transducer. As a matter of fact, it is known that the sequential uniformisation problem is undecidable for rational transductions [6].

The sequential uniformisation problem echoes a similar problem introduced by Church, the *synthesis problem*, which currently receives a lot of attention from the computer-aided verification community in the context of open reactive systems (see [24, 13, 4] for some work on this subject from the last decade). This problem asks whether given a logical specification of a system, there exists an implementation that satisfies it. In this context, reactive systems are non-terminating systems that react to some unpredictable environment stimuli in a *synchronised* fashion: for each environment input, they produce an output in a deterministic manner, such that the specification is met in the limit. Their executions are modelled by infinite words over a product alphabet, and the interaction with the environment makes game theory a powerful tool in this context. A seminal result due to Büchi and Landweber shows that the synthesis problem is decidable for MSO specifications [22] (see [31] for a modern presentation and an overview).

Restricted to finite words, the sequential uniformisation problem extends Church's problem to more general (asynchronous) classes of specifications and implementations, where the transduction R is the specification and the sequential uniformiser f the implementation.

Resynchronisers. One of the main difficulty of transducers is that two equivalent transducers may produce their outputs very differently: One transducer may go fast and be ahead of the other. By tagging symbols with two colours (for input and output), transductions can be seen as languages, called *synchronisation languages*. It is known by Nivat's theorem that rational transductions are synchronised by regular languages [26], and any transducer defines a regular synchronisation language. Other correspondences between classes of synchronisation languages and classes of rational transductions have been established in [11]. However in general, there is an infinite number of synchronisation languages for a single transduction, making problems such as equivalence and sequential uniformisation undecidable. To overcome this difficulty, Bojanczyk has introduced *transductions with origin information*, which amounts to adding the synchronisation information into the semantics of transducers, via an origin function mapping output positions to their originating input positions [5]. The main result of [5] is a machine-independent characterisation of transductions (with origin information) defined by two-way transducers. With respect to the equivalence problem, considering the origin information makes the problem easy: two transducers define the same transduction with same origin mappings if they have the same synchronisation language. In this paper, we generalise this idea and propose decision problems modulo resynchronisation. A *resynchroniser* $\mathbb{S}$ is a transduction, mapping a synchronisation language to another one. Then, we consider related equivalence and sequential uniformiser problems: for instance, given two transducers, are their synchronisation languages equal modulo $\mathbb{S}$? For the identity resynchroniser, it is the same as origin-equivalence.

Contributions. As a first contribution, we show that inclusion, equivalence and sequential uniformisation are decidable modulo *rational* resynchronisers. For equivalence, it easily reduces to an automata equivalence problem. For sequential uniformisation, it boils down to solving a two-player safety game. We then consider a particular class of resynchronisers, the *k-delay resynchronisers*, that can apply a fixed delay k to a synchronisation language, where the delay is a measure of how ahead an output word is from another one [1]. The k-delay resynchroniser is rational for each k, which implies the decidability of the corresponding decision problem. Interestingly, we show that for the class of real-time transducers (reading at least one input symbol in each transition), k-delay resynchronisers encompass all the power of rational synchronisers with respect to the decision problems we consider.

Our second main contribution is to show that equivalence and sequential uniformisation modulo k-delay resynchronisers are complete for finite-valued transducers. Given two finite-valued transducers, if they are equivalent, then some k can be computed such that they are k-delay equivalent. This yields another, delay-based, proof of the decidability of finite-valued transducer equivalence. We show a similar result for sequential uniformisation, by a pumping argument based on an analysis of the idempotent elements in the transition monoid of finitely-ambiguous transducers. This implies the following new result:

▶ **Theorem 16.** *The sequential uniformisation problem for finite-valued transducers is decidable.*

Finally, we consider deterministic rational transductions, i.e. transductions defined by deterministic transducers. A deterministic transducer is a deterministic automaton whose

states are partitioned into input and output states. They process pairs of words (u, v) as follows: two reading heads placed on u and v respectively process sequentially symbols from u and v. Whenever the current state is an input state, a symbol from u is read and the (input) head moves one step forward, and symmetrically on v when the current state is an output state. The equivalence problem for deterministic transducers is decidable [3], unlike the inclusion problem [14]. Our third main contribution is a decidability proof for the sequential uniformisation problem for deterministic rational transductions, extending a corresponding result for automatic relations from [6].

▶ **Theorem 18.** *The sequential uniformisation problem for deterministic transducers is decidable.*

Structure of the paper. In Section 2, we introduce automata, transducers and decision problems for them. In Section 3, we define the notion of resynchronisers for transductions and study their associated decision problems. We also introduce the particular class of bounded delay resynchronisers. In Section 4, we study the class of finite-valued rational transductions and prove decidability of their sequential uniformisation. Finally in Section 5, we prove decidability of sequential uniformisation for deterministic rational transductions.

2 Automata and Transducers

Let $\mathbb{N}$ denote the set of non-negative integers $\{0, 1, \dots\}$, and for every $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \dots, n\}$. Given a finite set A, let $|A|$ denote its cardinality.

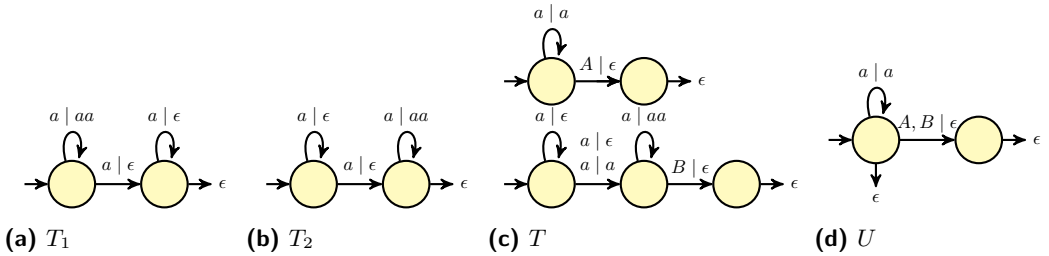
Languages and Transductions of Words. An alphabet Σ is a finite set of symbols. The elements of the free monoid Σ^* are called *words* over Σ. The length of a word w is the number of its symbols. It is written $|w|$. The empty word (of length 0) is denoted by ϵ, and $\Sigma^+ = \Sigma^* \setminus \{\epsilon\}$. The set Σ^* can be partially ordered by the word prefix relation $\preceq$.

We denote by Σ^{-1} the set of symbols σ^{-1} for all $\sigma \in \Sigma$. Any word $u \in (\Sigma \cup \Sigma^{-1})^*$ can be reduced into a unique irreducible word $\overline{u}$ by the equations $\sigma\sigma^{-1} = \sigma^{-1}\sigma = \epsilon$ for all $\sigma \in \Sigma$. Let G_Σ be the set of irreducible words over $\Sigma \cup \Sigma^{-1}$. The set G_Σ equipped with concatenation $u.v = \overline{uv}$ is a group, called the free group over Σ. We denote by u^{-1} the inverse of u. E.g. $(a^{-1}bc)^{-1} = c^{-1}b^{-1}a$. For $u \in G_\Sigma$, we denote by $|u|$ its number of symbols. E.g., $|a^{-1}b^{-1}| = 2$, $|a^{-1}bc^{-1}| = 3$.

A *language* L over Σ is a subset of Σ^*. A *transduction* R over Σ is a subset of $\Sigma^* \times \Sigma^*$. The *domain* of R is the set $\mathrm{dom}(R) = \{u \mid \exists v \in \Sigma^* \cdot (u, v) \in R\}$. For a word $u \in \Sigma^*$, we denote by $R(u)$ the set $\{v \mid (u, v) \in R\}$, and extend this notation to languages L by $R(L) = \bigcup_{u \in L} R(u)$. When R is a function, we simply write $R(u) = v$ instead of $R(u) = \{v\}$. Finally, we denote by id_{Σ^*} the identity relation on Σ^*.

Automata. A (finite state) *automaton* over an alphabet Σ is a tuple $A = (Q, I, F, \Delta)$, where Q is the finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\Delta \subseteq Q \times \Sigma^* \times Q$ is the finite transition relation. Given a transition $(q, w, q') \in \Delta$, q is called its source, q' its target, and w its label. An automaton is called *deterministic* if each of its transition is labelled by a single letter, and it admits no pair of transitions that have same source, same label, and different targets.

A *run* of A on a word $u \in \Sigma^*$ from state q to state p is either a single state $q \in Q$ if $u = \epsilon$ and $q = p$, or a word $r = (q_1, u_1, p_1)(q_2, u_2, p_2) \dots (q_n, u_n, p_n) \in \Delta^+$ if $u \in \Sigma^+$, where $u = u_1 \dots u_n$, $q_1 = q$ and $p_n = p$, and for all $i \in \{1, \dots, n-1\}$, $p_i = q_{i+1}$. We write $q_1 \xrightarrow{u}_A p_n$



(a) T_1 **(b)** T_2 **(c)** T **(d)** U

Figure 1 Transducers such that $T_1 \equiv T_2$ and T is seq-uniformisable by U.

(or simply $q_1 \xrightarrow{u} p_n$) if such a run exists. A run r from a state q to a state p is *accepting* if q is initial and p is final. The *language* recognised by A, written $\mathcal{L}_A$, is the set of words $w \in \Sigma^*$ such that there exists an accepting run of A on w. If B is an automaton, we write $A \subseteq B$ (resp. $A \equiv B$) whenever $\mathcal{L}_A \subseteq \mathcal{L}_B$ (resp. $\mathcal{L}_A = \mathcal{L}_B$).

Transducers. A (finite state) *transducer* over an alphabet Σ is a tuple $T = (Q, I, F, \Delta, f)$, where Q is the finite set of states, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ the set of final states, $\Delta \subseteq Q \times \Sigma^* \times \Sigma^* \times Q$ the transition relation, and $f : F \to \Sigma^*$ the final output function[1]

As for automata, a *run* of a transducer is either a single state or a sequence of transitions. The *input* (resp. *output*) of a run $r = (q_1, u_1, v_1, p_1) \dots (q_n, u_n, v_n, p_n) \in \Delta^*$ is $\text{in}(r) = u_1 \dots u_n$ (resp. $\text{out}(r) = v_1 \dots v_n$). If r is reduced to a single state, its input and output are both ϵ. We say that r is a run of T on $u_1 \dots u_n$. We write $q \xrightarrow{u|v} p$ to mean that there exists a run on input $u \in \Sigma^*$ whose output is $v \in \Sigma^*$. In particular, $q \xrightarrow{\epsilon|\epsilon} q$ for all $q \in Q$. The notion of accepting run of automata carries over to transducers. The *transduction* recognised by T, written $\mathcal{R}_T$ is the set of pairs $(u, vf(p)) \in \Sigma^* \times \Sigma^*$ such that there exists an accepting run of T on u from a state q to a state p whose output is v. We define $\text{dom}(T)$ as $\text{dom}(\mathcal{R}_T)$. The class of *rational transductions* is the class of relations definable by finite state transducers.

The *input automaton* of T is the automaton $A = (Q, I, F, \Delta')$ over the alphabet Σ, where $\Delta' = \{(q, u, q') | (q, u, v, q') \in \Delta\}$. A transducer is called *real time* if each of its transition is labelled by a pair (a, v), where $a \in \Sigma$ and $v \in \Sigma^*$. A transducer is called *sequential* if its input automaton is deterministic[2]. Sequential transducers define *sequential transductions*. A transducer is *trim* if all its accessible states are co-accessible, i.e. for all $q \in Q$, $q_0 \in I$, $u, v \in \Sigma^*$, if $q_0 \xrightarrow{u|v} q$, then there exist $u', v' \in \Sigma^*$ and $q_f \in F$ such that $q \xrightarrow{u'|v'} q_f$.

Decision Problems for Transducers. Let T_1, T_2 be two transducers over an alphabet Σ. We write $T_1 \subseteq T_2$ whenever $\mathcal{R}_{T_1} \subseteq \mathcal{R}_{T_2}$. The *inclusion problem* asks, given T_1, T_2, whether $T_1 \subseteq T_2$. Similarly, we define the equivalence problem by asking whether $\mathcal{R}_{T_1} = \mathcal{R}_{T_2}$, denoted $T_1 \equiv T_2$. Let T be a transducer over an alphabet Σ. A uniformiser of T is a transducer U such that $U \subseteq T$ and $\text{dom}(U) = \text{dom}(T)$. We sometimes write seq-uniformiser for sequential uniformiser. The *sequential uniformisation problem* (seq-uniformisation problem) asks, given a transducer T over Σ, whether T admits a seq-uniformiser.

[1] Allowing final output functions does not increase the expressiveness of the general model, but is required to define the notion of sequentiality

[2] The term subsequential was originally used in the literature. We follow the terminology of [25].

▶ **Example 1.** The transducers T_1 and T_2 of Fig. 1 both define the transduction $\{(a^n, a^{2i}) \mid n \geq 1, 0 \leq i \leq n-1\}$, thus are equivalent. The transducer T is over the alphabet $\{a, A, B\}$ and defines the transduction $\{(a^nA, a^n) \mid n \geq 0\} \cup \{(a^nB, a^i) \mid n \geq 1,\ 0 \leq i \leq 2n-1\}$. It is uniformisable by the sequential transducer U with $\mathcal{R}_U = \{(a^n\alpha, a^n) \mid n \geq 0, \alpha \in \{A, B\}\}$.

▶ **Theorem 2** ([17, 6]). *The inclusion, equivalence and sequential uniformisation problems for rational transductions are undecidable.*

3 Decision Problems Modulo Resynchronisers

A pair $(u, v) \in \Sigma^* \times \Sigma^*$ can be represented by a coloured word over $\Sigma \times \{\mathtt{i}, \mathtt{o}\}$, where the colours indicate whether a symbol in Σ is an input or an output symbol. Such a coloured word is called a synchronisation of (u, v). More generally, any language over the alphabet $\Sigma \times \{\mathtt{i}, \mathtt{o}\}$ represents a transduction $R \subseteq \Sigma^* \times \Sigma^*$, and is called a synchronisation language for R. This way of representing transductions is analysed in [11]. What we call a resynchroniser below, is a transduction of synchronisations, that is, over words in $(\Sigma \times \{\mathtt{i}, \mathtt{o}\})^*$, that preserves the represented pairs. In this section, we study stronger notion of inclusion, equivalence and sequential uniformisation, parametrised by such a resynchroniser. We show their decidability for rational resynchronisers and introduce the class of bounded delay resynchronisers, and show that it has appealing properties.

Synchronisations and resynchronisers. Given an alphabet Σ, we let $\Sigma_{\mathtt{io}} = \Sigma \times \{\mathtt{i}, \mathtt{o}\}$, $\Sigma_{\mathtt{i}} = \Sigma \times \{\mathtt{i}\}$ and $\Sigma_{\mathtt{o}} = \Sigma \times \{\mathtt{o}\}$. For $c \in \{\mathtt{i}, \mathtt{o}\}$, we write σ^c instead of (σ, c). The colouring c can be seen as a morphism $.^c : \Sigma^* \to \Sigma_c^*$ and we write u^c its application on a word $u \in \Sigma^*$. Conversely, for $c \in \{\mathtt{i}, \mathtt{o}\}$, we define two morphisms $\pi_c : \Sigma_{\mathtt{io}}^* \to \Sigma$ that extract the input and output words, by $\pi_c(\sigma^c) = \sigma$ and $\pi_c(\sigma^d) = \epsilon$, for all $\sigma \in \Sigma$, and $d \neq c$. Two words $u, v \in (\Sigma_{\mathtt{io}})^*$ are said to be *equivalent*, denoted by $u \sim_{\mathtt{io}} v$, if $\pi_c(u) = \pi_c(v)$ for all $c \in \{\mathtt{i}, \mathtt{o}\}$. For example, $a^{\mathtt{i}}b^{\mathtt{i}}a^{\mathtt{o}}$ and $a^{\mathtt{i}}a^{\mathtt{o}}b^{\mathtt{i}}$ are equivalent, and both are synchronisations of (ab, a). Any language $L \subseteq \Sigma_{\mathtt{io}}^*$ defines a transduction over Σ defined by $\mathcal{R}_L = \{(\pi_{\mathtt{i}}(w), \pi_{\mathtt{o}}(w)) \in \Sigma^* \times \Sigma^* \mid w \in L\}$, and L is called a *synchronisation* of a transduction $R \subseteq \Sigma^* \times \Sigma^*$ if $\mathcal{R}_L = R$. We also say that L synchronises R. Note that two different languages may synchronise the same transduction.

Mapping a synchronisation to another one is done through the notion of resynchroniser. A *resynchroniser* is a transduction $\mathbb{S} \subseteq \Sigma_{\mathtt{io}}^* \times \Sigma_{\mathtt{io}}^*$, such that (i) $\mathrm{id}_{\Sigma_{\mathtt{io}}^*} \subseteq \mathbb{S}$ and (ii) for all $(w, w') \in \mathbb{S}$, it holds $w \sim_{\mathtt{io}} w'$. For instance, the identity relation $\mathrm{id}_{(\Sigma_{\mathtt{io}})^*}$ is a resynchroniser that we shall denote by $\mathbb{I}$, as well as the relation $\mathbb{U}_{\Sigma_{\mathtt{io}}} = \{(w, w') \in \Sigma_{\mathtt{io}}^* \times \Sigma_{\mathtt{io}}^* \mid w \sim_{\mathtt{io}} w'\}$, called the *universal resynchroniser* over $\Sigma_{\mathtt{io}}$. We write $\mathbb{U}$ instead of $\mathbb{U}_{\Sigma_{\mathtt{io}}}$ when it is clear from the context. Note that for any resynchroniser $\mathbb{S}$, we have $\mathrm{id}_{\Sigma_{\mathtt{io}}^*} \subseteq \mathbb{S} \subseteq \mathbb{U}$. The properties (i) and (ii) of resynchronisers are chosen such that they preserve the represented transductions, as stated in the proposition below.

▶ **Proposition 3.** *For all $L \subseteq \Sigma_{\mathtt{io}}^*$ and all resynchronisers $\mathbb{S} \subseteq \Sigma_{\mathtt{io}}^* \times \Sigma_{\mathtt{io}}^*$, $\mathcal{R}_L = \mathcal{R}_{\mathbb{S}(L)}$.*

Classes of synchronisation languages and their correspondence with the classes of rational relations they synchronise have been studied in [11]. We can formulate in this framework a result known as Nivat's theorem [26] as follows.

▶ **Theorem 4.** *[26] A transduction R is rational iff it is synchronised by a regular language.*

A regular language synchronising a rational transduction can be obtained as follows. Any transducer $T = (Q, I, F, \Delta, f)$ naturally defines a regular synchronisation for $\mathcal{R}_T$ by its

underlying automaton, which is the automaton obtained by concatenating the pairs of input and output words on the transitions and marking them with the respective symbol from $\{\mathtt{i}, \mathtt{o}\}$. Formally, it is the automaton $A = (Q \cup \{q_\dashv\}, I, \{q_\dashv\}, \Delta')$ over $\Sigma_{\mathtt{io}}$, where $\Delta' = \{(q, v^{\mathtt{i}}w^{\mathtt{o}}, q') | (q, v, w, q') \in \Delta\} \cup \{(q, f(q)^{\mathtt{o}}, q_\dashv) | q \in F\}$. The *language recognised by* T is the language recognised by its underlying automaton, denoted by $\mathcal{L}_T$, i.e. $\mathcal{L}_T = \mathcal{L}_A$. Obviously, $\mathcal{L}_T$ is a synchronisation for the relation $\mathcal{R}_T$ (proving one direction of Thm 4).

Decision problems for transducers modulo resynchronisers. Let Σ be an alphabet, $\mathbb{S}$ be a resynchroniser over $(\Sigma_{\mathtt{io}})^*$, and T_1, T_2 be two transducers over Σ. We say that T_1 is *included in* T_2 *modulo* $\mathbb{S}$ (or $\mathbb{S}$-*included*), denoted by $T_1 \subseteq_{\mathbb{S}} T_2$, if $\mathcal{L}_{T_1} \subseteq \mathbb{S}(\mathcal{L}_{T_2})$. We say that T_1 is *equivalent to* T_2 *modulo* $\mathbb{S}$ (or $\mathbb{S}$-*equivalent*), denoted by $T_1 \equiv_{\mathbb{S}} T_2$, if $T_1 \subseteq_{\mathbb{S}} T_2$ and $T_2 \subseteq_{\mathbb{S}} T_1$. For a fixed synchroniser $\mathbb{S}$, the $\mathbb{S}$-*inclusion (resp.* $\mathbb{S}$-*equivalence) problem* asks, given two transducers T_1, T_2 over Σ, whether $T_1 \subseteq_{\mathbb{S}} T_2$ (resp. $T_1 \equiv_{\mathbb{S}} T_2$). We say that T_1 is *sequentially* $\mathbb{S}$-*uniformisable* if it admits a sequential uniformiser U such that $U \subseteq_{\mathbb{S}} T_1$, and in that case U is called a sequential $\mathbb{S}$-uniformiser of T_1 (seq-$\mathbb{S}$-uniformiser for short). The *sequential* $\mathbb{S}$-*uniformisation problem* asks whether a given transducer is seq-$\mathbb{S}$-uniformisable.

It should be clear from the definition that $\mathbb{I}$-inclusion implies $\mathbb{S}$-inclusion for any resynchroniser $\mathbb{S}$, which in turn implies $\mathbb{U}$-inclusion. As a matter of fact, it is easy to see that $\mathbb{U}$-inclusion is equivalent to classical inclusion. The same remarks can be made for equivalence and sequential uniformisation, and as a consequence of Theorem 2, we get:

▶ **Theorem 5.** *The* $\mathbb{U}$*-inclusion,* $\mathbb{U}$*-equivalence, sequential* $\mathbb{U}$*-uniformisation problems for rational transductions are undecidable.*

Decision problems for transducers modulo rational resynchronisers. The $\mathbb{U}$-decision problems are undecidable, this raises the question whether there is an interesting class of resynchronisers for which we can recover decidability. It turns out that $\mathbb{U}$ is not rational. In contrast, we show that, as long as $\mathbb{S}$ is rational, the $\mathbb{S}$-decision problems are reducible to the $\mathbb{I}$-decision problems, which in turn can be solved by reduction to decidable problems of automata and two-player games.

▶ **Proposition 6.** *The* $\mathbb{I}$*-inclusion and* $\mathbb{I}$*-equivalence problems are* PSpace-complete*. The sequential* $\mathbb{I}$*-uniformisation problem is* ExpTime-complete*.*

Proof. First, note that $T_1 \subseteq_{\mathbb{I}} T_2$ iff $\mathcal{L}_{T_1} \subseteq \mathcal{L}_{T_2}$ iff $A_1 \subseteq A_2$, where A_1, A_2 are the underlying automata of T_1, T_2 respectively. Automata inclusion and equivalence problems are PSpace-complete, and they easily reduce (by putting ϵ outputs) to $\mathbb{I}$-inclusion and $\mathbb{I}$-equivalence.

To get ExpTime membership of seq-$\mathbb{I}$-uniformisation, for a transducer T, we construct a two-player safety game $G_T = (V = V_{\mathsf{In}} \uplus V_{\mathsf{Out}}, v_0, E)$ between an adversary (Player In) who picks input symbols and controls positions in V_{In}, and a protagonist (Player Out) who picks sequences of output symbols and controls positions in V_{Out}. Wlog we assume that T has no final output function, by adding an endmarker $\dashv$ to words of its domain. Let $A = (Q, q_0, F, \delta)$ be a complete DFA equivalent to the the underlying automaton of T (whose size is at most exponential in the size of T). Player positions have three components: a residual language[3] of $\mathrm{dom}(T)$ controlling the possible continuations of the input word chosen so far by Player In, a state of A and a round $r \in \{\mathsf{In}, \mathsf{Out}\}$. Let $\mathcal{D} = \{u^{-1}\mathrm{dom}(T) \mid u \in \Sigma^*\}$ be the set of residuals of $\mathrm{dom}(T)$ (represented by the states of the minimal DFA for $\mathrm{dom}(T)$, computed in

[3] A residual of a language L over some alphabet Σ is a language $u^{-1}L = \{v \mid uv \in L\}$ for $u \in \Sigma^*$.

exponential time in the size of T). Then, $V_{\mathsf{In}} = \mathcal{D} \times Q \times \{\mathsf{In}\}$ and $V_{\mathsf{Out}} = \mathcal{D} \times Q \times \{\mathsf{Out}\}$. The initial position is $v_0 = (\mathrm{dom}(T), q_0, \mathsf{In})$ and the edge relation E as follows: from a position (D, q, In), there are outgoing edges to all states $(\sigma^{-1}D, \delta(q, \sigma^{\mathtt{i}}), \mathsf{Out})$ for all $\sigma \in \Sigma$. From a position (D, q, Out), Player Out can pick any state $q' \in Q$ such that there exists a sequence $v \in \Sigma_{\mathtt{o}}^*$ such that $q \xrightarrow{v}_A q'$, and in that case an outgoing edge to (D, q', In) is added to E. The *unsafe positions* for Player Out are all positions (D, q, In) such that $\epsilon \in D$ and $q \notin F$: at such positions, Player In could choose to terminate the sequence of input symbols (while staying in $\mathrm{dom}(T)$ since $\epsilon \in D$) and the sequence of output symbols chosen by Player Out, mixed with the input symbols chosen by Player In, does not belong to $L(A)$ (since $q \notin F$).

It can be shown that Player Out has a strategy to avoid the unsafe positions in G_T iff there exists a seq-$\mathbb{I}$-uniformiser of T. We briefly explain how to extract a uniformiser from a memoryless winning strategy. A memoryless winning strategy of Player Out can be represented by a function $\lambda : V_{\mathsf{Out}} \to V_{\mathsf{In}}$. The uniformiser U_λ has V_{In} as state set. Let v be a state of U_λ where $v = (D, q, \mathsf{In}) \in V_{\mathsf{In}}$ and $\sigma \in \Sigma$. Let $v' = (\sigma^{-1}D, \delta(q, \sigma^{\mathtt{i}}), \mathsf{Out})$ and $v'' = \lambda(v')$. By definition of G_T, $v'' = (\sigma^{-1}D, q', \mathsf{In})$ such that $\delta(q, \sigma^{\mathtt{i}}) \xrightarrow{w}_A q'$ for some word $w \in \Sigma^*$. We then add the transition (v, σ, w, v'') to U_λ. The word w can be uniquely chosen by taking the minimal word for some lexicographic order, making U_λ sequential. Accepting states are states $v = (D, q, \mathsf{In})$ with $\epsilon \in D$, thus ensuring $\mathrm{dom}(U_\lambda) = \mathrm{dom}(T)$. Since λ is winning, we then necessarily have $q \in F$, ensuring that the sequence of input and output symbols read and produced by U_λ belongs to $\mathcal{L}_A$, making U_λ an $\mathbb{I}$-uniformiser.

Since safety games can be solved in polynomial time and G_T has exponential size, we get the result. The results on safety games that we use here can be found, e.g., in [16].

For the ExpTime lower bound, we note that in our formalism we can model the synchronous uniformisation (or synthesis) problem, as considered in [28] for infinite words, by taking synchronisations that strictly alternate between input and output. It seems to be common knowledge in the synthesis community that the synchronous uniformisation problem is ExpTime-complete if the relation is given by a nondeterministic automaton. However, we were not able to find a reference for this result. We thus give a reduction from the acceptance problem for alternating Pspace Turing machines in a long version. ◀

For all transducers T and resynchronisers $\mathbb{S}$, $\mathbb{S}(\mathcal{L}_T)$ is a regular synchronisation language and by Nivat's theorem (Theorem 4), there exists a transducer $T^{\mathbb{S}}$ such that $\mathcal{L}_{T^{\mathbb{S}}} = \mathbb{S}(\mathcal{L}_T)$. It implies that the seq-$\mathbb{S}$-uniformisation of T reduces to the seq-$\mathbb{I}$-uniformisation of $T^{\mathbb{S}}$. Similar arguments apply for inclusion and equivalence and from Proposition 6 we obtain:

▶ **Theorem 7.** *Let $\mathbb{S}$ be a rational resynchroniser, given as a transducer. The $\mathbb{S}$-inclusion and $\mathbb{S}$-equivalence problems are* PSpace-complete. *The sequential $\mathbb{S}$-uniformisation problem is* ExpTime-complete.

Bounded delay resynchronisers. The notion of *delay* between outputs of transducers is a powerful way of comparing transducers, which has been used, for instance, to characterise sequential functions [2]. Intuitively, the delay between two runs on the same input is a parameter that measures how a run is ahead of the other, and the lag is the maximal delay over prefixes of the two runs. We adapt the notion of delay and lag to coloured words and define delay resynchronisers as resynchronisers that apply a fixed delay to words in $\Sigma_{\mathtt{io}}^*$ (our notion of lag is not related to the one from [11]). Our results show that delay resynchronisers form a fundamental class of resynchronisers.

The *delay* between two words u and v over an alphabet Σ is the element from the free group G_Σ defined by $\mathrm{delay}(u, v) = u^{-1}v$. E.g., $\mathrm{delay}(ab, acd) = b^{-1}cd$. Note that $\mathrm{delay}(u, v) \in \Sigma^*$ iff

$u \preceq v$, and $\text{delay}(u,v) \in (\Sigma^{-1})^*$ iff $v \preceq u$. The *lag mapping* $\text{lag} : (\Sigma_{\mathtt{i}\mathtt{o}})^* \times (\Sigma_{\mathtt{i}\mathtt{o}})^* \to \mathbb{N} \cup \{+\infty\}$ gives the maximal length of the delay between the output part of two words in $(\Sigma_{\mathtt{i}\mathtt{o}})^*$ that have the same input. It is the metric defined by $\text{lag}(u,v) = +\infty$ if $\pi_{\mathtt{i}}(u) \neq \pi_{\mathtt{i}}(v)$. If $\pi_{\mathtt{i}}(u) = \pi_{\mathtt{i}}(v)$, then u and v can be decomposed into $u = u_0 a_1 u_1 \dots u_{n-1} a_n u_n$ and $v = v_0 a_1 v_1 \dots a_{n-1} b_n v_n$, such that $a_1, \dots, a_n \in \Sigma_{\mathtt{i}}$, $u_0, v_0, \dots, u_n, v_n \in (\Sigma_{\mathtt{o}})^*$. Then $\text{lag}(u,v) = \max_{0 \leq i \leq n} |\text{delay}(u_0 \dots u_i, v_0 \dots v_i)|$. As an example, for $n \geq 1$, take $u_n = a^{\mathtt{i}} a^{\mathtt{o}} (a^{\mathtt{i}})^n$ and $v_n = (a^{\mathtt{i}})^n a^{\mathtt{i}} a^{\mathtt{o}}$. Then for all $n \geq 1$, $\text{lag}(u_n, v_n)$=1. Note that the occurrence of $a^{\mathtt{o}}$ in u_n is arbitrary far from that of $a^{\mathtt{o}}$ in v_n.

We now define the k-delay resynchroniser $\mathbb{D}_k$. Intuitively, it can shift output symbols of a word u to the left or to the right, as long as the lag between u and the new word obtained this way is bounded by k. Formally, the *k-delay resynchroniser* is defined by $\mathbb{D}_k = \{(u,v) \in (\Sigma_{\mathtt{i}\mathtt{o}})^2 \mid u \sim_{\mathtt{i}\mathtt{o}} v \wedge \text{lag}(u,v) \leq k\}$. We define the *$k$-inclusion*, *$k$-equivalence* and sequential *k-uniformisation* problems as the corresponding $\mathbb{D}_k$-decision problems, and write $\subseteq_k$ and $\equiv_k$ instead of $\subseteq_{\mathbb{D}_k}$ and $\equiv_{\mathbb{D}_k}$ respectively. We also say that a transduction is *seq-k-uniformisable* if it is seq-$\mathbb{D}_k$-uniformisable. An important property of $\mathbb{D}_k$ is:

▶ **Proposition 8.** *For all $k \geq 0$, $\mathbb{D}_k$ is rational.*

As a direct consequence of the latter proposition and Theorem 7, the k-delay decision problems are all decidable. We can be more precise:

▶ **Theorem 9.** *For all $k \geq 0$, the k-inclusion, k-equivalence and sequential k-uniformisation problems are decidable and* ExpSpace-hard *if k is part of the input. If k is fixed, then the k-inclusion and k-equivalence problems are* PSpace-complete*, and the sequential k-uniformisation problem is* ExpTime-complete*.*

Even if inclusion is undecidable while k-inclusion is decidable, it could be the case that inclusion reduces to k-inclusion, for some k that cannot be computed. We show that that it is not the case, by using the transducers of Fig. 1.

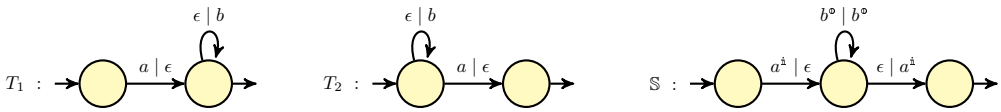
▶ **Proposition 10.** *There exist transducers T_1, T_2, T such that $T_1 \equiv T_2$, $T_1 \subseteq T_2$ and T is seq-uniformisable, but for all $k \geq 0$, $T_1 \not\equiv_k T_2$, $T_1 \not\subseteq_k T_2$, and T is not seq-k-uniformisable.*

Proof. Consider T_1, T_2 of Fig.1 and pairs of the form $(a^{n+1}, a^{2n}) \in \mathcal{R}_{T_1} = \mathcal{R}_{T_2}$. They both accept these pairs but T_2 will be arbitrarily late compared to T_1. Consider now the transducer T and its sequential uniformiser U. On inputs $a^n B$, U will be arbitrarily ahead of T, and one can show that is the case for *any* seq-uniformiser of T. ◀

Finally, we show that for real-time transductions, k-delay resynchronisers subsume any rational resynchroniser $\mathbb{S}$, in the sense that $\mathbb{S}$-inclusion implies k-inclusion, for some k that depends on $\mathbb{S}$. Similar results hold for equivalence and sequential uniformisation. The idea is that a rational synchroniser cannot advance or delay the production of outputs arbitrarily far away with a finite set of states.

▶ **Theorem 11.** *Let $\mathbb{S}$ be a rational synchroniser (given by a transducer). Let T_1, T_2, T be real-time transducers. There exists a computable integer $k \in \mathbb{N}$ such that:* (i) *if $T_1 \subseteq_{\mathbb{S}} T_2$, then $T_1 \subseteq_k T_2$,* (ii) *if $T_1 \equiv_{\mathbb{S}} T_2$, then $T_1 \equiv_k T_2$, and* (iii) *if T is seq-$\mathbb{S}$-uniformisable, then T is seq-k-uniformisable.*

One cannot drop the real-time assumption in the latter theorem. Indeed consider the following transducers $T_1, T_2, \mathbb{S}$, for which $T_1 \equiv_{\mathbb{S}} T_2$ but T_1, T_2 are not k-equivalent for any k:



4 Finite-valued transducers

Let $m \in \mathbb{N}$. We remind the reader that a transducer T is called *m-valued* if each input has at most m outputs, i.e. for all $w \in \mathrm{dom}(T)$, $|\mathcal{R}_T(w)| \leq m$. It is finite-valued if it is m-valued for some m. Finite-valuedness is decidable [33]. We prove that for the class of finite-valued transducers, k-inclusion and sequential k-uniformisation are complete. This yields, for finite-valued transducers, an alternative proof of the decidability of the inclusion problem, and a new result: The decidability of sequential uniformisation.

Let m be a natural number. An automaton A (resp. transducer T) is called *m-ambiguous* if it is real-time[4], and for any word $w \in \mathcal{L}_A$ (resp. $w \in \mathrm{dom}(T)$), there exist at most m accepting runs of A (resp. T) on w. An automaton (transducer) is called *finitely ambiguous* if there exists $m \in \mathbb{N}$ such that it is m-ambiguous, and *unambiguous* if it is 1-ambiguous. Our proofs uses the following known decomposition initially due to Weber:

▶ **Theorem 12.** *[32, 30] Any finite-valued transducer T is (effectively) equivalent to a union of unambiguous transducers.*

We first prove that, for the class of finitely ambiguous transducers, inclusion and equivalence reduces to k-inclusion and k-equivalence for some computable k. We state this result for inclusion, which immediately implies it for equivalence. The proof is based on similar pumping techniques than Lemma 2 in [9].

▶ **Theorem 13.** *Let T_1 and T_2 be two real-time transducers such that T_2 is m-ambiguous. Then there exists a computable integer k such that $T_1 \subseteq T_2 \implies T_1 \subseteq_k T_2$. Moreover, k can be chosen to be exponential in the size of T_2 and linear in the size of T_1.*

Since k-inclusion is decidable by Theorem 7, Theorem 13 implies that the inclusion and equivalence problems are decidable for finitely ambiguous transducers. From the decomposition of Theorem 12, we obtain an alternative proof of the decidability of equivalence of finite-valued transducers, which was proved in [20, 32].

▶ **Corollary 14** ([20, 32] Alternative proof)**.** *The inclusion and equivalence problems for finite-valued transducers are decidable.*

We now prove the two corresponding results for the sequential uniformisation problem.

▶ **Theorem 15.** *Let T be a real-time trim transducer given as a finite union of unambiguous transducers. Then there exists a computable integer N_T such that if T is sequentially uniformisable, then it is sequentially N_T-uniformisable.*

Sketch of proof. If T is seq-uniformisable, then there exists a sequential uniformiser U of T such that $\mathrm{dom}(U) = \mathrm{dom}(T)$ and $U \subseteq T$. The latter inclusion implies, by Theorem 13, that there exists an integer k such that $U \subseteq_k T$, and so U is a seq-k-uniformiser of T. However, k depends on the number of states of the hypothetical uniformiser U. We show how to construct, by simulating the behaviour of U, another seq-N_T-uniformiser U', where N_T only depends on T and can be computed.

More precisely, we define a function $\rho : \Sigma^* \to \Sigma^*$ and define U' such that on any input w, it simulates U on input $\rho(w)$. The function ρ iterates some well-chosen subwords of w to blow up the delay between the outputs of the runs of T on $\rho(w)$. On input $\rho(w)$, any

[4] For simplicity reasons, we put real-timeness in the definition, but it is known to be wlog.

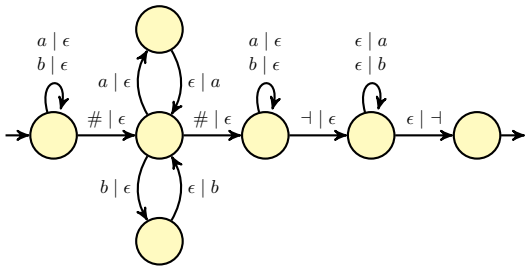


Figure 2 A deterministic transducer with endmarker for $\mathcal{R}_1$ from Ex. 17.

seq-k-uniformiser of T, and U in particular, is forced to make choices between possible outputs of T on $\rho(w)$, in order to decrease the delay. The main idea is that if, by making some good choice of output, U is able to react to a threat of exceeding delay k on $\rho(w)$, then by doing the same choice on w, U' can also react to a threat of exceeding delay N_T.

We identify several key properties that ρ must satisfy, in order to be able to construct U'. For instance, we require that $\rho(w)$ is a prefix of $\rho(wa)$ for all $w \in \Sigma^*$, $a \in \Sigma$, but also some property relating the delays between U and T on input w and on input $\rho(w)$.

The challenging part of the proof is to prove the existence of N_T and ρ. It is based on a study of the structural properties of the transition monoid of finitely ambiguous transducers (a monoid that captures the state behaviour of automata and transducers), and the effect of its elements on the delays. In particular, subwords of w that are iterated to define $\rho(w)$ correspond to idempotent elements in the transition monoid of T, and the bound N_T is obtained by an application of Ramsey's theorem. ◀

Since by Theorem 7 every finite-valued transducer is effectively equivalent to a finitely ambiguous transducer, the sequential uniformisation problem for finite-valued transducers reduces to sequential N-uniformisation, for computable integers N. Hence by Theorem 7 and the fact that any transducer can be trimmed in polynomial time, we get decidability of sequential uniformisation of finite-valued transducers, one of the main results of this paper.

▶ **Theorem 16.** *The sequential uniformisation problem for finite-valued transducers is decidable.*

5 Deterministic Rational Transductions

In this section we consider another subclass of rational transductions, namely the deterministic rational transductions, denoted by DRat. This class is defined in terms of specific deterministic transducers and some problems that are undecidable for general rational transductions are decidable in the case of DRat. For example, the equivalence problem is decidable [3] (while inclusion is easily seen to be undecidable [14]), and whether a given relation in DRat is recognisable [7] is also decidable. We obtain here another decidability result, namely that the sequential uniformisation problem is decidable for deterministic transducers.

For the definition of deterministic rational transducers, we work with endmarkers (this is the common way of doing it, see also [29]). The determinism includes the standard definition of unique successor states for each symbol and additionally a deterministic choice between input and output. This is enforced by a partition of the state space into states processing input symbols and states processing output symbols.

▶ **Example 17.** The transduction $\mathcal{R}_1 = \{(u\#v\#w, vx) \mid u, v, w, x \in \{a, b\}^*\}$ is in DRat since it is recognised by the deterministic transducer depicted in Fig. 2 over the alphabet $\{a, b, \#\}$ with endmarker $\dashv$. Note that for each state the outgoing transitions either all have ϵ as output component, or all have ϵ as input component. In the formal definition, this is captured by partitioning the states into input and output states.

Let Σ be an alphabet and $\dashv$ a fresh symbol used as endmarker. We let $\Sigma_\dashv := \Sigma \cup \{\dashv\}$. A deterministic transducer over the alphabet Σ with endmarker $\dashv$ is of the form $T = (Q^{\mathtt{i}}, Q^{\mathtt{o}}, F, q_0, \delta)$ with a set $Q^{\mathtt{i}}$ of input states, a set $Q^{\mathtt{o}}$ of output states (we write Q for the union of these two sets), a unique initial state q_0, a transition function $\delta : Q \times \Sigma_\dashv \to Q$, and a set $F \subseteq Q$ of accepting states. In the presence of endmarkers, the final output function is not required anymore.

For defining the semantics of such a deterministic transducer, one can transform it into a standard transducer. However, this transformation needs to take care of the endmarker only being allowed at the end of the word, which is not enforced in the definition of deterministic transducers. To avoid this, we rather define the semantics by extending the transition function to pairs of words (input and output word). For $(u, v) \in \Sigma_\dashv^* \times \Sigma_\dashv^*$ and $q \in Q$, we define $\delta^* : Q \times \Sigma_\dashv^* \times \Sigma_\dashv^* \to Q \times \Sigma_\dashv^* \times \Sigma_\dashv^*$ inductively as follows:

- If $q \in Q^{\mathtt{i}}$, then $\delta^*(q, \epsilon, v) = (q, \epsilon, v)$ and $\delta^*(q, au, v) = \delta^*(\delta(q, a), u, v)$.
- If $q \in Q^{\mathtt{o}}$, then $\delta^*(q, u, \epsilon) = (q, u, \epsilon)$ and $\delta^*(q, u, av) = \delta^*(\delta(q, a), u, v)$.

So δ^* applies δ to the next input letter from states in $Q^{\mathtt{i}}$ and to the next output letter from states in $Q^{\mathtt{o}}$ as long as possible. The transduction $\mathcal{R}_T$ defined by T is

$$\mathcal{R}_T = \{(u, v) \in \Sigma^* \times \Sigma^* \mid \delta^*(q_0, u \dashv, v \dashv) = (q, \epsilon, \epsilon) \text{ with } q \in F\}.$$

Recall from Section 4 that k-delay inclusion and equivalence are complete for finite-valued transducers, as stated in Theorem 13. We note that this is not the case for DRat.

▶ Remark. There are deterministic transducers T_1 and T_2 such that $T_1 \equiv T_2$ but there is no k such that $T_1 \equiv_k T_2$.

Proof. Consider the complete relation $\Sigma^* \times \Sigma^*$, and let T_1 be the deterministic transducer that first reads all input symbols (up to the endmarker $\dashv$), and then reads all output symbols. Let T_2 be the deterministic transducer that first reads all output symbols and then the input symbols. Obviously, $\mathcal{R}_{T_1} = \mathcal{R}_{T_2} = \Sigma^* \times \Sigma^*$. However, the lag for the two runs of T_1, T_2 on a pair (u, v) is $|v|$ and thus not bounded. ◀

Our main result for DRat is the following, which extends the corresponding result for automatic relations from [6].

▶ **Theorem 18.** *The sequential uniformisation problem for deterministic transducers is decidable.*

The proof uses the game-theoretic approach, building a game between players Input and Output. A winning strategy for player Output then corresponds to a sequential uniformiser. The moves of the game simulate the deterministic transducer T on the pairs of input and output word played by the two players in order to check whether the output indeed matches the input. However, Output might need to delay the moves to gain some lookahead on the input for making the next decisions. The main challenge in the proof is to find a way to keep the lookahead information bounded without losing too much information. It is not sufficient to simply store words of bounded length as lookahead. The information in the lookahead

rather provides information on the behaviour that the lookahead word induces in T. Player Output can delete parts of this information to reduce the size of the lookahead.

The sequential uniformiser that is constructed from the game in the decidability proof can be shown to have bounded delay. So we obtain the following result, showing that sequential k-uniformisation is complete for deterministic transducers.

▶ **Theorem 19.** *Any sequentially uniformisable deterministic transducer is sequentially k-uniformisable for some $k \in \mathbb{N}$ that can be computed from the given transducer.*

6 Conclusion

We have introduced the notion of resynchronisers, which are transformations for synchronisations of transductions. The decision problems of inclusion, equivalence, and sequential uniformisation, which are undecidable for general rational transductions, become decidable modulo rational resynchronisers. Furthermore, we have shown that it is sufficient to consider k-delay resynchronisers in the context of these decision problems. We have analysed two subclasses of transducers, finite-valued transducers and deterministic transducers. For both classes, sequential uniformisation is decidable, and the existence of a sequential uniformiser implies the existence of a sequential k-uniformiser. Additionally, for finite-valued transducers k-inclusion is shown to be complete. One interesting open question is the problem of deciding for a transducer whether it admits a sequential k-uniformiser for some k.

References

1 Marie-Pierre Béal, Olivier Carton, Christophe Prieur, and Jacques Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theoretical Computer Science*, 292(1):45–63, 2003.

2 Jean Berstel. *Transductions and Context-free Languages*. Teubner-Verlag, December 2009. URL: `http://www-igm.univ-mlv.fr/~berstel/`.

3 Malcolm Bird. The equivalence problem for deterministic two-tape automata. *J. Comput. Syst. Sci.*, 7(2):218–236, 1973. `doi:10.1016/S0022-0000(73)80045-5`.

4 Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive(1) designs. *J. Comput. Syst. Sci.*, 78(3):911–938, 2012. `doi:10.1016/j.jcss.2011.08.007`.

5 Mikolaj Bojanczyk. Transducers with origin information. *ICALP*, abs/1309.6124, 2013.

6 Arnaud Carayol and Christof Löding. Uniformization in Automata Theory. In *Proceedings of the 14th Congress of Logic, Methodology and Philosophy of Science Nancy, July 19-26, 2011*, pages 153–178. London: College Publications, 2014.

7 Olivier Carton, Christian Choffrut, and Serge Grigorieff. Decision problems among the main subfamilies of rational relations. *ITA*, 40(2):255–275, 2006. `doi:10.1051/ita:2006005`.

8 Rodrigo de Souza. On the decidability of the equivalence for k-valued transducers. In *Developments in Language Theory, 12th International Conference, DLT 2008, Kyoto, Japan, September 16-19, 2008. Proceedings*, pages 252–263, 2008.

9 Rodrigo de Souza and Nami Kobayashi. A combinatorial study of k-valued rational relations. *Journal of Automata, Languages and Combinatorics*, 13(3/4):207–231, 2008.

10 Samuel Eilenberg. *Automata, Languages, and Machines*. Academic Press, 1974.

11 Diego Figueira and Leonid Libkin. Synchronizing relations on words. *Theory Comput. Syst.*, 57(2):287–318, 2015.

12 Emmanuel Filiot, Ismaël Jecker, Christof Löding, and Sarah Winter. On equivalence and uniformisation problems for finite transducers. *CoRR*, abs/1602.08565, 2016. URL: `http://arxiv.org/abs/1602.08565`.

13 Emmanuel Filiot, Naiyong Jin, and Jean-François Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011.

14 Patrick C. Fischer and Arnold L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2(1):88–101, 1968. doi:10.1016/S0022-0000(68)80006-6.

15 Emily P. Friedman and Sheila A. Greibach. A polynomial time algorithm for deciding the equivalence problem for 2-tape deterministic finite state acceptors. *SIAM J. Comput.*, 11(1):166–183, 1982. doi:10.1137/0211013.

16 Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *LNCS*. Springer, 2002.

17 Timothy V. Griffiths. The unsolvability of the equivalence problem for lambda-free nondeterministic generalized machines. *Journal of the ACM*, 15(3):409–413, 1968.

18 Eitan M. Gurari and Oscar H. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Theory of Computing Systems*, 16(1):61–66, 1983.

19 Oscar H. Ibarra. The unsolvability of the equivalence problem for ϵ-free NGSM's with unary input (output) alphabet and applications. *SIAM Journal on Computing*, 7(4):524–532, November 1978.

20 Karel Culik II and Juhani Karhumäki. The equivalence of finite valued transducers (on HDT0L languages) is decidable. *TCS: Theoretical Computer Science*, 47, 1986.

21 J. H. Johnson. Do rational equivalence relations have regular cross-sections? In *Lecture Notes in Computer Science*, volume 194 of *LNCS*, pages 300–309. Springer, 1985.

22 Julius Richard Büchi and Lawrence H. Landweber. Solving sequential conditions finite-state strategies. *Trans. Ameri. Math. Soc.*, 138:295–311, 1969.

23 Kojiro Kobayashi. Classification of formal languages by functional binary transductions. *Information and Control*, 15(1):95–109, July 1969.

24 Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Safraless compositional synthesis. In *Computer Aided Verification, 18th International Conference, CAV 2006*, volume 4144 of *Lecture Notes in Computer Science*, pages 31–44. Springer, 2006.

25 Sylvain Lombardy and Jacques Sakarovitch. Sequential? *Theor. Comput. Sci.*, 356(1-2):224–244, 2006.

26 Maurice Nivat. Transductions des langages de Chomsky. *Ann. de l'Inst. Fourier*, 18:339–456, 1968. in french.

27 Maryse Pelletier and Jacques Sakarovitch. On the representation of finite deterministic 2-tape automata. *TCS: Theoretical Computer Science*, 225, 1999.

28 Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *ACM Symposium on Principles of Programming Languages (POPL)*. ACM, 1989.

29 Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

30 Jacques Sakarovitch and Rodrigo de Souza. Lexicographic decomposition of k-valued transducers. *Theory of Computing Systems*, 47(3):758–785, 2010.

31 Wolfgang Thomas. Church's problem and a tour through automata theory. In *Pillars of Computer Science, Essays Dedicated to Boris (Boaz) Trakhtenbrot on the Occasion of His 85th Birthday*, volume 4800 of *Lecture Notes in Computer Science*, pages 635–655. Springer, 2008.

32 Andreas Weber. A decomposition theorem for finite-valued tranducers and an application to the equivalence problem. In *13th International Symposium on Mathematical Foundations of Computer Science, MFCS 1988*, pages 552–562, 1988.

33 Andreas Weber. On the valuedness of finite transducers. *Acta Informatica*, 27(8):749–780, 1989.

34 Andreas Weber and Reinhard Klemm. Economy of description for single-valued transducers. *Information and Computation*, 118(2):327–340, 1995.

The Bridge Between Regular Cost Functions and Omega-Regular Languages*

Thomas Colcombet[1] and Nathanaël Fijalkow[2]

1 **CNRS, IRIF, Université Paris Diderot, Paris, France**
`thomas.colcombet@irif.univ-paris-diderot.fr`
2 **University of Oxford, Oxford, United Kingdom**
`nathanael.fijalkow@cs.ox.ac.uk`

Abstract

In this paper, we exhibit a one-to-one correspondence between ω-regular languages and a subclass of regular cost functions over finite words, called ω-regular like cost functions. This bridge between the two models allows one to readily import classical results such as the last appearance record or the McNaughton-Safra constructions to the realm of regular cost functions. In combination with game theoretic techniques, this also yields a simple description of an optimal procedure of history-determinisation for cost automata, a central result in the theory of regular cost functions.

1998 ACM Subject Classification F 1.1 Models of Computations

Keywords and phrases Theory of Regular Cost Functions, Automata with Counters, Cost-automata, Quantitative Extensions of Automata, Determinisation of Automata

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.126

1 Introduction

The theory of regular cost functions [4] aims at offering a uniform framework dealing with boundedness questions in automata theory. It provides a toolbox of concepts and results for solving questions involving resource constraints, such as the star height problem over finite words [10, 12] and finite trees [7], the finite power property [14], the boundedness of fixpoints for monadic second-order logic [2] or over guarded logic [1], or for attacking the Mostowski index problem [7]. The strength of regular cost functions is that it is a quantitative setting where many of the crucial results of regular languages generalise, including the cornerstone effective equivalence between logic, automata, algebra and expressions.

For regular languages, determinising plays a central role, as for instance for complementing automata over infinite trees, or for solving games. The situation is different for cost functions, even over finite words: it is impossible to determinise cost automata, deterministic cost automata being strictly less expressive. The notion of history-deterministic automata overcomes this shortcoming. These are non-deterministic cost automata that have the semantical property that an oracle resolves the non-determinism in an optimal way. The non-determinisability issue is resolved by establishing that cost automata can be effectively transformed into history-deterministic ones [4]. This is crucially used for instance when

* This project has received funding from the European Research Council Seventh Framework Programme (FP7/2007-2013) under grant agreement 259454 (GALE) and the European Union's Horizon 2020 research and innovation programme under grant agreement 670624 (DuaLL). The second author gratefully acknowledges the support of the EPSRC grant EP/M012298/1.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 126; pp. 126:1–126:13

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

developing the theory of regular cost functions over finite trees [7]. However, the proof of this result is so far very complicated. The original version [4] was going through algebra (stabilisation monoids), incurring a double exponential blowup. An optimal version inspired by the construction of Safra is known [5], but its description and correctness proof are extremely complicated.

One aim of the present work is to give a simple description and correctness proof of the construction from [5] which, given a cost-automaton as input, produces an equivalent history-deterministic automaton. The key advantage of the novel presentation in this work is that it uses the determinisation of ω-regular languages as a *black box*. In particular, it does not depend at all on the details of the (relatively complicated) Safra construction. This also makes both the construction and the proof much simpler. Further, it is optimal, meaning that it yields an automaton of exponential size, matching known lower bound for the case of ω-regular languages [8].

In order to obtain this completely new presentation, we describe a one-to-one correspondence between the theory of ω-regular languages and a subclass of regular cost functions, called the ω-regular like cost functions. This correspondence allows us to readily import from ω-regular languages constructions such as the last appearance record or determinisation results to regular cost functions. In other words, ω-regular like cost functions are determinisable.

In a second step, combining game theoretic techniques with an idea of Bojańczyk [3], we obtain a simple, direct and optimal construction of history-deterministic cost automata.

Structure of the document

We define the class of ω-regular like cost functions in Section 3, and study its properties. We give in Section 4 the history-determinisation procedure, relying on the results about ω-regular like cost functions combined with game theoretic techniques.

2 Definitions

Let $\mathtt{A}$ and $\mathtt{B}$ be *alphabets*. An *initial automaton structure* is denoted $\mathcal{A} = (Q, \mathtt{A}, \mathtt{B}, I, \Delta)$, where $\mathtt{A}$ is the *input alphabet*, $\mathtt{B}$ is the *output alphabet*, Q is a finite *set of states*, $I \subseteq Q$ is the set of *initial states* and $\Delta \subseteq Q \times \mathtt{A} \times \mathtt{B} \times Q$ is the *transition relation*. An element $(p, a, b, q) \in \Delta$ is called a *transition*. An *automaton structure* $\mathcal{A} = (Q, \mathtt{A}, \mathtt{B}, I, \Delta, F)$ is an initial automaton structure enriched with a set $F \subseteq Q$ of accepting states.

A *run* (which can be finite or infinite) is a sequence of transitions of the form

$$(p_0, a_1, b_1, p_1)(p_1, a_2, b_2, p_2) \cdots$$

such that p_0 is initial. We denote by $\rho|_A$ its projection to the alphabet $\mathtt{A}$, and by $\rho|_B$ its projection to the alphabet $\mathtt{B}$. For w a finite or infinite word over $\mathtt{A}$, we say that a run ρ is a run of w if $\rho|_A = w$ and a prefix run of w if $\rho|_A$ is a prefix of w. When dealing with an automaton structure, we further require that a run of a finite word ends in an accepting state.

ω-automata

An *ω-automaton* is denoted $\mathcal{A} = (Q, \mathtt{A}, \mathtt{B}, I, \Delta, W)$, where $(Q, \mathtt{A}, \mathtt{B}, I, \Delta)$ is an initial automaton structure, and $W \subseteq \mathtt{B}^\omega$ is called the *accepting condition*. The ω-language recognised by the ω-automaton is the set

$$\{w \in \mathtt{A}^\omega \mid \text{there exists a run } \rho \text{ over } w \text{ such that } \rho|_B \in W\}.$$

We define some of the classical ω-accepting conditions.

$$\begin{aligned}
\texttt{Büchi} &= \{w \in \{0,1\}^\omega \mid w \text{ contains infinitely many } 0's\} \\
\texttt{coBüchi} &= \{w \in \{1,2\}^\omega \mid w \text{ contains finitely many } 1's\} \\
\texttt{Rabin}_1 &= \left\{ w \in \{I,R,\epsilon\}^\omega \;\middle|\; \begin{array}{l} w \text{ contains infinitely many } I's \\ \text{and finitely many } R's \end{array} \right\} \\
\texttt{Rabin}_k &= \left\{ w \in \left(\{I,R,\epsilon\}^k\right)^\omega \;\middle|\; \begin{array}{l} \text{for some } \ell \in \{1,\dots,k\}, \\ w \text{ contains infinitely many } I'_\ell s \\ \text{and finitely many } R'_\ell s \end{array} \right\} \\
\texttt{Parity}_k &= \left\{ w \in \{1,\dots,k\}^\omega \;\middle|\; \begin{array}{l} \text{the smallest colour appearing} \\ \text{infinitely often in } w \text{ is even} \end{array} \right\}
\end{aligned}$$

A *Rabin automaton* is an ω-automaton with a Rabin condition, and similarly for the other conditions. It is known that Büchi, parity and Rabin automata recognise the same ω-languages, that are called the *ω-regular languages*.

Regular cost functions

We consider functions from $\mathtt{A}^*$ to $\mathbb{N} \cup \{\infty\}$. Let f be such a function, and $X \subseteq \mathtt{A}^*$, we say that $f|_X$ is bounded if there exists $n \in \mathbb{N}$ such that $f(u) \leq n$ for all $u \in X$.

Let f, g be two such functions, then g *dominates* f, denoted $f \preccurlyeq g$, if for all $X \subseteq \mathtt{A}^*$, if $g|_X$ is bounded then $f|_X$ is bounded. We say that f and g are equivalent, denoted $f \approx g$, if $f \preccurlyeq g$ and $g \preccurlyeq f$. The following lemma is central, see [6] for more considerations on this equivalence relation.

▶ **Lemma 1.** *$f \preccurlyeq g$ if, and only if, $f \leq \alpha \circ g$ for some function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\lim \alpha = \infty$, extended with $\alpha(\infty) = \infty$.*

A *cost function* is an equivalence class for the relation $\approx$.

Many equivalent formalisms can be used in order to define regular cost functions; this paper studies automata.

▶ **Definition 2.** A *min-cost-automaton* is denoted $\mathcal{A} = (Q, \mathtt{A}, \mathtt{B}, I, \Delta, F, f)$, given by an automaton structure $(Q, \mathtt{A}, \mathtt{B}, I, \Delta, F)$ together with an *accepting map* $f : \mathtt{B}^* \to \mathbb{N} \cup \{\infty\}$. It recognises the cost function induced by the map

$$\begin{aligned}
[\![\mathcal{A}]\!]_{\min} : \mathtt{A}^* &\to \mathbb{N} \cup \{\infty\} \\
w &\mapsto \inf \{f(\rho|_B) \mid \rho \text{ run over } w\} \ .
\end{aligned}$$

A *max-cost-automaton* is defined similarly, recognising the cost function induced by the map

$$\begin{aligned}
[\![\mathcal{A}]\!]_{\max} : \mathtt{A}^* &\to \mathbb{N} \cup \{\infty\} \\
w &\mapsto \sup \{f(\rho|_B) \mid \rho \text{ run over } w\} \ .
\end{aligned}$$

We define some of the classical accepting maps for regular cost functions.

We first define the $\texttt{cost}_\mathtt{B}$ map for one counter. The value of the counter is initialised by 0. The letter `i` is an increment, it adds 1 to the value of the counter, the letter `r` is a

reset, it resets the value of the counter to 0, and the letter ϵ does nothing. Formally, the $\mathtt{cost}_{\mathtt{B}}$ map for one counter is defined by

$$\begin{aligned}\mathtt{cost}_{\mathtt{B}} : \{\mathtt{i},\mathtt{r},\epsilon\}^* &\to \mathbb{N}\cup\{\infty\}\\ w &\mapsto \max\{n\in\mathbb{N} \mid w\in\{\mathtt{i},\mathtt{r},\epsilon\}^*\,(\epsilon^*\mathtt{i})^n\,\{\mathtt{i},\mathtt{r},\epsilon\}^*\}\,.\end{aligned}$$

The restrictions over $\{\epsilon,\mathtt{i}\}^*$ and over $\{\mathtt{r},\mathtt{i}\}^*$ are called distance and desert, respectively denoted $\mathtt{dist}_{\mathtt{B}}$ and $\mathtt{desert}_{\mathtt{B}}$.

The $\mathtt{cost}_{\mathtt{B}}$ map for k counters is defined similarly as for one counter, over the alphabet $\{\epsilon,\mathtt{i},\mathtt{r}\}^k$, by taking the maximum over all counters.

The $\mathtt{cost}_{\mathtt{hB}}$ map for k hierarchical counters is the restriction of $\mathtt{cost}_{\mathtt{B}}$ over the alphabet $\{I_1,R_1,\ldots,I_k,R_k\}$, where I_ℓ increments the ℓ^{th} counter and resets all counters of smaller index, and R_ℓ resets all counters of index smaller than or equal to ℓ.

A *B-automaton* is a *min-cost-automaton* equipped with a $\mathtt{cost}_{\mathtt{B}}$ map. Similarly, a *hB-automaton* is equipped with a $\mathtt{cost}_{\mathtt{hB}}$ map. The class of cost functions recognised by B-automata (or equivalently, hB-automata) is called regular cost functions.

We will make use of the following special case of max-cost-automata.

▶ **Definition 3.** A *prefix-max-cost-automaton* is denoted $\mathcal{A}=(Q,\mathtt{A},\mathtt{B},I,\Delta,f)$, given by an initial automaton structure $(Q,\mathtt{A},\mathtt{B},I,\Delta)$ together with an *accepting map* $f:\mathtt{B}^*\to\mathbb{N}\cup\{\infty\}$. It recognises the cost function induced by the map

$$\begin{aligned}[\![\mathcal{A}]\!]_{\text{pmax}} : \mathtt{A}^* &\to \mathbb{N}\cup\{\infty\}\\ w &\mapsto \sup\{f(\rho|_B) \mid \rho \text{ prefix run over } w\}\ .\end{aligned}$$

3 Omega Regular like Cost Functions

In this section we introduce the subclass of regular cost functions called ω-regular like cost functions, that we show are in one-to-one correspondence with ω-regular languages.

In Subsection 3.1 we define an operator defining the class and fleshing out the correspondence. We then explain how to construct ω-regular like cost functions with different models: in Subsection 3.2 using automata, and in Subsection 3.3 using algebra.

This strong correspondence allows us to transfer results from ω-regular languages to ω-regular like cost functions; in Subsection 3.4 we show how to transfer the latest appearence record and the Safra constructions.

Finally, we show the interplay between ω-regular like cost functions and games in Subsection 3.5.

3.1 Bijection with Omega-Regular Languages

The following is the main definition of this paper.

▶ **Definition 4.** Given a language L over infinite words, $L^{\mathtt{ol}}$ is defined by

$$\begin{aligned}L^{\mathtt{ol}} : \mathtt{A}^* &\to \mathbb{N}\cup\{\infty\}\\ w &\mapsto \sup\{n \mid w = uv_1\cdots v_nu',\ v_1,\ldots,v_n\neq\varepsilon,\ u\cdot\{v_1,\ldots,v_n\}^\omega\subseteq L\}\,.\end{aligned}$$

A cost function is *ω-regular like* if it contains a map $L^{\mathtt{ol}}$ for some ω-regular language L.

Note that we will mostly be interested in using the definition of $\cdot^{\mathtt{ol}}$ with ω-regular languages.

▶ **Example 5.**

- $\texttt{Büchi}^{\texttt{ol}} = \texttt{dist}_{\texttt{B}}$: it is the function counting the number of 1's, *i.e.* the distance map where 1 is ϵ and 0 is $\texttt{i}$.
- $\texttt{coBüchi}^{\texttt{ol}} = \texttt{desert}_{\texttt{B}}$: it is the function counting the size of the largest block of 2's, *i.e.* the desert map where 1 is $\texttt{r}$ and 2 is $\texttt{i}$.
- $\texttt{Rabin}_1{}^{\texttt{ol}} = \texttt{cost}_{\texttt{B}}$: it is the function counting the number of I's in a block containing no R's, *i.e.* the $\texttt{cost}_{\texttt{B}}$ map for one counter where I is $\texttt{i}$ and R is $\texttt{r}$.
- $\texttt{Rabin}_k{}^{\texttt{ol}} \approx \texttt{cost}_{\texttt{B}}$: it is the $\texttt{cost}_{\texttt{B}}$ map for k counters where I_ℓ is increment for the ℓ^{th} counter and R_ℓ is reset for the ℓ^{th} counter. Note that here the functions are not equal, one can see that $\texttt{cost}_{\texttt{B}} \leq \texttt{Rabin}_k{}^{\texttt{ol}} \leq k \cdot \texttt{cost}_{\texttt{B}}$.
- $\texttt{Parity}_{2k}{}^{\texttt{ol}} \approx \texttt{cost}_{\texttt{hB}}$: it is the $\texttt{cost}_{\texttt{hB}}$ map for k counters, where I_ℓ is the colour $2(k-\ell)$ and R_ℓ is $2(k-\ell)-1$.

The following lemma is central, it shows the interplay between the above definition and ultimately periodic words.

▶ **Lemma 6.** *Let L be a language over infinite words, and u, v two finite words with v non-empty. The following statements are equivalent:*

1. $uv^\omega \in L$,
2. $(L^{\texttt{ol}}(uv^n))_{n\in\mathbb{N}}$ *tends to infinity.*

Note that this lemma does not make any assumption on the regularity of L; in the rest of the paper, we shall always look at $L^{\texttt{ol}}$ for L an ω-regular language.

Proof. One direction is clear: if $uv^\omega \in L$, then $(L^{\texttt{ol}}(uv^n))_{n\in\mathbb{N}}$ tends to infinity.

We prove the converse implication. Assume that $(L^{\texttt{ol}}(uv^n))_{n\in\mathbb{N}}$ tends to infinity, and let n be larger than $|uv|$. There exists k such that uv^k can be factorised $u'v_1\cdots v_n u''$ such that $u' \cdot \{v_1, \ldots, v_n\}^\omega \subseteq L$.

Consider the lengths $|u'v_1\cdots v_\ell|$ for $\ell \in \{|u|, \ldots, n\}$: two of them have the same value modulo $|v|$, denote the corresponding words $u'v_1\cdots v_i$ and $u'v_1\cdots v_j$, with $i < j$. Note that since $\ell \geq |u|$ and $v_1, \ldots, v_\ell$ are not empty, the word u is a strict prefix of $u'v_1\cdots v_i$. It follows that we have $u'v_1\cdots v_i = uv^p x$ for some p and $v = xy$, and $v_{i+1}\cdots v_j = yv^q x$ for some q.

Consider the infinite word

$$u'v_1\cdots v_i(v_{i+1}\cdots v_j)^\omega = uv^p x(yv^q x)^\omega \ ,$$

by assumption it belongs to L. Thanks to the equality $s(ts)^\omega = (st)^\omega$, the word above is equal to $uv^p(xyv^q)^\omega = uv^p(v^{q+1})^\omega = uv^\omega$. Thus, $uv^\omega \in L$. ◀

▶ **Theorem 7.** *The map $\cdot^{\texttt{ol}}$ is a bijection between ω-regular languages and ω-regular like cost functions.*

In particular, two ω-regular like cost functions $L^{\texttt{ol}}, L'^{\texttt{ol}}$ are equal if, and only if, $L = L'$.

Proof. The map is surjective by definition of ω-regular like cost functions.

We show that it is injective: consider L, L' two ω-regular languages such that $L^{\texttt{ol}} \approx L'^{\texttt{ol}}$. It follows from Lemma 6 that L and L' coincide on ultimately periodic words; being ω-regular, this implies that they are equal. ◀

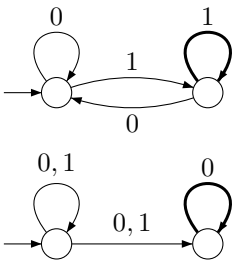


Figure 1 The Büchi automaton for Example 9.

3.2 Automata Constructions

The following theorem shows how to construct automata recognising ω-regular like cost functions. The construction is very simple, as it amounts to consider an ω-automaton and to see it as a prefix-max-cost-automaton, without any further changes. The correctness proof however is bit more involved.

▶ **Theorem 8.** *Let W be an ω-regular condition.*

Consider a W-automaton $\mathcal{A}$, and denote by L the language it recognises. The prefix-max-cost-automaton induced by $\mathcal{A}$ with the map $W^{\mathtt{ol}}$ recognises the cost function $L^{\mathtt{ol}}$.

Before proving this theorem, we give an example.

▶ **Example 9.** Consider the Büchi automaton represented in Figure 1. The alphabet is $\mathtt{A} = \{0, 1\}$, the Büchi transitions are represented by a boldface loop. The top part checks whether the word contains infinitely many 1's, and the bottom part checks whether the word contains finitely many 1's. It follows that this automaton recognises all ω-words, *i.e.* $L = \mathtt{A}^\omega$, so

$$\begin{aligned} L^{\mathtt{ol}} : \mathtt{A}^* &\to \mathbb{N} \cup \{\infty\} \\ w &\mapsto \text{length of } w \end{aligned}$$

The induced prefix-max-cost-automaton recognises the following cost function:

$$\begin{aligned} [\![\mathcal{A}]\!]_{\text{pmax}} : \mathtt{A}^* &\to \mathbb{N} \cup \{\infty\} \\ w &\mapsto \max\{\text{number of } 1's \text{ in } w, \text{size of the largest block of } 0's \text{ in } w\} . \end{aligned}$$

These two functions are indeed equivalent: $[\![\mathcal{A}]\!]_{\text{pmax}} \leq L^{\mathtt{ol}} \leq [\![\mathcal{A}]\!]^2_{\text{pmax}}$.

In the proof of Theorem 8, we will make use of Simon's theorem [13]. We state here the corollary that we use. Recall that a semigroup is a set equipped with an associative binary product, and that an idempotent in a semigroup is an element e such that $e \cdot e = e$.

For every morphism $\varphi : \mathtt{A}^+ \to M$ where M is a finite semigroup, there exists a function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\lim \alpha = \infty$ and for all words w of length n, there exists a factorisation $w = uv_1 \cdots v_{\alpha(n)}u'$ where the words $v_1, \ldots, v_{\alpha(n)}$ are non-empty and such that

$$\varphi(v_1) = \varphi(v_2) = \cdots = \varphi(v_{\alpha(n)})$$

is an idempotent.

Proof. For the sake of simplicity, we will assume that $\mathcal{A}$ is a parity automaton, *i.e.* W is the parity language. The proof generalises to the case of an ω-regular language W by considering a deterministic parity automaton recognising W.

We denote $\mathcal{A}^{\mathtt{ol}}$ the prefix-max-cost-automaton induced by $\mathcal{A}$ with the accepting map $\mathtt{Parity}^{\mathtt{ol}}$. By definition:

- $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}(w)$ is defined by

$$\sup\left\{\mathtt{Parity}^{\mathtt{ol}}(\rho|_B) \mid \rho \text{ prefix run over } w\right\}.$$

- $L^{\mathtt{ol}}(w)$ is defined by

$$\sup\left\{n \mid w = uv_1\cdots v_nu',\ v_1,\dots,v_n \neq \varepsilon,\ u\cdot\{v_1,\dots,v_n\}^\omega \subseteq L\right\}.$$

We will apply Simon's theorem twice, once for each direction. To this end, we construct a morphism $\varphi : \mathtt{A} \to M$, where M is the transition semigroup of $\mathcal{A}$. A transition profile is a tuple (p,c,q) where p and q are states and c is a colour. The product of transition profiles is (partially) defined by

$$(p,c,q)\cdot(r,c',s) = \begin{cases} (p,\min(c,c'),s) \text{ if } q=r \\ \text{undefined otherwise.}\end{cases}$$

An element of M is a set of transition profiles. The product is inherited by the product for transition profiles. The morphism φ associates to a letter a the set of transitions over the letter a in the automaton $\mathcal{A}$.

Assume $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}(w) \geq n$: there exists a prefix run ρ over w such that $\mathtt{Parity}^{\mathtt{ol}}(\rho|_B) \geq n$. It follows that ρ factorises $\rho\rho_1\cdots\rho_n\rho'$, where in each ρ_i the smallest colour appearing is even. We apply Simon's theorem to the word $\rho_1\cdots\rho_n$, seen as a word of length n, *i.e.* where we interpret each ρ_i as a single letter. Denote $m=\alpha(n)$. There exists a factorisation which we denote $\widetilde{\tau}\widetilde{\tau_1}\cdots\widetilde{\tau_m}\widetilde{\tau'}$ such that $\varphi(\widetilde{\tau_1}) = \cdots = \varphi(\widetilde{\tau_m})$ is idempotent, denoted S. This implies the existence of (q,c,q) in S, where c is even. Denote $w = uv_1\cdots v_mu'$ the factorisation of w it induces. Observe that $u\cdot\{v_1,\dots,v_m\}^\omega \subseteq L$, as for each such word one can construct a ultimately periodic run ρ in $\mathcal{A}$ whose smallest colour appearing infinitely often is c, hence such that $\rho|_B \in \mathtt{Parity}$. So $W^{\mathtt{ol}}(\rho) \geq \alpha(n)$.

It follows that $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}} \preccurlyeq L^{\mathtt{ol}}$.

Conversely, assume $L^{\mathtt{ol}}(w) \geq n$: there exists a factorisation of w in $uv_1\cdots v_nu'$ as in the definition of $L^{\mathtt{ol}}(w)$. We apply Simon's theorem to the word $v_1\cdots v_n$, seen as a word of length n, *i.e.* where we interpret each v_i as a single letter. Denote $m=\alpha(n)$. There exists a factorisation which we denote $\widetilde{u}\widetilde{v_1}\cdots\widetilde{v_m}\widetilde{u'}$ such that $\varphi(\widetilde{v_1}) = \cdots = \varphi(\widetilde{v_m})$ is idempotent, denoted S. Note that each $\widetilde{v_\ell}$ and $\widetilde{u}$ is an infix $v_i\cdots v_j$; denote $\widetilde{w}$ the infix corresponding to $\widetilde{v_1}\cdots\widetilde{v_m}$. The element $\varphi(\widetilde{w})$ is idempotent equal to S. Since $u\cdot\{v_1,\dots,v_n\}^\omega \subseteq L$, in particular $u\widetilde{u}\cdot\widetilde{w}^\omega \in L$. Because $\mathcal{A}$ recognises L, there exists an accepting run of $u\widetilde{u}\cdot\widetilde{w}^\omega$. Now, $\varphi(\widetilde{w})$ being idempotent, this implies that there exist:

- a transition profile in $\varphi(u\cdot\widetilde{u})$ of the form $(p,\_,q)$ where p is initial, and
- a transition profile in $\varphi(\widetilde{w})$ of the form (q,c,q) where c is even.

Recall that each $\varphi(\widetilde{v_\ell})$ is equal to $S=\varphi(\widetilde{w})$, so it contains (q,c,q). Thus, we obtain a run $\rho = \rho_{u\widetilde{u}}\rho_{\widetilde{v_1}}\cdots\rho_{\widetilde{v_m}}$ over $u\widetilde{u}\widetilde{v_1}\cdots\widetilde{v_m}$ such that $\rho_{u\widetilde{u}}\left\{\rho_{\widetilde{v_1}},\dots,\rho_{\widetilde{v_m}}\right\}^\omega \subseteq W$. This implies that $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}(w) \geq \alpha(n)$.

It follows that $L^{\mathtt{ol}} \preccurlyeq [\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}$.

We conclude that $L^{\mathtt{ol}}$ and $[\![\mathcal{A}^{\mathtt{ol}}]\!]_{\mathrm{pmax}}$ are equivalent. ◀

Recall that if W is the Büchi language, then $W^{\circ 1}$ is the distance map. Similarly, the Rabin condition induces the $\mathtt{cost}_{\mathtt{B}}$ map and the parity condition the $\mathtt{cost}_{\mathtt{hB}}$ map.

In particular, Theorem 8 implies that if L is recognised by a Büchi automaton (resp. Rabin automaton, parity automaton), then $L^{\circ 1}$ is recognised by a prefix-max-cost-automaton with the distance map (resp. $\mathtt{cost}_{\mathtt{B}}$ map, $\mathtt{cost}_{\mathtt{hB}}$ map).

3.3 Syntactical Constructions

The above subsection shows how to construct ω-regular like cost functions using automata.

3.4 Transferring Results

We show in this subsection how to use the above correspondence to transfer two automata theoretic constructions.

The first construction is the latest appearance record construction, which allows to transform a Rabin condition into a parity condition, as stated in the following theorem.

▶ **Theorem 10.** *For every k, there exists a deterministic parity automaton with $k!$ states and k colours recognising the language $\mathtt{Rabin}_k$.*

This yields the following corollary.

▶ **Corollary 11.** *For every k, there exists a hierarchical B-automaton (hB-automaton) with $k!$ states and k counters recognising the cost function $\mathtt{cost}_{\mathtt{B}}$.*

Consequently, for every regular cost function, one can effectively construct an hB-automaton recognising it.

The first part is obtained by applying Theorem 8 to the automaton constructed by Theorem 10. For the second part, it amounts to compose the B-automaton with the automaton constructed by the first item to obtain an hB-automaton.

The second construction is the determinisation of Büchi automata.

▶ **Corollary 12.** *For every ω-regular like cost function, one can effectively construct a deterministic B-automaton recognising it.*

Proof. Consider an ω-regular language L given by a non-deterministic Büchi automaton, inducing the ω-regular like cost function $L^{\circ 1}$.

The McNaughton-Safra construction yields an equivalent deterministic Rabin automaton, denoted $\mathcal{A}$. Thanks to Theorem 8, this implies a prefix-max-cost-automaton equipped with the $\mathtt{Rabin}^{\circ 1}$ condition recognising $L^{\circ 1}$. Since $\mathcal{A}$ is deterministic and $\mathtt{Rabin}^{\circ 1}$ is the $\mathtt{cost}_{\mathtt{B}}$ map, $\mathcal{A}$ is in fact a deterministic B-automaton recognising $L^{\circ 1}$. ◀

3.5 Games with Omega-Regular like Cost Functions

In this subsection, we show how to solve games with ω-regular like cost functions.

We refer to [9] for materials about games; here we only give the basic definitions.

A *game* is denoted $G = (V, \mathtt{A}, V_E, V_A, E)$, where V is a set of vertices, $\mathtt{A}$ is the output alphabet, V_E is the set of vertices controlled by the first player Eve, V_A is the set of vertices controlled by the opponent Adam with $V = V_E \uplus V_A$ and $E \subseteq V \times \mathtt{A} \times V$ is the set of edges. A game is said finite if V is finite.

A token is initially placed on a given initial vertex v_0, and the player who controls this vertex pushes the token along an edge, reaching a new vertex; the player who controls this

new vertex takes over, and this interaction goes on forever, describing an infinite path called a play. A *winning condition* is a language $L \subseteq \mathtt{A}^\omega$: a play is won by Eve if its projection on $\mathtt{A}$ belongs to L. A *strategy* for Eve is a map $\sigma : E^* V_E \to E$. A *memory structure* is denoted $\mathcal{M} = (M, m_0, \mu)$, where M is the (finite) set of memory states, $m_0 \in M$ is the initial memory state and $\mu : M \times E \to M$ is the (deterministic) update function. A *finite-memory strategy* is given by a memory structure $\mathcal{M}$ and a next-move function $\sigma : M \times V_E \to E$.

▶ **Theorem 13.** *Consider a finite game G and L an ω-regular language. The following are equivalent:*

1. *There exists n, there exists a strategy for Eve, such that for all plays, the value for $L^{\mathtt{o1}}$ is less than n,*
2. *Eve wins for the winning condition $L^{\mathtt{C}}$.*

Proof. Since L is ω-regular, it is recognised by a deterministic Rabin automaton. By considering the product of the game with this automaton, we can assume without loss of generality that $L = \mathtt{Rabin}$, so $L^{\mathtt{o1}} = \mathtt{cost}_{\mathtt{B}}$.

The top to bottom direction is clear: indeed, for a play π, if $\mathtt{cost}_{\mathtt{B}}(\pi) \leq n$, then $\pi \in \mathtt{Rabin}^{\mathtt{C}}$.

To prove the converse implication, we rely on the fact that since $L^{\mathtt{C}}$ is an ω-regular condition, Eve has a finite-memory winning strategy. By considering the product of the game with the memory structure, we observe that in each cycle, for each counter, either it is not incremented or it is both incremented and reset. It follows that this strategy ensures that the values for $\mathtt{cost}_{\mathtt{B}}$ is bounded over all plays by twice the size of the graph times the size of the memory. ◀

We can strengthen this theorem:

▶ **Theorem 14.** *Consider a finite game G and L, L' two ω-regular languages. The following are equivalent:*

1. *For all n, there exists n', there exists a strategy for Eve, such that for all plays:*

   if the value for $L^{\mathtt{o1}}$ is less than n then the value for $L'^{\mathtt{o1}}$ is less than n',

2. *Eve wins for the condition $L \cup L'^{\mathtt{C}}$.*

Proof. Since L and L' are ω-regular, they are each recognised by a deterministic Rabin automaton. By considering the product of the game with the two automata, we can assume without loss of generality that the alphabet is $\{\epsilon, \mathtt{i}, \mathtt{r}\}^k \times \{\epsilon, \mathtt{i}, \mathtt{r}\}^{k'}$ with $L = \mathtt{Rabin}^1$ and $L' = \mathtt{Rabin}^2$. Thus $L^{\mathtt{o1}} = \mathtt{cost}^1_{\mathtt{B}}$ and $L'^{\mathtt{o1}} = \mathtt{cost}^2_{\mathtt{B}}$.

Assume that Eve wins for the condition $L \cup L'^{\mathtt{C}}$: since it is ω-regular, Eve has a finite-memory winning strategy. By considering the product of the game with the memory structure, we observe that for each cycle,

if for each counter in $\{\epsilon, \mathtt{i}, \mathtt{r}\}^k$, it is either reset or not incremented,
then for each counter in $\{\epsilon, \mathtt{i}, \mathtt{r}\}^{k'}$, it is either reset or not incremented.

Let n be twice the size of the graph times the size of the memory. It follows that this strategy ensures that for all plays, if the value for $\mathtt{cost}^1_{\mathtt{B}}$ is less than n then the value for $\mathtt{cost}^2_{\mathtt{B}}$ is less than n.

To prove the converse, we proceed by contrapositive. Assume that Eve does not win for the condition $L \cup L'^{\mathtt{C}}$, since the game is determined this implies that Adam wins, and again because the winning condition is ω-regular Adam has a finite-memory winning strategy. The same reasoning as before concludes that each cycle satisfies the negation of the above

property, which implies for the same value of n that this strategy ensures the following: for all n', there exists a play such that the value for $\mathtt{cost}^1_{\mathtt{B}}$ is less than n and the value for $\mathtt{cost}^2_{\mathtt{B}}$ is greater than n'. ◀

4 History-Determinisation of Cost Automata

In this section we give a simple and direct procedure for history-determinisation of B-automata: given a B-automaton, construct an equivalent history-deterministic B-automaton. Note that for the sake of simplicity we consider here hierarchical B-automata. Our contruction relies on the properties we obtained for ω-regular like cost functions in the above section together with game theoretic techniques inspired by Bojańczyk [3].

An automaton is *history-deterministic* if it is non-deterministic but its non-determinism can be resolved by a function considering only the input read so far. This notion has been introduced for studying ω-automata in [11]. We specialise it here to the case of cost functions, involving a relaxation on the values allowing for a good interplay with the definition of equivalence for cost functions.

A B-automaton $\mathcal{B}$ is *history-deterministic* if there exists a function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\lim \alpha = \infty$ and for every n, there exists a strategy $\sigma : \mathtt{A}^* \to \Delta$ such that for all words w, we have

$$[\![\mathcal{B}]\!]_{\min}(w) \leq n \implies [\![\mathcal{B}_\sigma]\!]_{\min}(w) \leq \alpha(n).$$

The automaton $\mathcal{B}_\sigma$ is infinite but deterministic, as for each situation the strategy σ chooses the transition to follow.

▶ **Theorem 15.** *For every hB-automaton, one can effectively construct an equivalent history-deterministic hB-automaton.*

Let $\mathcal{A} = (Q, \mathtt{A}, \{I_1, R_1, \ldots, I_k, R_k\}, I, \Delta, F, \mathtt{cost}_{\mathtt{hB}})$ be a hB-automaton. We first sketch the construction, which involves two automata:

- a deterministic hB-automaton $\mathcal{C}$ recognising an ω-regular like cost function denoted $L^{\circ\mathtt{l}}$,
- a history-deterministic min-cost-automaton $\mathcal{B}$ equipped with the map $L^{\circ\mathtt{l}}$.

Recall that for a word w, the value of $[\![\mathcal{A}]\!]_{\min}(w)$ is the minimum value for the $\mathtt{cost}_{\mathtt{hB}}$ map over all runs of w.

The automaton $\mathcal{B}$ simulates $\mathcal{A}$ and is in charge of guessing an optimal run, *i.e.* having minimal value for the $\mathtt{cost}_{\mathtt{hB}}$ map. However, we want $\mathcal{B}$ to be history-deterministic; to achieve this, $\mathcal{B}$ will do something easier than guessing *one* run, it will guess for each transition whether it belongs to *some* optimal run. In other words, $\mathcal{B}$ guesses a run tree (see Figure 2 for the representation of a run tree). As we shall see, thanks to the positionality of hB-games, $\mathcal{B}$ can guess a set of near optimal runs in a history-deterministic fashion. In effect, $\mathcal{B}$ inputs a word and outputs a run tree.

The automaton $\mathcal{C}$ recognises the cost function which given a run tree computes the maximum value for the $\mathtt{cost}_{\mathtt{hB}}$ map over all paths in the run tree. The crucial point is that this cost function is ω-regular like, so one can effectively construct a deterministic hB-automaton recognising it.

The composition of the automata $\mathcal{B}$ and $\mathcal{C}$ yields a history-deterministic hB-automaton equivalent to $\mathcal{A}$. The correctness of this construction relies on the following two properties:

- $[\![\mathcal{A}]\!]_{\min}$ and $[\![\mathcal{B}]\!]_{\min}$ are equivalent,
- $\mathcal{B}$ is history-deterministic.

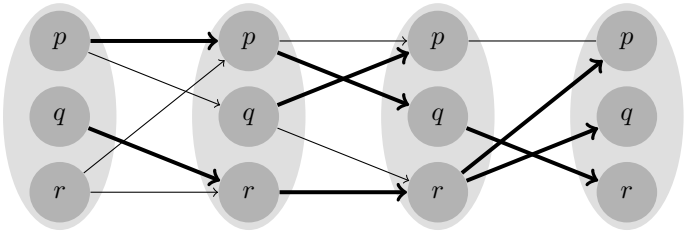


Figure 2 A run tree. Each state has at most one ingoing transition.

We proceed with the formal construction.

We define the alphabet $\mathtt{B}$, which is the output alphabet of $\mathcal{B}$ and the input alphabet of $\mathcal{C}$. A transition profile is a triple (p, act, q) where p and q are states and act is an action on the k counters, *i.e.* $act \in \{I_1, R_1, \dots, I_k, R_k\}$. An element of $\mathtt{B}$ is a set of transition profiles T such that for every state q, there exists *at most* one p such that $(p, act, q) \in T$ for some act. Equivalently, it is a partial function $T : Q \to \{I_1, R_1, \dots, I_k, R_k\} \times Q$; this backward point of view will be useful in proving that $\mathcal{B}$ is history-deterministic. A word over this alphabet is called a run tree; see Figure 2 for the representation of a run tree.

Construction of $\mathcal{C}$. The automaton $\mathcal{C}$ recognises an ω-regular like cost function; to construct it we define a parity automaton and turn it into a prefix-max-cost automaton relying on Theorem 8. Denote

$$L = \left\{ t \in \mathtt{B}^\omega \;\middle|\; \begin{array}{c} \text{there exists an infinite path in } t \text{ such that,} \\ \text{the minimal action performed infinitely often} \\ \text{for the ordering } I_1 < R_1 < \cdots < I_k < R_k \\ \text{is } R_\ell \text{ for some } \ell \end{array} \right\}.$$

The language L is recognised by a non-deterministic parity automaton of linear size which guesses the witnessing path. Formally, the parity automaton is

$$(Q, \mathtt{B}, \{I_1, R_1, \dots, I_k, R_k\}, I, \Delta_{\mathcal{C}}, \mathtt{Parity})$$

The transition relation is $\Delta_{\mathcal{C}} = \{(p, T, act, q) \mid (p, act, q) \in T\}$. The parity condition is obtained by seeing R_ℓ as the colour 2ℓ and I_ℓ as the colour $2\ell - 1$.

Theorem 8 implies that

$$L^{\mathtt{ol}} \approx \left\{ \begin{array}{lcl} \mathtt{B}^* & \to & \mathbb{N} \cup \{\infty\} \\ t & \mapsto & \max \{\mathtt{cost}_{\mathtt{hB}}(\pi) \mid \pi \text{ prefix path in } t\} \end{array} \right.$$

Following Corollary 12, we can effectively construct a deterministic hB-automaton $\mathcal{C}$ recognising $L^{\mathtt{ol}}$.

Construction of $\mathcal{B}$. The automaton $\mathcal{B}$ is a min-cost-automaton equipped with the map $L^{\mathtt{ol}}$, in charge of guessing a run tree and deterministically checking whether it contains a run.

Formally, $\mathcal{B} = (\mathcal{P}(Q), \mathtt{A}, \mathtt{B}, \mathcal{P}(I), \Delta_{\mathcal{B}}, F_{\mathcal{B}}, L^{\mathtt{ol}})$. The transition relation $\Delta_{\mathcal{B}}$ is defined by

$$\{(S, a, T, S') \mid S' \text{ is the set of states reached from } S \text{ using the transitions in } T\}.$$

The set of final states $F_{\mathcal{B}}$ is $\{S \subseteq Q \mid S \cap F \neq \emptyset\}$.

▶ **Lemma 16.** *$[\![\mathcal{A}]\!]_{\min}$ and $[\![\mathcal{B}]\!]_{\min}$ are equivalent.*

Proof. By definition

- $[\![\mathcal{A}]\!]_{\min}(w)$ is the minimum value for the $\mathtt{cost}_{\mathtt{hB}}$ map over all runs of w.
- $[\![\mathcal{B}]\!]_{\min}(w)$ is the minimum value for the $L^{\mathtt{o1}}$ map over all runs of w. By construction of $\mathcal{B}$, the runs of w are the run trees of w that contain a run of w over $\mathcal{A}$.

Thanks to the equivalence above about $L^{\mathtt{o1}}$, this implies that

$$[\![\mathcal{B}]\!]_{\min} \approx \left\{ \begin{array}{lcl} \mathtt{A}^* & \to & \mathbb{N} \cup \{\infty\} \\ w & \mapsto & \min \left\{ \mathtt{cost}_{\mathtt{hB}}(t) \;\middle|\; \begin{array}{c} t \text{ run tree of } w \text{ which} \\ \text{contains a run of } w \text{ over } \mathcal{A} \end{array} \right\}, \end{array} \right.$$

where $\mathtt{cost}_{\mathtt{hB}}(t) = \max\{\mathtt{cost}_{\mathtt{hB}}(\pi) \mid \pi \text{ prefix path in } t\}$.

Let $[\![\mathcal{A}]\!]_{\min}(w) \leq n$: there exists a run ρ of w such that $\mathtt{cost}_{\mathtt{hB}}(\rho) \leq n$. Consider the run tree t consisting of exactly ρ, it satisfies $\mathtt{cost}_{\mathtt{hB}}(t) \leq n$. It follows that $[\![\mathcal{B}]\!]_{\min} \preccurlyeq [\![\mathcal{A}]\!]_{\min}$.

Conversely, let $[\![\mathcal{B}]\!]_{\min}(w) \leq n$: there exists a run tree t of w such that $\mathtt{cost}_{\mathtt{hB}}(t) \leq n$. Because it is a run of $\mathcal{B}$, there exists a run ρ in t, and $\mathtt{cost}_{\mathtt{hB}}(t) \leq n$ implies that $\mathtt{cost}_{\mathtt{hB}}(\rho) \leq n$. It follows that $[\![\mathcal{A}]\!]_{\min} \preccurlyeq [\![\mathcal{B}]\!]_{\min}$.

We conclude that $[\![\mathcal{A}]\!]_{\min}$ and $[\![\mathcal{B}]\!]_{\min}$ are equivalent. ◀

▶ **Lemma 17.** *$\mathcal{B}$ is history-deterministic.*

This relies on the following positionality result, which is proved in [7]. It is also in essence in the proof of Bojańczyk [3].

▶ **Theorem 18** ([7])**.** *Eve has positional uniform strategies in hB-games.*

We now prove Lemma 17.

Proof. To prove that $\mathcal{B}$ is history-deterministic, we show that there exists a function $\alpha : \mathbb{N} \to \mathbb{N}$ such that $\lim \alpha = \infty$ and for every n, there exists a strategy $\sigma : \mathtt{A}^* \to \Delta_{\mathcal{B}}$ such that for all words w, if $[\![\mathcal{B}]\!]_{\min}(w) \leq n$ then $[\![\mathcal{B}_\sigma]\!]_{\min}(w) \leq \alpha(n)$.

Observe that $\sigma : \mathtt{A}^* \to \Delta_{\mathcal{B}}$ can equivalently defined as a partial function $\sigma : Q \times \mathtt{A}^* \to \{I_1, R_1, \ldots, I_k, R_k\} \times Q$; what $\mathcal{B}$ guesses is for each state q, at most one transition leading to q.

We define an hB-game. The set of vertices is $Q \times \mathtt{A}^*$. The edges are

$$\{((wa, q), act, (w, p)) \mid (p, a, act, q) \in \Delta\}.$$

By definition, for all words w such that $[\![\mathcal{B}]\!]_{\min}(w) \leq n$, there exists $q \in F$ such that Eve has a strategy ensuring $\mathtt{cost}_{\mathtt{hB}}(n) \cap \mathtt{Safe}(\varepsilon, Q \setminus I)$. It follows from Theorem 18 that there exists a uniform positional strategy, *i.e.* $\sigma : Q \times \mathtt{A}^* \to \Delta$. By definition, for this strategy we have $[\![\mathcal{B}_\sigma]\!]_{\min}(w) \leq n$. It follows that $\mathcal{B}$ is history-deterministic. ◀

Composing the two automata yields a history-deterministic automaton equivalent to $\mathcal{A}$. Denote n the number of states of $\mathcal{A}$, the constructed automaton has $2^n \times \mathtt{Safra}(n)$ states, where $\mathtt{Safra}(n)$ is the number of states obtained by applying the Safra determinisation on an ω-automaton with n states. Since $\mathtt{Safra}(n) = 2^{O(n \log(n))}$, the constructed automaton also has $2^{O(n \log(n))}$ states.

Acknowledgements. We thank the anonymous reviewers for their constructive comments and suggestions.

References

1. Michael Benedikt, Balder ten Cate, Thomas Colcombet, and Michael Vanden Boom. The complexity of boundedness for guarded logics. In *LICS*, pages 293–304, 2015.
2. Achim Blumensath, Martin Otto, and Mark Weyer. Decidability results for the boundedness problem. *Logical Methods in Computer Science*, 10(3), 2014.
3. Mikołaj Bojańczyk. Star height via games. In *LICS*, pages 214–219, 2015. doi:10.1109/LICS.2015.29.
4. Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *ICALP*, pages 139–150, 2009.
5. Thomas Colcombet. Safra-like constructions for regular cost functions over finite words. Unpublished, March 2011.
6. Thomas Colcombet. Regular cost functions, part I: logic and algebra over words. *Logical Methods in Computer Science*, 9(3), 2013.
7. Thomas Colcombet and Christof Löding. The non-deterministic Mostowski hierarchy and distance-parity automata. In *ICALP*, pages 398–409, 2008.
8. Thomas Colcombet and Konrad Zdanowski. A tight lower bound for determinization of transition labeled büchi automata. In *ICALP*, pages 151–162, 2009.
9. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002.
10. Kosaburo Hashiguchi. Improved limitedness theorems on finite automata with distance functions. *Theoretical Computer Science*, 72(1):27–38, 1990.
11. Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *CSL*, pages 395–410, 2006.
12. Daniel Kirsten. Distance desert automata and the star height problem. *ITA*, 39(3):455–509, 2005.
13. Imre Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.
14. Imre Simon. On semigroups of matrices over the tropical semiring. *ITA*, 28(3-4):277–294, 1994.

Solutions of Word Equations Over Partially Commutative Structures*

Volker Diekert[1], Artur Jeż[†2], and Manfred Kufleitner[‡3]

1 **Institut für Formale Methoden der Informatik, Universität Stuttgart, Stuttgart, Germany**
2 **Institute of Computer Science, University of Wrocław, Wrocław, Poland**
3 **Institut für Formale Methoden der Informatik, Universität Stuttgart, Stuttgart, Germany**

Abstract

We give NSPACE($n \log n$) algorithms solving the following decision problems. Satisfiability: Is the given equation over a free partially commutative monoid with involution (resp. a free partially commutative group) solvable? Finiteness: Are there only finitely many solutions of such an equation? PSPACE algorithms with worse complexities for the first problem are known, but so far, a PSPACE algorithm for the second problem was out of reach. Our results are much stronger: Given such an equation, its solutions form an EDT0L language effectively representable in NSPACE($n \log n$). In particular, we give an effective description of the set of all solutions for equations with constraints in free partially commutative monoids and groups.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, F.4.2 Grammars and Other Rewriting Systems, F.4.3 Formal Languages

Keywords and phrases Word equations, EDT0L language, trace monoid, right-angled Artin group, partial commutation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.127

1 Introduction

Free partially commutative monoids (a.k.a. *trace monoids*) and groups (a.k.a. *RAAGs*: *right-angled Artin groups*) are well-studied objects, both in computer science (latest since [18]) and in mathematics (with increasing impact since [24]). For years, decidability of the *satisfiability problem* (i.e., the problem whether a given equation is solvable) over these structures was open. A positive solution for trace monoids was obtained by Matiyasevich [17] and for RAAGs by Diekert and Muscholl [8]. The known techniques did not cope with the *finiteness problem* (i.e., the problem whether a given equation has only finitely many solutions). Decidability of finiteness for trace monoids was wide open, whereas for RAAGs a sophisticated generalization of Razborov-Makanin diagrams and geometric methods, available for groups, yielded decidability [3], but without any complexity estimation.

We give a simple and effective description of the set of all solutions for equations with constraints in free partially commutative monoids and groups; the correctness proof is mathematically challenging. Once the correctness is established, the simplicity is also reflected in a surprisingly low complexity. We give an NSPACE($n \log n$) upper bound for both

* Full proofs can be found on arXiv [6].
† Artur Jeż was supported by a return fellowship of the Alexander von Humboldt Foundation.
‡ Manfred Kufleitner was supported by the grants DI 435/5-2 and KU 2716/1-1 of the DFG.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 127; pp. 127:1–127:14


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

satisfiability and finiteness—each problem for trace monoids as well as for RAAGs. Even for satisfiability this complexity improves the previously known upper bounds. On the other hand these problems are NP-hard. It remains open whether $\mathsf{NSPACE}(n \log n)$ is optimal.

To obtain these results we apply a recent *recompression technique* [12], which was used as a simple method to solve word equations. It uses simple compression operations: compress ab into a letter c; and modify the equation so that such operations are sound. An algebraic setting of the current paper enables a shift of perspective: the inverse operation, replacing c by ab, is an endomorphism. Thus, the set of all solutions of an equation (solvable or not) can be represented as a graph, whose nodes are labeled with equations and edges by endomorphisms of free monoids. This graph can also be seen as a nondeterministic finite automaton (NFA) that accepts a rational set of endomorphisms over a free monoid. (Recall that a subset in a monoid M is *rational* if it is accepted by some NFA whose transitions have labels from M.) It is known that applying a rational set of endomorphisms to a letter yields an EDT0L language [1], and our construction guarantees that the obtained EDT0L language describes exactly the set of all solution of the given equation. Moreover, as usual in automata theory, the structure of the NFA reflects whether the solution set is finite. Last not least, our method is conceptually simpler than all previously known approaches to solving equations over free partially commutative structures.

Studying word equations is part of combinatorics on words for more than half a century [2]. From the very beginning, motivation came partly from group theory: the goal was to understand and parametrize solutions for equations in free groups. For example, Lyndon and Schützenberger needed sophisticated combinatorial arguments to give a parametrized solution to the equation $a^m = b^n c^p$ in a free group [14]. On the other hand, it is known that a parametric description of the solution set is not always possible [10]. The satisfiability of word equations in free monoids and free groups became a main open problem due to its connection with Hilbert's tenth problem. The problem was solved affirmative by Makanin in his seminal papers [15, 16]. His algorithms became famous also due to the difficulty of the termination proof and the extremely high complexity. A breakthrough to lower the complexity was initiated by Plandowski and Rytter [21], who were the first to apply compression techniques on word equations. Compression was also essential in showing that the satisfiability of word equations is in PSPACE [19]. This approach was further developed [12] using the "recompression technique", which simplified all existing proofs for solving word equations; in particular, it provided an effective description of all solutions; a similar representation was given earlier by Plandowski [20]. In free groups, an algorithmic description of all solutions was known much earlier due to Razborov [22]. His description became known as a *Makanin-Razborov diagram*, a major tool in the positive solution of Tarski's conjectures about the elementary theory in free groups [13, 23]. None of these results provided a structural result on the set of all solutions; interest in such results was explicitly expressed [11] by asking whether it is an "indexed language". Apparently, this question was posed without too much hope that a positive answer is within reach. However, the answer was positive for quadratic equations [9] (which is a severe restriction); the general case was established in [4]. Actually, a stronger result holds: the set of all solutions for equations in free monoids (as well as in free groups) is an EDT0L language, which is a proper subclass of indexed languages. The closest results on word equations with partial commutation are in [8], but techniques used there do not apply here as they boil down to a purely combinatorial construction of a normal form and ignore the algebraic structure as well as the set of all solutions.

2 Main result

Given a finite *alphabet* Γ, the *free monoid* Γ^* is the set of all finite words over Γ with concatenation. The empty word is denoted by 1. The length of a word w is denoted by $|w|$; by $|w|_a$ we count how often the letter a appears in w. A *resource function* $\rho : \Gamma \to 2^{\mathfrak{R}}$ maps elements of Γ to subsets of a finite *set of resources* $\mathfrak{R}$. We assume that $\mathfrak{R}$ is of constant size. The pair (Γ, ρ) is called a *resource alphabet*. If $\rho(a) = S$, then a is called an *S-constant*; a nonempty sequence of S-constants is an *S-run*.

A *resource monoid* $M(\Gamma, \rho)$ is the quotient of all finite words Γ^* by a partial commutation: $M(\Gamma, \rho) = \Gamma^* / \{ab = ba \mid \rho(a) \cap \rho(b) = \emptyset\}$, i.e., letters $a \neq b$ commute if and only if they do not share a resource. Resource monoids can equivalently be seen as *free partially commutative monoids* or *trace monoids*. We choose the resource-based approach as it best suits our purposes. Elements of a resource monoid are called *traces*. The natural projection π maps elements of the free monoid Γ^* to traces in $M(\Gamma, \rho)$; this is not a bijection and we view $w \in \Gamma^*$ as a word representation of the trace $\pi(w)$. In a monoid, an element v is a *factor* of w if $w = pvq$ for some p, q. We assume that the monoid $M(\Gamma, \rho)$ is equipped with an *involution*, that is, a bijection $x \mapsto \overline{x}$ on $M(\Gamma, \rho)$ such that $\overline{\overline{x}} = x$, $\overline{xy} = \overline{y}\,\overline{x}$ for all $x, y \in M(\Gamma, \rho)$. To make the definition well defined, we require that $\rho(x) = \rho(\overline{x})$ for $x \in \Gamma$. In the following, a *trace monoid* means a *resource monoid with involution*. A *morphism* $\varphi : M \to M'$ between monoids with involution is a homomorphism additionally respecting the involution. If Δ is a subset of M, then we often denote the restriction of φ to Δ by φ. If $\varphi(d) = d$ for all $d \in \Delta$, then φ is a *Δ-morphism*.

If there is no letter $a \in \Gamma$ with $a = \overline{a}$, then, by adding defining relations $a\overline{a} = 1$ for all $a \in \Gamma$, we obtain the *free partially commutative group* $G(\Gamma, \rho)$. Free partially commutative groups are also known as *right-angled Artin groups* or *RAAGs* for short. As a set, we can identify a RAAG $G(\Gamma, \rho)$ with the subset traces of the trace monoid $M(\Gamma, \rho)$ without factors $a\overline{a}$. Such traces are called *reduced*. We take inversion on groups as involution; the canonical projection of the monoid $M(\Gamma, \rho)$ onto the group $G(\Gamma, \rho)$ respects the involution.

Let (Γ, ρ) be a resource alphabet. An *equation* is a pair of words (U, V) over an alphabet $\Gamma = A \cup \mathcal{X}$ has a partition into *constants* A and *variables* $\mathcal{X}$, both sets are closed under involution. A *constraint* is a morphism $\mu : M(\Gamma, \rho) \to N$, where N is a finite monoid with involution. For our purposes, it is enough to consider constraints such that the elements of N can be represented by $\mathcal{O}(\log |\Gamma|)$ bits, and that all necessary computations in N (multiplication, involution, etc.) can be performed in space $\mathcal{O}(\log |\Gamma|)$ and the specification of these operations requires $\mathcal{O}(|\Gamma| \log |\Gamma|)$ space. If (U, V) is an equation over (Γ, ρ), then we define the input size of an equation with constraints as $n = |UV| + |\Gamma|$.

We write (U, V, μ) for an equation (U, V) with constraints μ. A *solution* of (U, V, μ) over $M(A, \rho)$ is an A-morphism $\sigma : M(A \cup \mathcal{X}, \rho) \to M(A, \rho)$ such that $\sigma(U) = \sigma(V)$ and $\mu\sigma(X) = \mu(X)$ for all $X \in \mathcal{X}$. If the equation is over $G(A, \rho)$, then instead of $\sigma(U) = \sigma(V)$ we require $\pi\sigma(U) = \pi\sigma(V)$ for the canonical projection $\pi : M(A, \rho) \to G(A, \rho)$. We also say that σ *solves* (U, V, μ) in $M(A, \rho)$ (resp. in $G(A, \rho)$). For equations over $G(A, \rho)$ we only allow solutions where the trace $\sigma(X)$ is reduced for all $X \in \mathcal{X}$. The main result of this paper is that the set of all solutions of a trace equation (resp. an equation in a RAAG) with rational constraints is an effectively computable EDT0L language, and the underlying automaton reflects whether there are infinitely many solutions.

▶ **Theorem 1.**

[Monoid version] *There is an* NSPACE$(n \log n)$ *algorithm for the following task. The input is a resource alphabet* $(A \cup \mathcal{X}, \rho)$ *with involution and a trace equation* (U, V, μ) *with constraints*

μ in constants A and variables $\mathcal{X} = \{X_1, \ldots, X_k\}$. The algorithm computes an alphabet $C \supseteq A$ of size $\mathcal{O}(n)$, constants $c_1, \ldots, c_k \in C$, and an NFA $\mathcal{A}$ accepting a rational set $\mathcal{R}$ of A-endomorphisms on C^* such that: $h(C^*) \subseteq A^*$ for all $h \in \mathcal{R}$ and under the canonical projection $\pi : A^* \to M(A, \rho)$ we have

$$\{(\pi h(c_1), \ldots, \pi h(c_k)) \mid h \in \mathcal{R}\} = \{(\sigma(X_1), \ldots, \sigma(X_k)) \mid \sigma \text{ solves } (U, V, \mu) \text{ in } M(A, \rho)\}.$$

Thus, the set of all solutions is an effectively computable EDT0L language. Furthermore, (U, V, μ) has a solution if and only if $\mathcal{A}$ accepts a nonempty set; (U, V, μ) has infinitely many solutions if and only if $\mathcal{A}$ has a directed cycle. These conditions can be tested in $\mathsf{NSPACE}(n \log n)$.

[Group version] *The same, but solutions σ satisfy $\sigma(U) = \sigma(V)$ in the RAAG $G(A, \rho)$ and for a variable X the solution $\sigma(X)$ is restricted to be a reduced trace.*

Theorem 1 generalizes to systems of equations. Another generalization are finitely generated *graph products* with involution over free monoids, free groups, and finite groups. See [7] for a definition and the known results concerning solvability of equations in graph products. This generalization is rather technical but does not reveal new ideas; it is done elsewhere.

3 Basic concepts

Equations in RAAGs can be reduced to equations in resource monoids [8], such an approach is standard since its introduction for free groups [5], which are reduced to free monoids. In essence, the reduction simulates the inverse operation by involution and it enforces that the solution in the monoids is in the reduced form by (additional) constraints. We employ a similar approach; thus, our presentation focuses on the equations over resource monoids.

Using a standard technique one can ensure that there are no self-involuting constants in the initial equation [8]. This step is not needed for RAAGs as $a = \overline{a} = a^{-1}$ implies $a^2 = 1$ in groups, but RAAGs are torsion-free. For technical reasons we introduce a new special symbol $\#$, which serves as a marker and becomes the only self-involuting constant; set $\rho(\#) = \mathfrak{R}$ and $\emptyset \neq \rho(a) \subsetneq \mathfrak{R}$ for all other constants. We let

$$W_{\text{init}} = \#X_1 \# \cdots X_k \# U \# V \# \overline{U} \# \overline{V} \# \overline{X_k} \# \cdots \overline{X_1} \#.$$

During the process, the $\#$'s will not be touched, so we keep control over the prefix corresponding to $\#X_1\# \cdots \#X_k\#$ which encodes the tuples $(\sigma(X_1), \ldots, \sigma(X_k))$. Moreover, we have $\sigma(U) = \sigma(V)$ if and only if $\sigma(W) = \sigma(\overline{W})$. Thus, we can treat a single trace as an equation. Solutions become A-morphisms σ satisfying $\sigma(W) = \sigma(\overline{W})$.

The equations that we consider are over a more general structure than a trace monoid. To simplify the notion, we denote the equation and a monoid over which it is by a tuple $(W, B, \mathcal{X}, \rho, \theta, \mu)$, where $W \in (B \cup \mathcal{X})^*$ is the "equation" with constants B and variables $\mathcal{X}$, the mapping $\rho : B \cup \mathcal{X} \to 2^{\mathfrak{R}}$ is the resource function, and $\mu : M(B \cup \mathcal{X}, \rho) \to N$ represents the constraints (given by the mapping $\mu : B \cup \mathcal{X} \to N$). Since $2^{\mathfrak{R}}$ is a commutative monoid, we shall view ρ as a morphism from $M(B \cup \mathcal{X}, \rho)$ to $2^{\mathfrak{R}}$, too. The symbol θ refers to a "type" which adds partial commutation. A type is given by a list of certain pairs $(x, y) \in (B \cup \mathcal{X})^+ \times B^+$; and each such pair yields a defining equation $xy = yx$. For example, we typically have $(X, y) \in \theta$ when considering a solution σ with $\sigma(X) \in y^+$. Then $\rho(X) = \rho(y)$, but we wish to use the fact that $\sigma(X)y = y\sigma(X)$. This is the purpose of a

type. We only use types in the subprocedures of block and quasi-block compression, see Section 4.3.1. Such a monoid is denoted as $M(B \cup \mathcal{X}, \rho, \theta, \mu)$. In most cases, θ is empty. Then we use $(W, B, \mathcal{X}, \rho, \mu)$ as an abbreviation of $(W, B, \mathcal{X}, \rho, \emptyset, \mu)$ and $M(B \cup \mathcal{X}, \rho, \theta, \mu)$ as an abbreviation of $M(B \cup \mathcal{X}, \rho, \mu)$.

A *B-solution* of $(W, B, \mathcal{X}, \rho, \mu)$ is a B-morphism $\sigma : M(B \cup \mathcal{X}, \rho, \mu) \to M(B, \rho, \mu)$ such that $\sigma(W) = \sigma(\overline{W})$ (i.e., it solves the equation) and $\mu(\sigma(X)) = \mu(X)$ (i.e., it satisfies the constraints).

During the algorithm we "increase the resources" of constants. It is useful to assume that for every constant $a \in A$ and every set of resources S with $\rho(a) \subsetneq S$ the alphabet A has a corresponding constant with set of resources S. We denote such a constant by (a, S), the involution on it is defined by $\overline{(a, S)} = (\overline{a}, S)$. We naturally identify a with $(a, \rho(a))$. We assume that the initial alphabet A is closed under taking such constants, i.e., if $a \in A$ and $\rho(a) \subseteq S$, then $(a, S) \in A$.

In some cases, when we "increase the resources", we prefer to use a fresh constant of appropriate resources: For a constant a with $\rho(a) \subsetneq S$, by $[a, S]$ we denote a "fresh" S-constant outside A such that $\rho([a, S]) = S$, $\mu([a, S]) = \mu(a)$ and $\overline{[a, S]} = [\overline{a}, S]$; replacing a with $[a, S]$ is called *lifting*.

During the algorithm we perform various operations on variables and constants. As a rule, whenever we perform such an operation, we perform a symmetric action on the involuted constants/variables. That is, whenever we replace X by aX, we replace $\overline{X}$ by $\overline{X}\,\overline{a}$; and when we replace ab by c, then we also replace $\overline{b}\,\overline{a}$ by $\overline{c}$. This simplifies the description, as actions performed on "the right side" of X are actions performed on "the left side" of $\overline{X}$.

Whenever we perform operations on variables/constants, we want the constraints and resources to remain unaffected (except for lifting, in which case we explicitly change the set of resources); if we replace a trace W by a trace W', then (if not explicitly stated otherwise) we ensure that $\rho(W) = \rho(W')$ and $\mu(W) = \mu(W')$. For instance, when replacing X by aX', we set $\mu(X')$ so that $\mu(aX') = \mu(X)$. The same applies to ρ. Similarly, when replacing ab by c, we set $\rho(c) = \rho(ab)$ and $\mu(c) = \mu(ab)$. In particular, we do not mention in the description of the procedures that we perform such operations.

A trace $a_1 a_2 \cdots a_n$ has many word representations and we would like to formalize a notion that some constants occur before others (in all word representations). To this end consider a set of positions $\{1, 2, \ldots, n\}$ and the smallest partial order $\preceq$ such that $i \preceq j$ if both $i \leq j$ and $\rho(a_i) \cap \rho(a_j) \neq \emptyset$. A *Hasse diagram* $H(W)$ of a trace $W = a_1 a_2 \cdots a_n$ is a graph with a set of nodes $\{1, 2, \ldots, n\}$, labeled with $a_1, a_2, \ldots, a_n$. It contains (directed) edges between immediate successors, i.e., (i, j) is an edge if $i \prec j$ and $i \preceq k \preceq j$ implies $k \in \{i, j\}$. By a standard result in trace theory [18], we have $W = W'$ in $M(\Gamma, \rho)$ if and only if $H(W)$ and $H(W')$ are isomorphic as abstract node-labeled directed graphs. When considering traces we usually work with their Hasse diagrams. If this causes no confusion, we identify $W = a_1 \cdots a_n$ with $H(W)$ and refer to labels $a_1, a_2, \ldots$ rather than to node names.

A constant $a \in A$ is *minimal* in a trace W if it is minimal in its Hasse diagram, which means that $W = aY$ for some trace Y. We denote the set of minimal elements of W by $\min(W)$. Maximal elements are left-right dual; they are denoted by $\max(W)$.

An arc $a \to b$ is an *S-arc* if $S \in \{\rho(a), \rho(b)\}$; it is *balanced* if $\rho(a) = \rho(b)$, *unbalanced* otherwise.

4 NFA recognising the set of all solutions

In this section we define the NFA $\mathcal{A}$ that recognises the set of all solutions of a trace equation, treated as a set of endomorphisms $\mathrm{End}(C^*)$ over an alphabet $C \supseteq A$.

4.1 The automaton

We first fix an alphabet of constants $C \supseteq A$ of size κn, where κ is a suitable constant (which depends on the number of resources $|\mathfrak{R}|$, viewed as $\mathcal{O}(1)$) and a set of variables Ω with $|\Omega| \leq |C|$. Henceforth, we assume $A \subseteq B \subseteq C$ and $\mathcal{X} \subseteq \Omega$.

The states of the automaton $\mathcal{A}$ are equations of the form $(W, B, \mathcal{X}, \rho, \theta, \mu)$. Each state $V = (W, B, \mathcal{X}, \rho, \theta, \mu)$ has a *weight* $\|V\|$ which is a 5-tuple of natural numbers:

$$\|V\| = (|W|, \omega, \omega', |\theta|, |B|) \in \mathbb{N}^5$$

with $\omega = \sum_{a \in B} (|\mathfrak{R}| - |\rho(a)|) \cdot |W|_a$ and $\omega' = |W| - |\{a \in B \mid |W|_a \geq 1\}|$. We order tuples in $\mathbb{N}^5$ lexicographically. The NFA contains only states V whose *max-norm* $\|V\|_\infty = \max\{|W|, \omega, \omega', |\theta|, |B|\} \in \mathbb{N}$ is at most $\kappa' n$ for a suitable constant κ'.

The initial state is $(W_{\mathrm{init}}, A_{\mathrm{init}}, \mathcal{X}_{\mathrm{init}}, \rho_{\mathrm{init}}, \mu_{\mathrm{init}})$, it corresponds to the input equation. A state $(W, B, \emptyset, \rho, \mu)$ without variables is final if $W = \overline{W}$ has the prefix $\#c_1\# \cdots \#c_k\#$, where $c_1, \ldots, c_k$ are the distinguished constants. We require that the initial state has no incoming and the final states no outgoing transitions.

The transitions, say between $V = (W, B, \mathcal{X}, \rho, \theta, \mu)$ and $V' = (W', B', \mathcal{X}', \rho', \theta', \mu')$, are labeled by A-morphisms and they either affect the variables (*substitution transitions*), or the monoid (*compression transitions*). The former is formalized using a B-morphism $\tau : M(B \cup \mathcal{X}, \rho, \theta, \mu) \to M(B' \cup \mathcal{X}', \rho', \theta', \mu')$. In this case we put several requirements on the equations: the new equation should be obtained by substitution $\tau(X)$ for each X, there are no new constants, resources and constraints of X and $\tau(X)$ should be the same; this is formalized as

$$W' = \tau(W), \quad B' = B, \quad \rho = \rho'\tau, \quad \mu = \mu'\tau. \tag{1}$$

Moreover, $\tau(X)$ is either the empty word (it removes X from W) or $\tau(X) \in \mathcal{X}^* B^+ \mathcal{X}^*$ (at least one constant pops up in the substituted variable). Note that the requirement $W' = \tau(W)$ implicitly upper-bounds the *size* $\|\tau\|$, defined as $\sum_{a \in B \cup \mathcal{X}} |\tau(a)|$, to be linear.

Furthermore, as $B' = B$ we have a natural identity morphism from $M(B', \rho', \theta', \mu')$ to $M(B, \rho, \theta, \mu)$, call it the *associated morphism* and denote it by ε. This morphism labels the transition, its direction is opposite of the transition and τ; we denote the transitions from V to V' with a corresponding morphism h by $V \xrightarrow{h} V'$.

A compression transition leaves the variables invariant and so it is defined by an $(A \cup \mathcal{X})$-morphism $h : M(B' \cup \mathcal{X}, \theta', \rho') \to M(B \cup \mathcal{X}, \theta, \rho)$, note that it could be that $\theta \neq \theta'$ which corresponds to a type introduction or removal; this is the associated morphism in this case. A morphism h defined by, say $h(c) = ab$, represents a compression of a factor ab into a single letter c. For its properties, W is obtained by decompression of new constants, and the resources and constraints are preserved:

$$W = h(W'), \quad \rho' = \rho h, \quad \mu' = \mu h. \tag{2}$$

As in the case of substitutions, the assumption that $W = h(W')$ implicitly upper-bounds $\|h\|$ to linear values.

The transition in the NFA is in the direction of the compression which is opposite to direction of the morphism h. Note that $W = h(W')$ implies $\|V'\| < \|V\|$. For technical reasons we do not allow compression transitions which introduce self-involuting letters (such as $c \mapsto a\overline{a}$); we never compress the marker symbol #. Moreover, following the last compression transition to final states, the restriction $\|V'\| < \|V'\|$ is not applied.

So far the defined NFA can have many useless states, so as a last step we *trim* the automaton, i.e., we remove all vertices not appearing on some accepting path.

The algorithmic part is finished: $\mathcal{A}$ can be constructed using standard arguments in $\mathsf{NSPACE}(n \log n)$.

By the usual definition, the recognized language $\mathcal{R}$ consists of all A-morphisms $h_1 \cdots h_k$, where $h_1, \ldots, h_k$ are consecutive labels on an accepting path. We claim that the set of all solutions is exactly $\{(\pi h(c_1), \ldots, \pi h(c_k)) \mid h \in \mathcal{R}\}$ where $\pi : A^* \to M(A, \rho)$ is the natural projection.

The correctness proof boils down to show that we can calculate the exact constants κ, κ' (depending on $\mathfrak{R}$ but not on n) and to prove soundness and completeness, i.e., that $h \in \mathcal{R}$ yields a solution and that every solution can be obtained in this way. Out of those, soundness is relatively easy to show, see Section 4.2, the completeness argument spans over Sections 4.3–4.4. Those arguments also show the other claims on the automaton (conditions for emptiness and acyclicity).

4.2 Soundness

As the final states have only one solution (identity), using an induction on the following Lemma, any accepting path labeled with $h_1, \ldots, h_k$ yields a solution $\pi h_1 \cdots h_k$, which shows soundness.

▶ **Lemma 2.** *Given two states $V = (W, B, \mathcal{X}, \rho, \theta, \mu)$ and $V' = (W', B', \mathcal{X}', \rho', \theta', \mu')$, if $V \xrightarrow{h} V'$ and V' has a B'-solution σ' then V has a B-solution $\sigma = h\sigma'$.*

The proof follows by a mechanical application of (1) or (2).

4.3 On-the-fly construction of the NFA

While we described the NFA recognizing all solutions, we did not discuss how to find the appropriate constants κ, κ' nor how to show completeness. For this it is easier to first describe the construction as an "on-the-fly" algorithm, that is, given an equation $(W, B, \mathcal{X}, \rho, \theta, \mu)$ (= current state V of the NFA) and its B-solution σ we will transform it into a different equation $(W', B', \mathcal{X}', \rho', \theta', \mu')$ (= next state V' of the NFA) and a corresponding B'-solution σ', where $V \xrightarrow{h} V'$ and $\sigma = h\sigma'$. Thus we moved from one state of the NFA to the other, without the knowledge of the full NFA. Note that the solutions are not given explicitly, but they are "used" in the nondeterministic choices of the algorithm.

For a fixed set of resources S traces consisting only of S-constants and variables behave as words and we apply to them the known recompression approach: we iteratively apply compression operations to S-runs (so we replace S-runs by new single S-constants). Those operations are applied on constants in the equation, but conceptually we apply them to a solution of the trace equation. To make this approach sound, we also modify the variables, by popping S-constants from them. We apply these operations until each S-run (in the solution) is reduced to a single S-constant; when the compressions and popping operations are applied in appropriate order, the size of the trace equation remains linear.

Compression of S-runs alone is not enough, as there are constants of different resources in the solution of the trace equation. To remedy this, we gradually linearize the solution. This is done by increasing the set of resources of particular constants: when we compressed each S-run to a single constant, we lift all S-constants, so that all S-constants and S-variables are eliminated from the equation. To make the whole approach work, we define an order $\leq$ on sets of resources: it is any linear order that extends the partial ordering by the size, i.e., $|S| \leq |T|$ implies $S \leq T$. A set of resources S is called *minimal* (for a solution σ), if it is minimal according to $\leq$ in the set $\{T \mid \text{there is a } T\text{-constant in } \sigma(W)\}$. We process the sets of resources according to $\leq$, each time treating a minimal sets of resources.

4.3.1 Fixed resources

We define the actions of the algorithm eliminating the S-constants for a fixed minimal set of resources S. To thi end, we need some notions of "easy" and "difficult" factors of $\sigma(W)$.

▶ **Definition 3.** Let $(W, B, \mathcal{X}, \rho, \mu)$ be a state and σ its B-solution. A factor v of $\sigma(W)$ is *visible* if for some occurrence at least one of its positions is obtained form a position labeled by a constant in W; a factor is *invisible* if it is not visible. A trace v is *crossing* if for some occurrence of v in $\sigma(W)$ some but not all positions belong to the substitution of a variable X by $\sigma(X)$; and this occurrence is *visibly crossing*. A trace is *noncrossing* if it is not crossing.

The factors that we typically consider are *pairs*, i.e., ab where $a \neq b \neq \overline{a}$, *$a$-blocks*, i.e., a maximal factor of the form a^ℓ (this occurrence of a^ℓ if not part of a factor $a^{\ell+1}$), and *a-quasi-blocks*, i.e., $(a\overline{a})^\ell$ that is not part of a factor $(a\overline{a})^{\ell+1}$. In the latter case, $a\overline{a}$ is called a *quasi-letter*. The intuitive meaning of a quasi-letter is that we cannot compress $a\overline{a}$ into a single constant as it is would be self-involuting, hence we treat those two letters as if they were a single constant.

Given a subalphabet $S_\pm$, we consider an *involuting partition* (S_+, S_-) that satisfies the conditions $\overline{S_+} = S_-$, $S_+ \cap S_- = \emptyset$ and $S_+ \cup S_- = S_\pm$. Such a partition is *crossing* if at least one pair $ab \in S_+S_-$ is; it has crossing quasi-blocks if there is $a \in S_+$ that has crossing quasi-blocks. Lastly, $S_\pm$ has crossing blocks if there is $a \in S_\pm$ that has crossing blocks.

Pair compression is implemented essentially in the same way as in the case of word equations. Given a pair ab with $a \neq b \neq \overline{a}$ we want to replace each factor ab in $\sigma(W)$ with a fresh constant c. This is easy, when ab is noncrossing: it is enough to perform this operation on W and each $\sigma(X)$, the latter is done implicitly and we obtain a different solution σ' in this way. We also set ρ and μ for c appropriately: $\rho(c) = \rho(ab)$ and $\mu(c) = \mu(ab)$. Performing several such compressions is possible for $ab \in S_+S_-$, where (S_+, S_-) is a noncrossing involuting partition, as for each constant in $\sigma(W)$ we can uniquely determine to which replaced pair it belongs (if any). We do not compress pairs $a\overline{a}$, though, as this would create a self-involuting letter.

We need to ensure that indeed (S_+, S_-) is noncrossing. A pair $ab \in S_+S_-$ is crossing if aX is a factor of W and $b \in \min(\sigma(X))$. The other option is that Xb is a factor and $a \in \max(\sigma(X))$; it is taken care by considering $\overline{b}\,\overline{X}$ and the pair $\overline{b}\overline{a}$. Then we replace X with bX. After doing this, for all variables, the partition (S_+, S_-) is noncrossing and so we can compress pairs in this partition.

Pair compression cannot be applied to aa, as it makes the compression of longer blocks ambiguous. However, when a has no crossing block, (in several steps) we replace each a-block a^λ by c_λ. Similarly as in the case of pair compression, we can compress blocks of several letters in parallel, as blocks of different letters do not overlap.

Again, to apply this subprocedure we need to ensure that each $a \in S_\pm$ has no crossing blocks. Given a visibly crossing block a^ℓ, popping one node may be not enough as this block may still be crossing. Thus for each variable X we pop its whole a-prefix whenever $a \in \min(X) \cap S$, where a^ℓ is the a-prefix of a trace V when ℓ is maximal with $V = a^\ell V'$.

We do not apply the pair compression to $a\overline{a}$ as this introduces self-involuting letters. Instead, we perform a variant of block compression on them: the quasi-block compression. We replace each a-quasi-block $(a\overline{a})^\lambda$ with $c_\lambda \overline{c_\lambda}$; note that we treat a and $\overline{a}$ asymmetrically. We again perform this operation in parallel (in several steps), for all $a \in S_+$, where (S_+, S_-) is an involuting partition.

For uncrossing of quasi-blocks we act the same as for uncrossing of blocks, but we pop the whole $(a\overline{a})$-prefix when $a \in S_+$; the $a\overline{a}$ prefix of V is the longest factor $V' \in \overline{a}(a\overline{a})^* \cup (a\overline{a})^*$ such that $V = V'V''$.

Using those operations we can process a minimal set of resources S: We iterate the following operations as long as something changes in the equation. For each variable we guess whether it has a minimal S-letter and if so we pop this letter. Then we compute the set $S_\pm$ of visible S-constants. We uncross blocks from $S_\pm$ and then compress blocks of $S_\pm$. We then arbitrarily partition $S_\pm$ into an involuting partition (S_+, S_-). Then we uncross quasi-blocks for S_+ and then compress quasi-blocks from S_+. We again partition $S_\pm$ into an involuting partition (S_+, S_-); the partition is chosen so that there are many occurrences of pairs in S_+S_- in the equation, see the appendix. Finally, we uncross (S_+, S_-) for pair compression and perform the pair compression for (S_+, S_-).

Using similar arguments as in the case of word equations, one can show that the procedure $\mathsf{FixedResources}(S)$ for a fixed set of resources uses linear space. Concerning the S-runs after $\mathsf{FixedResources}(S)$, ideally all S-runs are of length 1 and are either visible or invisible. This is not entirely true, as $a\overline{a}$ cannot be compressed, but those are the longest visible S-runs that can prevail.

▶ **Lemma 4.** *Let S be minimal. The length of the equation during* $\mathsf{FixedResources}(S)$ *is linear. After* $\mathsf{FixedResources}(S)$ *there are no crossing S-runs, no S-variables. Furthermore, visible S-runs have length at most* 2.

4.3.2 Lifting arcs

Compression of S-runs alone is not enough, as there are runs for different sets of resources. To remedy this we linearize the trace, for technical reasons it is easier to lift whole Hasse arcs rather than individual nodes.

To lift a Hasse arc $e = (a \to b)$ we want to relabel its ends by $[a, \rho(a) \cup \rho(b)]$ and $[b, \rho(a) \cup \rho(b)]$, i.e., by fresh $(\rho(a) \cup \rho(b))$-constants. For correctness reasons we need to also lift the edges that "correspond" to e; moreover, as in the case of compression, lifting may be difficult when an arc connects constants in the equation with constants in the substitution for a variable. Those notions are formalized below.

▶ **Definition 5.** Let $(W, B, \mathcal{X}, \rho, \mu)$ be a state and σ its B-solution. A Hasse-arc $a \to b$ in $\sigma(W)$ is *visible* (*invisible*, *visibly crossing*) if the corresponding factor ab in $\sigma(W)$ has this property. Let $\sim$ be the smallest equivalence relation which satisfies the following conditions:

- If $e = (a \to b)$ in $\sigma(W)$ and $f = (\overline{b} \to \overline{a})$ is the corresponding arc in $\sigma(\overline{W})$, then $e \sim f$.
- If e is invisible and inside some $\sigma(X)$ where $X \in \mathcal{X}$ and f is a corresponding arc in some different $\sigma(X)$, then $e \sim f$.

We say that e is *crossing* if there is exists a visibly crossing f with $f \sim e$; e is *free* otherwise.

Note that for arcs the notion of crossing/free is finer than for traces: since it is possible that $e \not\sim e'$ while both are of the form $(a \to b)$, in particular e could be free and e' crossing.

When $e = (a \to b)$ is a free unbalanced arc, the promised linearization of traces is done through *lifting*: let $S = \rho(a) \cup \rho(b)$, then for $f \sim e$ we change the label on each of its ends from $c \in \{a, b, \overline{a}, \overline{b}\}$ to $[c, S]$. Note that this balances f. To make this operation well defined, we partially linearizes a trace: each position that was before (after) any of relabeled a, b is now before (after) both of $[a, S], [b, S]$ (the same is done for arc $\overline{b} \to \overline{a}$).

We can lift free arcs "for free", but some S-arcs may be crossing. Freeing them is similar to uncrossing factors, but we need to take into the account that $\rho(a) \neq \rho(b)$. Thus ab could be a crossing arc in aX and b is not a minimal element of $\sigma(X)$, so it cannot be popped. Freeing is done in two stages: first we deal with the case when b is an S-letter. Then for $\sigma(X) = PbQ$, such that $S \neq \rho(P) \subsetneq \rho(X)$ we pop the whole P, which is done by introducing a fresh variable, i.e., we substitute $X \mapsto X'bX$. The new solution is $\sigma'(X') = P$ and $\sigma'(X) = Q$. Then we deal with the case when a is an S-letter (and b not). Thus for $\sigma(X) = PbQ$, where $\rho(a) \cap \rho(P) = \emptyset$, we substitute $X \mapsto X'bX$. The new solution is $\sigma'(X') = P$ and $\sigma'(X) = Q$. Those operations are called splitting of variables. Observe that the first splitting can be done for any set of S-constants and all variables in parallel, while the second can be performed in parallel for all variables and any set of constants that is a subset of $\{b \mid \rho(b) \cap S \neq \emptyset\}$.

We want to lift all unbalanced S arcs, but this is not possible for *all* such arcs in parallel due to involution: for an S-letter a and a trace bac we have to choose which arc, $b \to a$ or $a \to c$, we lift. But it can be done in stages: let (S_+, S_-) and (T_+, T_-) be involuting partitions of all S-constants and all constants having a common resource with S, i.e., $\{a \mid \rho(a) \cap S \neq \emptyset\}$. Then we process all S arcs in four groups S_+T_+, S_-T_+, S_+T_- and S_-T_-; processing of each one is similar, we describe processing of one — S_+T_+. We first split the variables for S_+ and then for T_+, as described above. Then each arc $(a \to b)$ with $ab \in S_+T_+$ is free, thus we lift those arcs. We continue with groups S_-T_+, S_+T_- and S_-T_-. Note that the processing may introduce new crossing arcs, but it can be shown that they are always in next groups. Afterwards, there are no S arcs.

Let $\mathsf{Remove}(S)$ be the above procedure for lifting the S-letters. It is easy to show that after $\mathsf{Remove}(S)$ all S-constants and S-variables are eliminated.

▶ **Lemma 6.** *After* $\mathsf{Remove}(S)$ *there are neither* S*-constants nor* S*-variables in* $\sigma(X)$*.*

4.3.3 The algorithm

TrEqSat considers possible sets of resources S in order $\leq$ on them. For a fixed S it first runs $\mathsf{FixedResources}(S)$ and then $\mathsf{Remove}(S)$.

4.4 Analysis

We begin with estimating the space usage. Firstly we upper-bound the number of introduced variables: they are introduced only during splitting of variables, which happens $\mathcal{O}(1)$ times per resource set, and each variable introduces $\mathcal{O}(1)$ variables, which have less resources; this yields that the number of occurrences of variables is linear in the size of the input equation.

We then estimate the length of the equation, which is also linear in the size of the input equation: Here the estimations are similar as in the case of word equations. For a fixed resource set S we claim that the number of S-constants in the equation stays linear and that processing S introduces in total $\mathcal{O}(1)$ constants per variable. Together with the estimation on the number of variables this yields a bound on the size of the equation.

This guarantees that our algorithm does not exceed a space limit, but may loop forever. Thus we want to show that solutions in consecutive steps get "smaller". Unfortunately, the length of $\sigma(W)$ is not good enough for our purposes, but we can define the weight of the solutions (for an equation) and indeed show that our subprocedures decrease it. This guarantees termination.

We then move to the correctness of the algorithm, i.e., we show how the algorithm transforms the solutions between different equations obtained on the way. In a first step we equip each solution with a function that tells us, what solution of the *input* equation it represents. Then we show that if subprocedure transforms one equation into the other, then the morphism associated with this transition transforms the solution of the latter equation to a solution of the former, so that they they represent the same solution of the input equation.

4.4.1 Space usage

The below estimations of space usage do not depend on the nondeterministic choices, they apply to *all* executions of the algorithm.

Comparing to the algorithms in the free monoid case, the main difference is that our algorithm introduces new variables to the equation. This is potentially a problem, as the whole recompression is based on the assumption that the number of constants is not altered. However, we can still bound the number of introduced variables.

▶ **Lemma 7.** *During* TrEqSat *there are $\mathcal{O}(n)$ occurrences of variables in the trace equation.*

Fix a variable X for which initially $T = \rho(X)$. Observe that $\rho(X)$ cannot increase, though it can decrease: resources increase by lifting arcs and we only lift free arcs, thus, each resource of the new constant was present on one of the ends of the arc. On the other hand, popping constants as well as splitting may decrease the resources of a variable.

We say that X *directly created* an occurrence of X' when X' was created in Split when it considered X; X created X' when there is a sequence $X = X_1, X_2, \ldots, X_k = X'$ such that X_i directly created X_{i+1}. Consider a variable X, it can be split at most eight times during lifting of crossing arcs when we consider $T' \subseteq \mathfrak{R}$. This gives all variables that are directly created by X. Note, that each of the directly generated variable has less resources than X: when we replace X with $X'bX$, then we require that $\rho(X') \subsetneq \rho(X'bX)$.

Let $f(k)$ be the maximal number of occurrences of variables that can be created by a variable with at most k resources. Using the above analysis we can write a recursive formula for f; as the number of resources is a constant, this yields the bound.

We show that during TrEqSat the length of the trace equation is linear in the size of the variables, this is similar as in the case of word equations and in fact the proof proceeds using similar steps. First, we focus on FixedResources and its processing of a fixed set of resources S. In each application of the while loop we introduce $\mathcal{O}(1)$ S-constants per variable (in case of block and quasi-block compression we may introduce long blocks but they are replaced with $\mathcal{O}(1)$ constants afterwards). On the other hand, using standard expected value argument we can show that compression of a randomly chosen partition results in removal of a constant fraction of S-constants from the equation.

Comparing the number of constants in the equation before and after processing S, it increases only by the S-factors that were popped from variables. There are $\mathcal{O}(1)$ such factors for a variable and each is of length at most 2. Thus for a fixed set of resources the size of the equation increases by $\mathcal{O}(n)$. Summing over possible sets of resources (which is of constant size) yields the claim.

▶ **Lemma 8.** *During* TrEqSat *the length of the trace equation is $\mathcal{O}(n)$.*

4.4.2 Weight of solutions

To guarantee the termination, we show that all subprocedures decrease the (appropriately defined) weight of a solution. This weight is in fact defined with respect to the original solution: The B-solution σ corresponds to some solution of the input equation, as letters of B correspond to some traces in the original equation. To keep track of those traces we use an A-morphism $\alpha : M(B, \rho, \theta, \mu) \to M(A, \rho_0, \mu_0)$; the idea is that $c \in B$ represents a trace $\alpha(c)$ in $M(A, \rho_0, \mu_0)$. Conceptually, $\alpha(\sigma(W))$ is the corresponding solution of the input equation. We call a pair (σ, α) a *solution* at $(W, B, \mathcal{X}, \rho, \mu)$, where σ is a B-solution. Note that this morphism is a tool of analysis and proof, it is neither computed nor stored anywhere by the algorithm.

Using the morphism we define the *weight* of a solution (α, σ) as $\|\alpha, \sigma\| = \sum_{X \in \mathcal{X}} |\alpha\sigma(X)|$. All subprocedures performed by our algorithm do not increase the weight. In order to ensure that they all decrease some "weight", we take into the account also the weight of the equations and define a weight of a solution (α, σ) at a state V as $(\|\alpha, \sigma\|, \|V\|)$ which is evaluated in lexicographic order. All subprocedures decrease such defined weight. Thus, the path in NFA for a fixed solution is finite and terminates in a final state.

4.4.3 Internal operations

So far all the described operations were performed on the equation and had some influence also on the solutions. However, there are also operations that are needed for the proof but are performed either on the monoid or on the solutions alone, hence they do not affect the equation at all. For this reason we call them *internal*. In essence, we apply them to the equation whenever this is possible.

A constant $a \in B \setminus A$ is *useless* if it does not occur in $\sigma(W)$; it is *useful* otherwise; useless constants are invisible. A variable is *useless* if it does not occur in W. We remove from the monoid all useless constants and variables.

Due to compression we can be left with invisible but useful constants, i.e., such that they occur in $\sigma(W)$ but not in W.

We cannot remove such constants from B, as we deal with all solutions. However, we can replace them with corresponding traces over $M(A, \rho_0, \mu_0)$. The idea is that we replaced $\alpha(c)$ with c too eagerly. We revert this compression. We do not revert the linearization of the trace, though. Thus we lift each letter in $\alpha(c)$ so that it has the same resources as c: we replace every invisible letter c with $(a_1, \rho(c))(a_2, \rho(c)) \cdots (a_\ell, \rho(c))$, where $\alpha(c) = a_1 a_2 \cdots a_\ell$, i.e., with a chain of letters corresponding to the trace compressed into c but lifted into current resources of c. Note that we use letters from $A \subseteq B$, so the procedure is not applicable to letters that it just introduced.

4.4.4 Completeness

The last step to show completeness is an observation that each given subprocedure corresponds to a composition of finitely many substitution and compression transitions: indeed, this is done by mechanical verification.

The completeness, formulated below, easily follows: given an equation with a solution (α, σ) we apply the subprocedures that lead to a final state. By observation above each subprocedure corresponds to a short path in the NFA. The guarantee on the size of the states follows from Lemma 8. Finally, we cannot iterate forever, as each subprocedure decreases the weight of the solution at a state.

▶ **Lemma 9.** *There is constant $\kappa'' \geq 1$ (depending on $\mathfrak{R}$ but independent of n) such that for all states V, if $\|V\| \leq \kappa'' \cdot n$ and V has a solution (α, σ), then there exists a path to final state labeled with $h_1, h_2, \ldots, h_k$ such that*

$$\sigma = h_1 h_2 \cdots h_k.$$

References

1. Peter R.J. Asveld. Controlled iteration grammars and full hyper-AFL's. *Information and Control*, 34(3):248–269, 1977.
2. Jean Berstel and Dominique Perrin. The origins of combinatorics on words. *Eur. J. Comb.*, 28:996–1022, 2007.
3. Montserrat Casals and Ilya Kazachkov. On systems of equations over partially commutative groups. *Memoirs Amer. Math. Soc.*, 212:1–153, 2011.
4. Laura Ciobanu, Volker Diekert, and Murray Elder. Solution sets for equations over free groups are EDT0L languages. In *ICALP 2015, Proceedings*, volume 9135 of *LNCS*, pages 134–145. Springer, 2015.
5. Volker Diekert, Claudio Gutiérrez, and Christian Hagenah. The existential theory of equations with rational constraints in free groups is PSPACE-complete. *Information and Computation*, 202:105–140, 2005.
6. Volker Diekert, Artur Jeż, and Manfred Kufleitner. Solutions of word equations over partially commutative structures. *ArXiv e-prints*, 2016. `arXiv:1603.02966`.
7. Volker Diekert and Markus Lohrey. Word equations over graph products. *Int. J. Algebra Comput.*, 18:493–533, 2008.
8. Volker Diekert and Anca Muscholl. Solvability of equations in free partially commutative groups is decidable. *Int. J. Algebra Comput.*, 16:1047–1070, 2006.
9. Julien Ferté, Nathalie Marin, and Géraud Sénizergues. Word-mappings of level 2. *Theory Comput. Syst.*, 54:111–148, 2014.
10. Ju. I. Hmelevskiĭ. *Equations in Free Semigroups*. Number 107 in Proc. Steklov Institute of Mathematics. American Mathematical Society, 1976. Translated from the Russian original: Trudy Mat. Inst. Steklov. 107, 1971.
11. Sanjay Jain, Alexei Miasnikov, and Frank Stephan. The complexity of verbal languages over groups. In *LICS 2012, Proceedings*, pages 405–414. IEEE Computer Society, 2012.
12. Artur Jeż. Recompression: a simple and powerful technique for word equations. *J. ACM*, 63:1–51, 2016. Conference version at STACS 2013.
13. O. Kharlampovich and A. Myasnikov. Elementary theory of free non-abelian groups. *J. of Algebra*, 302:451–552, 2006.
14. Roger C. Lyndon and Marcel-Paul Schützenberger. The equation $a^M = b^N c^P$ in a free group. *Michigan Math. J.*, 9:289–298, 1962.
15. Gennadiĭ S. Makanin. The problem of solvability of equations in a free semigroup. *Math. Sbornik*, 103:147–236, 1977. English transl. in Math. USSR Sbornik 32 (1977).
16. Gennadiĭ S. Makanin. Equations in a free group. *Izv. Akad. Nauk SSR*, Ser. Math. 46:1199–1273, 1983. English transl. in Math. USSR Izv. 21 (1983).
17. Yuri Matiyasevich. Some decision problems for traces. In *LFCS 1997, Proceedings*, volume 1234 of *LNCS*, pages 248–257. Springer, 1997.
18. Antoni Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, Aarhus, 1977.
19. Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51:483–496, 2004.
20. Wojciech Plandowski. An efficient algorithm for solving word equations. In *STOC*, pages 467–476. ACM, 2006.

21 Wojciech Plandowski and Wojciech Rytter. Application of Lempel-Ziv encodings to the solution of word equations. In *ICALP 1998, Proceedings*, volume 1443 of *LNCS*, pages 731–742. Springer, 1998.

22 Alexander A. Razborov. *On Systems of Equations in Free Groups*. PhD thesis, Steklov Institute of Mathematics, 1987. In Russian.

23 Zlil Sela. Diophantine geometry over groups VIII: Stability. *Ann. of Math.*, 177:787–868, 2013.

24 Daniel Wise. *From Riches to Raags: 3-Manifolds, Right-Angled Artin Groups, and Cubical Geometry*. American Mathematical Society, 2012.

The Taming of the Semi-Linear Set*

Dmitry Chistikov[†1] **and Christoph Haase**[‡2]

1 **Max Planck Institute for Software Systems (MPI-SWS), Kaiserslautern and Saarbrücken, Germany**
dch@mpi-sws.org

2 **LSV, CNRS & ENS Cachan, Université Paris-Saclay, France**
haase@lsv.ens-cachan.fr

Abstract

Semi-linear sets, which are rational subsets of the monoid $(\mathbb{Z}^d, +)$, have numerous applications in theoretical computer science. Although semi-linear sets are usually given implicitly, by formulas in Presburger arithmetic or by other means, the effect of Boolean operations on semi-linear sets in terms of the size of description has primarily been studied for explicit representations. In this paper, we develop a framework suitable for implicitly presented semi-linear sets, in which the size of a semi-linear set is characterized by its norm – the maximal magnitude of a generator.

We put together a toolbox of operations and decompositions for semi-linear sets which gives bounds in terms of the norm (as opposed to just the bit-size of the description), a unified presentation, and simplified proofs. This toolbox, in particular, provides exponentially better bounds for the complement and set-theoretic difference. We also obtain bounds on unambiguous decompositions and, as an application of the toolbox, settle the complexity of the equivalence problem for exponent-sensitive commutative grammars.

1998 ACM Subject Classification G.2 Discrete Mathematics

Keywords and phrases semi-linear sets, convex polyhedra, triangulations, integer linear programming, commutative grammars

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.128

1 Introduction

Semi-linear sets [20] are a generalisation of ultimately periodic sets of natural numbers to any dimension d. By a classic result due to Ginsburg and Spanier [6], they coincide with the sets of integers[1] definable in Presburger arithmetic (the first-order theory of the integers with addition and order), and hence enjoy closure under all Boolean operations. Their nice properties make them a versatile tool in many application domains such as formal language theory, automata theory, and database theory.

More formally, semi-linear sets are finitely represented finite and infinite subsets of $\mathbb{Z}^d$. For $d \geq 1$, a *semi-linear set* M in dimension d is a finite union of *linear sets*. The latter are presented as a base vector $\boldsymbol{b} \in \mathbb{Z}^d$ and a finite set of period vectors $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\} \subseteq \mathbb{Z}^d$ and have the form

$$L(\boldsymbol{b}, P) := \boldsymbol{b} + \{\lambda_1 \cdot \boldsymbol{p}_1 + \cdots + \lambda_n \cdot \boldsymbol{p}_n : \lambda_1, \ldots, \lambda_n \in \mathbb{N}\}. \tag{1}$$

* This research was sponsored in part by the ERC Synergy award ImPACT.
† Present address: Department of Computer Science, University of Oxford, UK.
‡ Supported by Labex Digicosme, Univ. Paris-Saclay, project VERICONISS.
1 In the literature, semi-linear sets are often defined as subsets of $\mathbb{N}^d$ instead of $\mathbb{Z}^d$ as in this paper. All of our results do, however, carry over if one wishes to restrict semi-linear sets to $\mathbb{N}^d$.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 128; pp. 128:1–128:14


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

Such representations are, in fact, only rarely encountered in applications, because in many contexts semi-linear sets are defined implicitly. A semi-linear set can, for instance, be succinctly encoded by a formula in Presburger arithmetic; or a set can be just *proved* to be semi-linear with an estimation of its *norm*, $\|M\|$. The norm is the absolute value of the largest number occurring in the smallest description of M as a union of sets of the form (1). Examples of implicitly presented semi-linear sets include languages of various types of commutative grammars [10, 18] and reachability sets of reversal-bounded counter automata [13, 9].

The effect of Boolean operations is, however, not easy to track in terms of the size of vectors $\boldsymbol{b}$ and $\boldsymbol{p}_i$ if semi-linear sets are only presented implicitly. As an example, consider the set of non-negative integer solutions to a system of linear inequalities $\mathfrak{S}\colon A \cdot \boldsymbol{x} \leq \boldsymbol{c}$, which is a semi-linear set $S \subseteq \mathbb{N}^d$ encoded by $\mathfrak{S}$ with exponential succinctness. Huynh [12, 11] shows that for a given semi-linear set M, in general, whenever the complement $\overline{M} := \mathbb{N}^d \setminus M$ of M (with respect to $\mathbb{N}^d$; the same holds for $\mathbb{Z}^d$) is non-empty, then there is some $\boldsymbol{u} \in \overline{M}$ whose entries are bounded by an exponential in the explicit representation of M – which amounts to doubly exponential in the size of description of $\mathfrak{S}$. This upper bound is far from optimal: by Farkas' lemma, $\overline{M}$ contains an element $\boldsymbol{u}$ whose magnitude $\|\boldsymbol{u}\|$ is at most singly-exponential in the size of description of $\mathfrak{S}$.

Somewhat surprisingly, to the best of the authors' knowledge, there has been no unified framework for deriving bounds of this kind for implicitly presented semi-linear sets. Even if we take an explicitly given linear set as in (1) and describe it by an existential formula $\Psi(\boldsymbol{x})$ in Presburger arithmetic, the representation of the complement with a universally quantified formula $\neg\Psi(\boldsymbol{x})$ provides poor estimates on the magnitude of small elements: although upper bounds can be derived from an analysis of quantifier-elimination procedures, these bounds are only doubly exponential (see, e.g., [25]) and hence far from being optimal.

Our contribution

In this paper, we develop a framework suitable for implicitly presented semi-linear sets (explicitly presented sets are, of course, included as the simplest special case). In this framework the size of a semi-linear set $M \subseteq \mathbb{Z}^d$ is characterised by its norm, $\|M\|$, rather than the full bit-size of the description of M. We prove novel upper bounds in which, as a rule of thumb, the norm of the result of an operation is upper-bounded by $\|M\|^E$ where the exponent E behaves in a "controlled" way (say, $E = \mathsf{poly}(d)$), thus *taming* the effect of Boolean operations and decompositions. In more detail, our contributions are as follows:

- We put together a "toolbox" of operations and decompositions for semi-linear sets, with *tame* bounds, unified presentation, and simplified proofs. This toolbox includes improved bounds on the norm of the complement and, as a corollary, improved bounds on the norm of the set-theoretic difference. These bounds can give an exponential advantage over previously known techniques that upper-bound the bit-size of the result by n^E where n is the bit-size of the description of M – because n can be exponential in $\|M\|$.
- We derive from our toolbox an alternative proof of the Π_2^{P} upper bound for deciding semi-linear set inclusion, shown originally by Huynh [12, 11]. As a further application, we settle the complexity of the equivalence problem for exponent-sensitive commutative grammars, which have recently been introduced by Mayr and Weihmann [18].
- We give a new proof of and provide an explicit upper bound on unambiguous decomposition of semi-linear sets. It was first asked by Ginsburg [5] whether any semi-linear set is equivalent to a semi-linear set in which every element is generated in a unique way by exactly one linear set. This question was independently positively answered by Eilenberg

and Schützenberger [3] and by Ito [14]. However, to the best of our knowledge, no bounds on this decomposition have been established so far.

We now give a brief guide to the developed techniques and to the remainder of the paper. Our starting point is the fact that the set of non-negative solutions of a system of inequalities $\mathfrak{S}$ can be obtained as $L(B,P) := \bigcup_{\boldsymbol{b}\in B} L(\boldsymbol{b},P)$ for some finite sets $B, P \subseteq \mathbb{N}^d$. We call semi-linear sets of the form $L(B,P)$ *hybrid linear sets* and use them, instead of linear sets, as basic building blocks for general semi-linear sets. A hybrid linear set preserves more structural information about the "infinitary behaviour" of the linear sets it contains; it is, in fact, a discrete analogue of the Minkowski–Weyl representation of a convex polyhedron as the sum of a polytope and a convex cone.

Since the effect of operations on linear sets is primarily dominated by the magnitude and number of period vectors, reasoning in terms of hybrid linear sets lets us treat a potentially exponential number of linear sets in a uniform way. This, in turn, enables us, for instance, to obtain bounds on the representation of the intersection of two hybrid linear sets of the form $L(B,P)$ where, as one would indeed expect, the magnitude of the generators of the result does not depend on the cardinality of B (Subsection 2.3).

Our further path to the results on the complement and set-theoretic difference of semi-linear sets (Section 4) goes through another development, a *proper disjoint decomposition theorem*. It splits a hybrid linear set into a union $\bigcup_{i\in I} L(B_i,P_i)$ where each P_i is *proper* (i.e., consists of linearly independent vectors) and the *convex hulls* of $L(B_i,P_i)$ are disjoint (Section 3). For this result, we use the concept of a generalised simplex in order to construct triangulations of infinite polyhedra in $\mathbb{Q}^d$, and use the technique of half-open decompositions to ensure the disjointness of the aforementioned convex hulls.

Decomposing $\mathbb{Q}^d$ into convex polyhedra is by no means a new technique in the study of semi-linear sets. In particular, such decompositions were used by Huynh [12, 11] and recently by Kopczyński [15] in the context of semi-linear set inclusion. However, our decomposition theorem is different from theirs and gives stronger corollaries, in that we obtain a full semi-linear representation of the complement and, through intersection, of the set-theoretic difference. While the window theorem of Kopczyński in [15] gives an upper bound on the magnitude of the smallest vector in the set difference, our results upper-bound the magnitude of the largest generator.

2 Preliminaries

2.1 Basic definitions

Let $\mathbb{Z}$, $\mathbb{N}$, $\mathbb{Q}$, and $\mathbb{Q}_{\geq 0}$ denote the set of integers, non-negative integers, rationals, and non-negative rationals, respectively. For $x \in \mathbb{Q}$, $\lfloor x \rfloor$ is the largest integer that does not exceed x. For subsets of numbers or vectors A and B, we use the Minkowski sum notation: $A + B := \{a + b : a \in A, b \in B\}$. In this and other contexts, we often omit the curly braces when referring to singletons. For sets of vectors $P = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n\}, Q \subseteq \mathbb{Z}^m$, we may assume some fixed ordering on their elements, e.g., a lexicographic ordering, and thus sometimes treat P as a matrix whose column vectors are $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$. This leads to the notation $P \cdot \boldsymbol{\lambda}$ and $P \cdot Q$, for products of P with a vector $\boldsymbol{\lambda}$ and a matrix Q, respectively.

Linear, hybrid linear, and semi-linear sets

Suppose a natural $d \geq 1$ is fixed; we will call this d the *dimension*. A set $L \subseteq \mathbb{Z}^d$ is called *linear* if it is of the form

$$L = L(\boldsymbol{b}, P) := \{\boldsymbol{b} + \lambda_1 \boldsymbol{p}_1 + \cdots + \lambda_k \boldsymbol{p}_k : \lambda_1, \ldots, \lambda_k \in \mathbb{N}, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_k \in P\} \tag{2}$$

where $\boldsymbol{b} \in \mathbb{Z}^d$ and $P \subseteq \mathbb{Z}^d$ is a finite set. We call the vector $\boldsymbol{b}$ the *base vector* and vectors $\boldsymbol{p} \in P$ the *period vectors* (or simply *base* and *periods*) of L. A set $S \subseteq \mathbb{Z}^d$ is called *semi-linear* if it is a finite union of linear sets. Semi-linear sets can be represented as

$$S = \bigcup_{i \in I} L(B_i, P_i) \quad \text{where} \tag{3}$$

$$L(B_i, P_i) := \bigcup_{\boldsymbol{b}_i \in B_i} L(\boldsymbol{b}_i, P_i) \tag{4}$$

and $L(\boldsymbol{b}_i, P_i)$ is as in (2); we call sets $L(B_i, P_i)$ in (4) *hybrid linear sets*. Every linear set is also a hybrid linear set, and every hybrid linear set is semi-linear, but the converse statements are not true in general.

A hybrid linear set $L(B_i, P_i)$ is *proper* if the vectors P_i are linearly independent. Moreover, a hybrid linear set $L(B_i, P_i)$, $\#P_i = r$, is called *unambiguous* if for every $\boldsymbol{x} \in L(B_i, P_i)$ there exist a unique $\boldsymbol{b} \in B_i$ and a unique $\boldsymbol{\lambda} \in \mathbb{N}^r$ such that $\boldsymbol{x} = \boldsymbol{b} + P_i \cdot \boldsymbol{\lambda}$. A representation $\bigcup_{i \in I} L(B_i, P_i)$ is an *unambiguous decomposition* if each hybrid linear set $L(B_i, P_i)$ is unambiguous and the union is disjoint.

From the computational perspective, it is standard to represent semi-linear sets of the form (3) by listing all vectors in the sets B_i, P_i for all $i \in I$; components of the vectors are written in binary. We use the following notation to refer to various size measures for this representation. For any set A, the number of elements of A is $\#A$. For any $\boldsymbol{v} = (v_1, \ldots, v_d) \in \mathbb{Z}^d$, $\|v\| := \max_{1 \le i \le d} |v_i|$; similarly, for any $A \subseteq \mathbb{Z}^d$ we denote $\|A\| := \max_{\boldsymbol{v} \in A} \|\boldsymbol{v}\|$; observe that $\#A \le (2\|A\| + 1)^d$. Finally, for the representation (3) of a semi-linear set S we write $\|S\| := \max(\max_{i \in I} \|B_i\|, \max_{i \in I} \|P_i\|, 2)$, $\#_{\mathsf{b}} S := \max_{i \in I} \#B_i$, and $\#_{\mathsf{p}} S := \max_{i \in I} \#P_i$.

Convex polyhedra

We now introduce some terminology and notation from convex geometry (see, e.g., [22, 4, 19, 23]). For a system of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k \in \mathbb{Q}^d$, a linear combination $\lambda_1 \boldsymbol{v}_1 + \ldots + \lambda_k \boldsymbol{v}_k$ with $\lambda_1, \ldots, \lambda_k \in \mathbb{Q}$ is called: *non-negative*, or *conical*, if all $\lambda_i \ge 0$; *affine* if $\sum_{i=1}^k \lambda_i = 1$; and *convex* if it is non-negative and affine. For a possibly infinite set of vectors $A \subseteq \mathbb{Q}^d$, by $\operatorname{cone} A$, $\operatorname{aff} A$, and $\operatorname{conv} A$ we denote the *(rational) cone generated by* A, the *affine hull of* A, and the *convex hull of* A, respectively: they are the sets of all non-negative, affine, and convex combinations of finite subsets of A, respectively. We use the convention that $\boldsymbol{0} \in \operatorname{cone} A$ for any A; in particular, $\operatorname{cone} \emptyset = \{\boldsymbol{0}\}$. However, $\operatorname{conv} \emptyset = \emptyset$. Sets of the form $\boldsymbol{b} + \operatorname{cone} A$, for $\boldsymbol{b} \in \mathbb{Q}^d$, are *shifted cones*; we often refer to them simply as cones.

For any non-empty set $X \subseteq \mathbb{Q}^d$ its affine hull satisfies $\operatorname{aff} X = X_0 + \boldsymbol{v}$ for some vector $\boldsymbol{v} \in \mathbb{Q}^d$ and a uniquely determined linear subspace of $\mathbb{Q}^d$ denoted X_0. The *dimension* of X, written as $\dim X$, is the dimension of the subspace X_0.

A *(rational) convex polyhedron* in $\mathbb{Q}^d$ is a set of the form $\{\boldsymbol{x} \in \mathbb{Q}^d : A \cdot \boldsymbol{x} \le \boldsymbol{c}\}$ where $A \in \mathbb{Z}^{m \times d}$, $\boldsymbol{c} \in \mathbb{Z}^m$ for some m, and $\le$ is interpreted compontent-wise. A *face* of a convex polyhedron $W \subseteq \mathbb{Q}^d$ is a set of points where some linear function $\eta \colon \mathbb{Q}^d \to \mathbb{Q}$ achieves its maximum η^* over W; if η is non-constant, the hyperplane $h = \{\boldsymbol{x} \in \mathbb{Q}^d : \eta(\boldsymbol{x}) = \eta^*\}$ is a *supporting hyperplane* of W. A face of a convex polyhedron is always a convex polyhedron itself. Faces of dimension 0, 1, and $\dim W - 1$ are *vertices*, *edges*, and *facets* respectively. All faces of W form a partial order with respect to set inclusion, the largest element being the set W itself (it is always a face).

For a hybrid linear set $L(B, P)$, we denote $K(B, P) := \operatorname{conv} L(B, P) = \operatorname{conv} B + \operatorname{cone} P$. Note that if B is a singleton, i.e., if $L(B, P)$ is a linear set, then $K(B, P)$ is a rational cone; in general, though, $K(B, P)$ is a convex polyhedron.

Given a set S, we call its representation (3) a *proper disjoint decomposition* if each hybrid linear set $L(B_i, P_i)$ is proper and $K(B_i, P_i) \cap K(B_j, P_j) = \emptyset$ for $i \neq j$.

2.2 Auxiliary tools: Systems of linear inequalities

Let $A \in \mathbb{Z}^{m \times n}$ be an integer $m \times n$ matrix and $\boldsymbol{c} \in \mathbb{Z}^m$. We call $\mathfrak{S}\colon A \cdot \boldsymbol{x} \leq \boldsymbol{c}$ a *system of linear inequalities* and $\mathfrak{T}\colon A \cdot \boldsymbol{x} = \boldsymbol{c}$ a *system of linear equations*. By $[\![\mathfrak{S}]\!], [\![\mathfrak{T}]\!] \subseteq \mathbb{Z}^n$ we denote the *solution set* of $\mathfrak{S}$ and $\mathfrak{T}$, i.e, the set of all $\boldsymbol{v} \in \mathbb{Z}^n$ such that $A \cdot \boldsymbol{v} \leq \boldsymbol{c}$ and $A \cdot \boldsymbol{v} = \boldsymbol{c}$, respectively. We use $[\![\mathfrak{T}]\!]_{\geq 0}$ as a shorthand for $[\![\mathfrak{T}]\!] \cap \mathbb{N}^n$, and write $(\!|\mathfrak{S}|\!)$ for the set of rational solutions from $\mathbb{Q}^n$ of $\mathfrak{S}$. Moreover, we define $\|\mathfrak{S}\|, \|\mathfrak{T}\| := \max\{\|A\|, \|\boldsymbol{c}\|\}$.

The following two propositions connect two representations of polyhedra in $\mathbb{Q}^d$.

▶ **Proposition 1** ([23]). *Let $\mathfrak{S}\colon A \cdot \boldsymbol{x} \leq \boldsymbol{c}$ be a convex polyhedron in $\mathbb{Q}^d$. Then there are $B \subseteq \mathbb{Q}^d$ and $P \subseteq \mathbb{Z}^d$ such that $(\!|\mathfrak{S}|\!) = \operatorname{conv} B + \operatorname{cone} P$, $\|P\| \leq 2^{O(d \log d)} \cdot \|\mathfrak{S}\|^d$, and all numerators and denominators in B are bounded by $2^{O(d \log d)} \cdot \|\mathfrak{S}\|^d$.*

Proof. By the Minkowski–Weyl theorem, there exist $C, Q \subseteq \mathbb{Q}^d$ such that $(\!|\mathfrak{S}|\!) = \operatorname{conv} C + \operatorname{cone} Q$. In fact, it is possible (cf. [23, Theorem 10.2]) to find C and Q in which all vectors have numerators and denominators of all entries bounded by $d! \cdot \|\mathfrak{S}\|^d$: they can essentially be chosen as solutions to linear systems defined by square submatrices of the matrix $\left[\begin{array}{c|c} A & \boldsymbol{c} \end{array}\right]$. ◀

▶ **Proposition 2** ([23]). *Let $M = L(\boldsymbol{b}, P) \subseteq \mathbb{Z}^d$ be a proper linear set with $r = \#P$. Then there exists a system of linear inequalities $\mathfrak{S}\colon A \cdot \boldsymbol{x} \leq \boldsymbol{c}$ such that*

- *A is a $(2d - r) \times d$ matrix that does not depend on $\boldsymbol{b}$;*
- *$\|A\| \leq 2^{O(r \log r)} \cdot \max(\|P\|, 1)^r$; $\|\boldsymbol{c}\| \leq d \cdot \|A\| \cdot \|\boldsymbol{b}\|$; and*
- *$\operatorname{conv} L(\boldsymbol{b}, P) = (\!|\mathfrak{S}|\!)$.*

Proof (sketch). Since M is proper, $\operatorname{conv} L(\boldsymbol{b}, P)$ has exactly r facets; r inequalities in $\mathfrak{S}$ define them, with another $2(d - r)$ for $\operatorname{aff} M$. It can be shown (cf. [23, Theorem 10.2]) that there exists an appropriate A such that $(\!|A \cdot \boldsymbol{x} \leq \boldsymbol{0}|\!) = \operatorname{conv} L(\boldsymbol{0}, P)$; and then $\boldsymbol{c} = A \cdot \boldsymbol{b}$. ◀

We will also need a result of von zur Gathen and Sieveking on the sets of all integer solutions of systems of linear inequalities [24].

▶ **Proposition 3.** *Let $\mathfrak{S}\colon A \cdot \boldsymbol{x} \leq \boldsymbol{c}$ be a system of inequalities such that $A \in \mathbb{Z}^{m \times n}$. Then $[\![\mathfrak{S}]\!] = \bigcup_{i \in I} L(B_i, P_i)$ such that*

- *$K(B_i, P_i) \cap K(B_j, P_j) = \emptyset$ for all $i \neq j$,*
- *$\max_{i \in I} \|B_i\|, \max_{i \in I} \|P_i\| \leq 2^{O(n \log n)} \cdot \|A\|^{n-1} \cdot \|\mathfrak{S}\|$, and*
- *$\#I \leq 2^n$.*

Next, we additionally recall a result on the sets of integer solutions of linear equalities that follows from results of Pottier [21].

▶ **Proposition 4.** *Let $\mathfrak{S}_0\colon A \cdot \boldsymbol{x} = \boldsymbol{0}$ and $\mathfrak{S}\colon A \cdot \boldsymbol{x} = \boldsymbol{c}$ be systems of linear Diophantine equations, where $A \in \mathbb{Z}^{m \times n}$. Then $[\![\mathfrak{S}_0]\!]_{\geq 0} = L(\boldsymbol{0}, P)$ and $[\![\mathfrak{S}]\!]_{\geq 0} = L(B, P)$ such that*

- *$\|B\| \leq ((n+1) \cdot \|A\| + \|\boldsymbol{c}\| + 1)^m$, $\|P\| \leq (n \cdot \|A\| + 1)^m$.*

Finally, we will need a discrete version of Carathéodory's theorem:

▶ **Proposition 5.** *Let $M = L(C, Q) \subseteq \mathbb{Z}^d$ be a hybrid linear set. Then $M = \bigcup_{i \in I} L(B_i, P_i)$ such that*

- *$\max_{i \in I} \|B_i\| \leq \|C\| + (\#Q \cdot \|Q\|)^{O(d)}$,*

- $\max_{i\in I} \#P_i \leq d$, $P_i \subseteq Q$ *and each* P_i *is linearly independent, and*
- $\#I \leq (\#Q)^d$.

The statement of Proposition 5 can essentially be shown by a combination of Lemmas 2.7 and 2.8 in [10], which, however, do not establish any concrete bounds. In our proof, we use the result on the intersection of hybrid linear sets from the following subsection 2.3.

2.3 Intersection of semi-linear sets

▶ **Theorem 6.** *Let* M *and* N *be semi-linear sets with representations* $M = \bigcup_{j\in J} L(C_j, Q_j)$, $N = \bigcup_{k\in K} L(D_k, R_k)$. *Then the set* $L := M \cap N$ *is a semi-linear set with representation* $L = \bigcup_{i\in I} L(B_i, P_i)$ *such that* $I = J \times K$,

- $\max_{i\in I} \|B_i\|, \max_{i\in I} \|P_i\| \leq ((\#_{\mathsf{p}} M + \#_{\mathsf{p}} N) \cdot \max(\|M\|, \|N\|))^{O(d)}$, *and*
- $\#I \leq \#J \cdot \#K$.

Moreover, if $Q_j \subseteq R_k$ *and* $i = (j, k)$ *then* $P_i = Q_j$.

Proof (sketch). We have $M \cap N = \bigcup_{j\in J} L(C_j, Q_j) \cap \bigcup_{k\in K} L(D_k, R_k) = \bigcup_{j\in J, k\in K} L(C_j, Q_j) \cap L(D_k, R_k)$. Hence it suffices to show that every $L(C_j, Q_j) \cap L(D_k, R_k)$ is some $L(B_{j,k}, P_{j,k})$ with the desired properties. To this end, one can obtain the set of elements in the intersection as the set of solutions to a suitable system of linear equations and then apply the bounds from Proposition 4. Finally, the fact that if $Q_j \subseteq R_k$ then $P_i = Q_j$ follows from Theorem 5.6.1 in [5, p. 180]. ◀

3 Hybrid linear sets

In the sequel, we develop a close connection between hybrid linear sets and convex polyhedra viewed as generalized convex hulls. Convex polyhedra in $\mathbb{Q}^d$ are sets of the form $\operatorname{conv} C + \operatorname{cone} Q$ for $C, Q \subseteq \mathbb{Q}^d$; they can be viewed as a convex hulls of a set of points C and *directions* Q. Suppose $C = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_r\}$ and $Q = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}$. The connection builds upon on the similarity of the following sets:

$$\operatorname{conv} C + \operatorname{cone} Q = \left\{ \sum_{i=1}^{r} \lambda_i \boldsymbol{b}_i + \sum_{j=1}^{m} \mu_j \boldsymbol{p}_j : \lambda_i \in \mathbb{Q}_{\geq 0},\ \sum_{i=1}^{r} \lambda_i = 1,\ \mu_j \in \mathbb{Q}_{\geq 0} \right\} \text{ and}$$

$$L(C, Q) = \left\{ \sum_{i=1}^{r} \lambda_i \boldsymbol{b}_i + \sum_{j=1}^{m} \mu_j \boldsymbol{p}_j : \lambda_i \in \mathbb{N},\ \sum_{i=1}^{r} \lambda_i = 1,\ \mu_j \in \mathbb{N} \right\}.$$

As mentioned above, $\operatorname{conv} L(C, Q) = K(C, Q) = \operatorname{conv} C + \operatorname{cone} Q$.

3.1 Proper disjoint decompositions (PDD)

Recall that $S = \bigcup_{i\in I} L(B_i, P_i)$ is a proper disjoint decomposition if vectors in each P_i are linearly independent and the convex hulls $K(B_i, P_i) = \operatorname{conv} L(B_i, P_i)$ are pairwise disjoint.

▶ **Theorem 7** (PDD for hybrid linear sets)**.** *Every hybrid linear set* $M = L(C, Q)$ *has a proper disjoint decomposition* $\bigcup_{i\in I} L(B_i, P_i)$ *where each* P_i *is a subset of* Q *and the following inequalities hold:*

- $\|B_i\| \leq (\#Q + \|C\| + \|Q\| + d)^{O(d)} \leq \|M\|^{O(d^2)}$ *and*

- $\#I \leq (\#Q)^{d+1}$.

The idea of the proof of Theorem 7 is to rely on the connection between hybrid linear sets and convex polyhedra. We will use the observation that each set of the form $\operatorname{conv} C + \operatorname{cone} Q$ has a triangulation. While this term usually refers to the basic construction that splits a convex polygon in a plane into a number of non-overlapping triangles, we will use a construction that extends this concept in two ways: first, instead of $\mathbb{Q}^2$ the sets are in $\mathbb{Q}^d$, so triangles become simplices; second, the sets can be infinite, i.e., with $Q \neq \emptyset$.

The strategy of the proof of Theorem 7 is depicted in the following diagram:

$$\begin{array}{ccc} L(C,Q) & \xrightarrow{3} & \Pi, \text{ a proper disjoint decomposition of } L(C,Q) \\ \big\downarrow 1 & & \big\uparrow 3 \\ K(C,Q) & \xrightarrow{2} & \mathcal{T}, \text{ a triangulation of } K(C,Q) \end{array}$$

Step 1 is taking the convex hull, step 2 is triangulation in $\mathbb{Q}^d$, and step 3 constructs a proper disjoint decomposition given the original set $L(C,Q)$ and the triangulation of $K(C,Q)$.

A *generalized δ-dimensional simplex* T is a set of the form $T = \operatorname{conv} V + \operatorname{cone} D \subseteq \mathbb{Q}^d$ where $\#V + \#D = \delta + 1$, $V \neq \emptyset$, and the dimension of the affine hull of T is exactly δ (cf. [22, pp. 153f]). Elements of V are ordinary vertices of T, and elements of D are vertices at infinity and can be understood as directions. (The set D is, in fact, the set of *extreme directions* of the set T; see [22, p. 162].) Faces of generalized simplices $\operatorname{conv} V + \operatorname{cone} D$ are also generalized simplices and have the form $\operatorname{conv} V' + \operatorname{cone} D'$ where $V' \subseteq V$ and $D' \subseteq D$.

A *triangulation* of a set $W \subseteq \mathbb{Q}^d$ is a collection $\mathcal{T}$ of generalized simplices that satisfies the following properties:

1. $\bigcup_{F \in \mathcal{T}} F = W$.
2. For every $F \in \mathcal{T}$ and every face F' of F, it holds that $F' \in \mathcal{T}$.
3. The intersection of any two $F_1, F_2 \in \mathcal{T}$ is either empty or is a face of both F_1 and F_2.
4. All (generalized) simplices in the set of maxima of $\mathcal{T}$, denoted $\operatorname{Max} \mathcal{T} := \{F' \in \mathcal{T} : \nexists F \in \mathcal{T}.\ F' \text{ is a face of } F \text{ and } F \neq F'\}$, have the same dimension δ, denoted $\dim \mathcal{T}$.

In other words, a triangulation of W is a pure polyhedral complex (see, e.g., [4, Chapter 6]) that consists of generalized simplices and covers exactly W.

To simplify notation, we write $\mathcal{T} = (T_1, \ldots, T_m)$ whenever $\operatorname{Max} \mathcal{T} = \{T_1, \ldots, T_m\}$; of course, the set $\{T_1, \ldots, T_m\}$ is a subset of the set $\mathcal{T}$. It is straightforward that $W = T_1 \cup \ldots \cup T_m$ if $\mathcal{T} = (T_1, \ldots, T_m)$ is a triangulation of W. Conversely, if $T_1, \ldots, T_m$ are (generalized) simplices of equal dimension such that the collection $\mathcal{T}$ of all their faces satisfies Condition 3 in the definition of triangulation, then this collection $\mathcal{T}$ is a triangulation of $T_1 \cup \ldots \cup T_m$. Lemma 8 triangulates possibly unbounded convex polyhedra (for non-empty Q, it treats its elements as vertices at infinity) without introducing new vertices or directions.

▶ **Lemma 8.** *Every polyhedron of the form $W = \operatorname{conv} C + \operatorname{cone} Q \subseteq \mathbb{Q}^d$ has a triangulation $\mathcal{T} = (T_1, \ldots, T_m)$ where $m \leq (\#C + \#Q)^{d+1}$ and $T_i = \operatorname{conv} C_i + \operatorname{cone} Q_i$ with $C_i \subseteq C$ and $Q_i \subseteq Q$ for all i.*

Note that adjacent simplices T_i and T_j in a triangulation can share points in common lower-dimensional faces. However, for our purposes they should be made disjoint. Suppose U is a polyhedron of the form $X = \{\boldsymbol{x} \in \mathbb{Q}^d : \boldsymbol{a}_i \cdot \boldsymbol{x} \leq c_i, 1 \leq i \leq m\}$ where $\boldsymbol{a}_i \in \mathbb{Z}^d$ and $c_i \in \mathbb{Z}$ for all i. For any $A \subseteq \{1, \ldots, m\}$, we call the set

$$X_A = \{\boldsymbol{x} \in \mathbb{Q}^d : \boldsymbol{a}_i \cdot \boldsymbol{x} < c_i, i \in A, \text{ and } \boldsymbol{a}_i \cdot \boldsymbol{x} \leq c_i, i \in \{1, \ldots, m\} \setminus A\}$$

a *half-opening* of U obtained by cutting off the hyperplanes $\boldsymbol{a}_i \cdot \boldsymbol{x} = c_i$, $i \in A$.

▶ **Lemma 9.** *Let W be a δ-dimensional polyhedron in $\mathbb{Q}^d$. For each triangulation $\mathcal{T} = (T_1, \ldots, T_m)$ of W there exists a collection of sets $\mathcal{T}^0 = (T_1^0, \ldots, T_m^0) \subseteq \mathbb{Q}^d$ that satisfies the following conditions:*
1. *$T_1^0 \cup \ldots \cup T_m^0 = W$.*
2. *For every i, T_i^0 is a half-opening of T_i.*
3. *T_i and T_j are disjoint for $i \neq j$.*

Lemma 9 is the *half-open decomposition*, originally from [1] and [16]. Our formulation is a direct corollary of Theorem 3 in the latter paper; see also [8, Section 3.2].

▶ **Lemma 10.** *Suppose $T = \operatorname{conv} V + \operatorname{cone} D$ is a generalized δ-dimensional simplex in $\mathbb{Q}^d$ where $V, D \subseteq \mathbb{Z}^d$ and $\#V + \#D = \delta + 1$. Then for any half-opening T^0 of T it holds that $T^0 \cap \mathbb{Z}^d = L(E, D)$ where $\|E\| \leq \|V\| + (d+1) \cdot \|D\|$.*

Lemma 10 makes it possible to use half-open decomposition in the proof of Theorem 7.

Proof of Theorem 7 (sketch). Take a triangulation of $W = K(C, Q) = \operatorname{conv} C + \operatorname{cone} Q$, which exists by Lemma 8, and apply Lemma 9 to this triangulation. The result is a collection $\mathcal{T}^0 = (T_1^0, \ldots, T_m^0)$ where each T_i^0 is a half-opening of some generalized simplex $\operatorname{conv} C_i + \operatorname{cone} Q_i$ such that $C_i \subseteq C$ and $Q_i \subseteq Q$. By Lemma 10, $T_i^0 \cap \mathbb{Z}^d = L(D_i, Q_i)$. We now apply Theorem 6: since $Q_i \subseteq Q$, we have $L(D_i, Q_i) \cap L(C, Q) = L(B_i, P_i)$ where $P_i = Q_i$. Vectors in each set $P_i = Q_i$ are, in fact, linearly independent, because $\operatorname{conv} C_i + \operatorname{cone} Q_i$ is a generalized simplex. Moreover, $K(B_i, P_i) \subseteq \operatorname{conv} L(D_i, Q_i) \subseteq T_i^0$ for each i; since the sets $T_1^0, \ldots, T_m^0$ are pairwise disjoint, so are the sets $K(B_i, P_i)$. Finally,

$$\begin{aligned}\bigcup_{i=1}^{m} L(B_i, P_i) &= \bigcup_{i=1}^{m} T_i^0 \cap \mathbb{Z}^d \cap L(C, Q) = L(C, Q) \cap \bigcup_{i=1}^{m} T_i^0 \\ &= L(C, Q) \cap W = L(C, Q) \cap \operatorname{conv} L(C, Q) = L(C, Q).\end{aligned}$$ ◀

3.2 Unambiguous decompositions (UD)

The main results of this subsection are the following theorems:

▶ **Theorem 11** (UD for proper hybrid linear sets)**.** *Every proper hybrid linear set $M = L(C, Q)$ has an unambiguous decomposition $\bigcup_{i \in I} L(B_i, P_i)$ where each P_i is a subset of Q and the following conditions are satisfied:*
- *$\|B_i\| \leq \|C\|$ and*
- *$\#I \leq (2 \cdot \#C)^{\#Q}$.*

▶ **Theorem 12** (UD for hybrid linear sets)**.** *Every hybrid linear set $M = L(C, Q)$ has an unambiguous decomposition $\bigcup_{i \in I} L(B_i, P_i)$ where each P_i is a subset of Q and the following inequalities hold:*
- *$\|B_i\| \leq (\#Q + \|C\| + \|Q\| + d)^{O(d)} \leq \|M\|^{O(d^2)}$ and*
- *$\#I \leq ((\|C\| + \|Q\| + d)^{O(d)} + \#C)^d \cdot (d + \#Q)^{O(d^2)} \leq \|M\|^{O(d^3)}$.*

We now show how to prove Theorem 11. The idea is to reduce the disambiguation of a proper hybrid linear set to disambiguation of an ideal in a finitely generated commutative monoid, captured by the following lemma. Here and below, by $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_r$ we denote coordinate vectors in $\mathbb{N}^r$.

▶ **Lemma 13.** *Every set of the form $U = L(F, \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_r\}) = F + \mathbb{N}^r$ with a finite $F \subseteq \mathbb{N}^r$ has a representation $U = \bigcup_{k \in K} L(G_k, E_k)$ such that the following conditions are satisfied:*

- *each set* $L(G_k, E_k)$ *is unambiguous,*
- *the polyhedra* $\operatorname{conv} L(G_k, E_k)$ *and* $\operatorname{conv} L(G_{k'}, E_{k'})$ *are disjoint for* $k \neq k'$*,*
- $\|G_k\| \leq \|F\|$*,*
- $E_k \subseteq \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_r\}$*, and*
- $\#K \leq (\#F + 1)^r$*.*

Proof (sketch). The condition that a vector $\boldsymbol{x}$ belongs to $F + \mathbb{N}^r$ can be specified by a logical formula Φ over predicates of the form $x_j \geq c$. These predicates break up $\mathbb{N}^r$ into at most $(\#F + 1)^r$ disjoint regions, and each region is described by a unambiguous hybrid linear set in a straightforward way. ◀

Proof of Theorem 11 (sketch). Take $M = L(C, Q) \subseteq \mathbb{Z}^d$ where $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_r\} \subseteq \mathbb{Z}^d$ and vectors in Q are linearly independent, $r \leq d$. Consider the *point lattice* $\mathcal{L} = Q \cdot \mathbb{Z}^r = \{Q \cdot \boldsymbol{\lambda} : \boldsymbol{\lambda} \in \mathbb{Z}^r\}$; see, e.g., [17, Chapter 2]. Vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^r$ are *congruent* modulo $\mathcal{L}$, written $\boldsymbol{x} \equiv \boldsymbol{y} \pmod{\mathcal{L}}$, if and only if $\boldsymbol{x} - \boldsymbol{y} \in \mathcal{L}$. This congruence splits the set C into a disjoint union $C = C_1 \cup \ldots \cup C_s$ where $\boldsymbol{x} \in C_i$ and $\boldsymbol{y} \in C_j$ are congruent if and only if $i = j$. It is easy to see that $M = \bigcup_{1 \leq j \leq s} L(C_j, Q)$ is a disjoint union, and disambiguating each $L(C_j, Q)$ separately will disambiguate M.

Suppose $C_1 = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\} \subseteq \boldsymbol{x}_1 + \mathcal{L}$. Since the vectors in $Q = \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_r\}$ are linearly independent, each vector from the set $\boldsymbol{x}_1 + \mathcal{L}$ has a unique expansion of the form $\boldsymbol{x}_1 + \sum_{j=1}^r a_j \boldsymbol{q}_j$. Consider the mapping $\psi\colon \boldsymbol{x}_1 + \mathcal{L} \to \mathbb{Z}^r$ taking each vector $\boldsymbol{x}_1 + \sum_{j=1}^r a_j \boldsymbol{q}_j$ to the vector $(a_1, \ldots, a_r) \in \mathbb{Z}^r$. For each j, let a_j^0 be the smallest of the numbers $\psi(\boldsymbol{x}_t)[j]$ over $1 \leq t \leq m$; here $[j]$ refers to the jth component of an r-dimensional vector. Denote $\boldsymbol{a}^0 = (a_1^0, \ldots, a_r^0)$ and let $\psi'\colon \boldsymbol{x}_1 + \mathcal{L} \to \mathbb{Z}^r$ be given by $\psi'(\boldsymbol{x}) = \psi(\boldsymbol{x}) - \boldsymbol{a}^0$. Observe that the mapping ψ' is injective and maps C_1 to some finite set $F \subseteq \mathbb{N}^r$; in fact, $\psi'(L(C_1, Q_1)) = F + \mathbb{N}^r$. After this, it remains to apply Lemma 13. ◀

4 Semi-linear sets

In this section, we derive our main results on the complement, set-theoretic difference, and decompositions of semi-linear sets. We will rely on Theorems 7 and 11 from Section 3.

4.1 Geometric ingredients: Splitting into atomic polyhedra

Consider a semi-linear set given by $M = \bigcup_{j \in J} L(C_j, Q_j)$. Take the proper disjoint decomposition of each $L(C_j, Q_j)$ according to Theorem 7; this decomposes M as

$$M = \bigcup_{j \in J} \bigcup_{t \in T_j} L(C_{jt}, Q_{jt}) \tag{5}$$

where hybrid linear sets $L(C_{jt}, Q_{jt})$ are proper, $Q_{jt} \subseteq Q_j$, and, moreover, for any fixed j the polyhedra $K(C_{jt}, Q_{jt})$ are pairwise disjoint.

Denote by $\mathcal{H}$ the collection of principal supporting hyperplanes for shifted cones $K(\boldsymbol{b}, Q_{jt})$, $\boldsymbol{b} \in C_{jt}$, $t \in T_j$, and $j \in J$: for each cone, take its d principal supporting hyperplanes, i.e., the hyperplanes obtained in Proposition 2, each of the form $h\colon \boldsymbol{a} \cdot \boldsymbol{x} = c$ (with fixed $\boldsymbol{a} \in \mathbb{Z}^d$ and $c \in \mathbb{Z}$), and put them into $\mathcal{H}$. Note that each hyperplane h' is associated with half-spaces $h^-\colon \boldsymbol{a} \cdot \boldsymbol{x} \leq c$ and $h^+\colon \boldsymbol{a} \cdot \boldsymbol{x} \geq c+1$; moreover, we can pick the signs so that $K(\boldsymbol{b}, Q_{jt}) \subseteq (\!|h^-|\!)$. An *atomic polyhedron* with respect to $\mathcal{H}$ is a non-empty set of the form

$$A(H) = \bigcap_{h \in H} (\!|h^-|\!) \cap \bigcap_{h \in \mathcal{H} \setminus H} (\!|h^+|\!),$$

where $H \subseteq \mathcal{H}$. Clearly, $\mathbb{Z}^d \subseteq \bigcup_{H \subseteq \mathcal{H}} A(H)$, and $A(H) \cap A(H') = \emptyset$ whenever $H \neq H'$.

▶ **Lemma 14.** *For every $L(\boldsymbol{b}, Q_{jt})$ with $\boldsymbol{b} \in C_{jt}$ and every $A = A(H)$, either $A \subseteq \operatorname{conv} L(\boldsymbol{b}, Q_{jt})$ or $A \cap \operatorname{conv} L(\boldsymbol{b}, Q_{jt}) = \emptyset$.*

Take a hybrid linear set $L(C_{jt}, Q_{jt})$ and let $\boldsymbol{b} \in C_{jt}$. We say that the linear set $L(\boldsymbol{b}, Q_{jt})$ *shares* an atomic polyhedron A iff $A \subseteq \operatorname{conv} L(\boldsymbol{b}, Q_{jt})$; otherwise we say that it *avoids* A.

▶ **Lemma 15.** *Every atomic polyhedron $A(H)$ is the set of rational solutions to a system of at most $O(d \cdot \sum_{j\in J}(\#Q_j)^{d+1})$ linear inequalities with entries bounded by $\|M\|^{O(d^2)}$.*

▶ **Lemma 16.** *The number of atomic polyhedra is at most $\left(d \cdot \sum_{j\in J} \#C_j \cdot (\#Q_j)^{d+1}\right)^{d+1}$.*

Consider an atomic polyhedron A; we will assume in the remainder of this subsection that A is shared by at least one linear set of the form $L(\boldsymbol{b}, Q_{jt})$. Even though the total number of linear sets of this form that share A can be large, the following property holds.

▶ **Lemma 17.** *If linear sets $L(\boldsymbol{b}, Q_{jt})$ and $L(\boldsymbol{b}', Q_{jt'})$ share A, then $t = t'$. In particular, the number of pairs (j, t) such that some linear set $L(\boldsymbol{b}, Q_{jt})$ shares A does not exceed $\#J$.*

▶ **Lemma 18.** *For every A there exist finite sets $E \subseteq \mathbb{Q}^d$ and $G \subseteq \mathbb{Z}^d$ that satisfy the following conditions:*

1. *$A = \operatorname{conv} E + \operatorname{cone} G$.*
2. *For every linear set $L(\boldsymbol{b}, Q_{jt})$ that shares A, the set G is a subset of $L(\boldsymbol{0}, Q_{jt})$.*
3. *Numerators and denominators of all entries in all $\boldsymbol{e} \in E$ are bounded by $\|M\|^{O(d^3)}$.*
4. *$\|G\| \leq \|M\|^{\#J \cdot O(d^4)}$.*
5. *$\#G \leq \|M\|^{O(d^4)}$.*

The proof of Lemma 18 first applies Proposition 1 to the representation of A from Lemma 15. The upper bound on $\|G\|$ then relies on the fact that, for every $j \in J$, our decomposition (5) ensures disjointness of $K(C_{jt}, Q_{jt})$ among $t \in T_j$; the proof uses this property via Lemma 17.

4.2 Decompositions, complement, and difference

We first state the results on decompositions of semi-linear sets and on the semi-linear representation of the complement.

▶ **Theorem 19** (PDD for semi-linear sets)**.** *Every semi-linear set $M = \bigcup_{j\in J} L(C_j, Q_j)$ has a proper disjoint decomposition $\bigcup_{i\in I} L(B_i, P_i)$ where*

- *$\|B_i\| \leq \|M\|^{\#J \cdot O(d^6)}$,*
- *$\|P_i\| \leq \|M\|^{\#J \cdot O(d^4)}$, and*
- *$\#I \leq \|M\|^{O(d^5)}$.*

▶ **Corollary 20** (UD for semi-linear sets)**.** *Every semi-linear set $M = \bigcup_{j\in J} L(C_j, Q_j)$ has an unambiguous decomposition $\bigcup_{i\in I} L(B_i, P_i)$ where*

- *$\|B_i\| \leq \|M\|^{\#J \cdot O(d^6)}$ and*
- *$\|P_i\| \leq \|M\|^{\#J \cdot O(d^4)}$.*

▶ **Theorem 21** (complement of semi-linear sets)**.** *The complement of every semi-linear set $M = \bigcup_{j\in J} L(C_j, Q_j)$ has a representation of the form $\bigcup_{i\in I} L(B_i, P_i)$ where*

- *$\|B_i\| \leq \|M\|^{\#J \cdot O(d^4)}$ and*
- *$\|P_i\| \leq \|M\|^{\#J \cdot O(d^4)}$.*

We will state the results on set difference at the end of this subsection, and now we focus our attention on Theorems 19 and 21. Corollary 20 follows from Theorems 19 and 11.

Recall that in Subsection 4.1 we decomposed the space into disjoint atomic polyhedra A. Each $A = \operatorname{conv} E + \operatorname{cone} G$ by Lemma 18, with $E \subseteq \mathbb{Q}^d$ and $G \subseteq \mathbb{Z}^d$. By Carathéodory's theorem, for every vector $\boldsymbol{x} \in A$ there are $\nu_{\boldsymbol{e}} \in \mathbb{N}$, $\boldsymbol{e} \in E$, and $\mu_{\boldsymbol{g}} \in \mathbb{N}$, $\boldsymbol{g} \in G$, such that $\boldsymbol{x}$ has an expansion of the form

$$\boldsymbol{x} = \sum_{\boldsymbol{e} \in E} \nu_{\boldsymbol{e}} \cdot \boldsymbol{e} + \sum_{\boldsymbol{g} \in G'} \mu_{\boldsymbol{g}} \cdot \boldsymbol{g} = \tau(\boldsymbol{x}) + \pi(\boldsymbol{x}), \tag{6}$$

where $\tau(\boldsymbol{x}) = \sum_{\boldsymbol{e} \in E} \nu_{\boldsymbol{e}} \cdot \boldsymbol{e} + \sum_{\boldsymbol{g} \in G'} (\mu_{\boldsymbol{g}} - \lfloor \mu_{\boldsymbol{g}} \rfloor) \cdot \boldsymbol{g}$ denotes the *truncation* of $\boldsymbol{x}$, $\pi(\boldsymbol{x}) = \sum_{\boldsymbol{g} \in G'} \lfloor \mu_{\boldsymbol{g}} \rfloor \cdot \boldsymbol{g}$ denotes the *periodic part* of $\boldsymbol{x}$, and $G' \subseteq G$ is some subset of linearly independent vectors in G. We will consider sets $X = A \cap \mathbb{Z}^d \setminus M$ and $Y = A \cap \mathbb{Z}^d \cap M = A \cap M$.

It is not difficult to show that $\|\tau(X)\|$ and $\|\tau(Y)\|$ are bounded from above by $\|M\|^{\#J \cdot \mathsf{poly}(d)}$; these estimations are relevant, as we prove that the equalities $X = L(\tau(X), G)$ and $Y = L(\tau(Y), G)$ hold. While the latter equality requires no sophisticated arguments, a proof of the former turns out to be somewhat delicate. As an auxiliary statement, we show that $\tau(X) \subseteq X$; with this fact at hand, the proof of the inclusion $L(\tau(X), G) \subseteq X$ goes as follows. Suppose, for the sake of contradiction, that there exists a vector $\boldsymbol{z} \in L(\tau(X), G) \cap M$, say with $\boldsymbol{z} \in L(\boldsymbol{b}, Q_{jt})$ such that $L(\boldsymbol{b}, Q_{jt})$ shares A. This implies the existence of another vector $\boldsymbol{x} \in X$ with $\tau(\boldsymbol{x}) \in \boldsymbol{b} + Q_{jt} \cdot \mathbb{Z}^\delta$ where δ is the cardinality of Q_{jt}. At the same time, this $\tau(\boldsymbol{x})$ also belongs to X and thus to A and to the cone $K(\boldsymbol{b}, Q_{jt}) = \boldsymbol{b} + Q_{jt} \cdot \mathbb{Q}^\delta_{\geq 0}$. Since the vectors in Q_{jt} are linearly independent (recall that sets Q_{jt} come from a *proper* disjoint decomposition of $L(C_j, Q_j)$), it follows that $\tau(\boldsymbol{x}) \in \boldsymbol{b} + Q_{jt} \cdot \mathbb{N}^\delta = L(\boldsymbol{b}, Q_{jt})$, which contradicts the fact that $\tau(X) \subseteq X$, because X excludes M.

As seen from this sketch, our ability to construct the hybrid linear representation of X (which corresponds to the complement of M) relies on the fact that our decomposition of M in (5) uses linear sets with linearly independent periods only.

Proofs of Theorems 19 and 21 (sketch). Use equalities

$$M = \bigcup_{H \subseteq \mathcal{H}} A(H) \cap M \qquad \text{and} \qquad \mathbb{Z}^d \setminus M = \bigcup_{H \subseteq \mathcal{H}} A(H) \cap \mathbb{Z}^d \setminus M$$

where it suffices to consider only (non-empty) atomic polyhedra $A = A(H)$. Whenever all linear sets $L(\boldsymbol{b}, Q_{jt})$, $\boldsymbol{b} \in C_{jt}$ (see (5)) avoid a polyhedron A, we have $Y = A \cap M = \emptyset$ and $X = A \cap \mathbb{Z}^d \setminus M = A \cap \mathbb{Z}^d$. Here the case of Y is trivial, and the case of X sends us to Proposition 3. Otherwise, if at least one linear set shares A, we use the representations $X = L(\tau(X), G)$ and $Y = L(\tau(Y), G)$ as discussed above. For the purposes of proper disjoint decomposition (Theorem 19), we need to invoke Theorem 7 on $L(\tau(Y), G)$. Upper bounds on $\|B_i\|$, $\|P_i\|$, and $\#I$ follow from Lemmas 18 and 16 and from Theorem 7. ◀

▶ **Corollary 22** (difference of semi-linear sets). *The set-theoretic difference $M \setminus N$ of semi-linear sets $M = \bigcup_{j \in J} L(C_j, Q_j)$ and $N = \bigcup_{k \in K} L(D_k, R_k)$ has a representation of the form $L = \bigcup_{i \in I} L(B_i, P_i)$, where*

- $\max_{i \in I} \|B_i\|, \max_{i \in I} \|P_i\| \leq \left(\#_{\mathsf{p}} M + \|M\| + \|N\|^{\#K \cdot d^5}\right)^{O(d)}$.

The following result combines Corollary 22 with Proposition 5.

▶ **Corollary 23** (small vector in set difference). *Let M, N be semi-linear sets such that $\|M\|, \|N\| \leq m$ and $M \setminus N \neq \emptyset$. Then there is $\boldsymbol{v} \in M \setminus N$ such that $\|\boldsymbol{v}\| \leq 2^{m^{O(d^2)}}$.*

5 An application: Exponent-sensitive commutative grammars

In this section, we show that our bounds on the difference of semi-linear sets yield a novel and tight upper bound for the equivalence problem for a class of commutative grammars.

Let $\Sigma = \{a_1, \ldots, a_m\}$ be a finite alphabet. The free commutative monoid generated by Σ is denoted by $\Sigma^\odot$, and we treat elements of $\Sigma^\odot$ as vectors in $\mathbb{N}^d$ where $d = |\Sigma|$. By $\Sigma^\oplus := \Sigma^\odot \setminus \{\mathbf{0}\}$ we denote the free commutative semi-group generated by Σ. An *exponent-sensitive commutative grammar* (ESCG) is a tuple $G = (N, \Sigma, S, \Pi)$, where N is a finite set of non-terminal symbols; Σ is a finite alphabet, the set of terminal symbols such that $N \cap \Sigma = \emptyset$; $S \in N$ is the axiom; and $\Pi \subseteq (\bigcup_{U \in N} \{U\}^\oplus) \times (N \cup \Sigma)^\odot$ is a finite set of productions.

ESCG are essentially equivalent to a generalisation of communication-free Petri nets in which incoming arcs may have multiplicity greater than one [18]. The size $|G|$ of G is the number of symbols required write it down; in particular we assume that commutative words from $\Sigma^\odot$ are encoded in *binary*. Subsequently, we write $V \to W$ whenever $(V, W) \in \Pi$. Let $D, E \in (N \cup \Sigma)^\odot$, we say D directly generates E, written $D \Rightarrow_G E$, iff there are $F \in (N \cup \Sigma)^\odot$ and $\pi \in \Pi$ such that $\pi = V \to W$, $D = V + F$ and $E = F + W$. We write $U \Rightarrow_G^* W$ for the reflexive transitive closure of $\Rightarrow_G$ and say that U generates W in this case. If G is clear from the context, we omit the subscript G. The language $\mathcal{L}(G)$ generated by G is defined as

$$\mathcal{L}(G) := \{W \in \Sigma^\odot : S \Rightarrow^* W\}.$$

Given ESCG G, H and $w \in \Sigma^\odot$, the word problem is to decide whether $w \in \mathcal{L}(G)$, and equivalence is to decide whether $\mathcal{L}(G) = \mathcal{L}(H)$. The word problem is PSPACE-complete; the equivalence problem was shown PSPACE-hard and decidable in 2-EXPSPACE by Mayr and Weihmann [18]. The latter result has recently been improved to coNEXP-hardness and membership in co-2NEXP in [7]. An application of Corollary 23 enables us to settle the complexity of the equivalence problem for ESCG.

▶ **Theorem 24.** *Equivalence for ESCG is* coNEXP*-complete.*

Proof (sketch). Let G, H be ESCG such that $\mathcal{L}(G) \neq \mathcal{L}(H)$, and with no loss of generality assume that there is some $w \in \mathcal{L}(G) \setminus \mathcal{L}(H)$. It is shown in [18] that $M = \mathcal{L}(G)$ and $N = \mathcal{L}(H)$ are semi-linear with $\|M\|, \|N\| \leq 2^{p(|G|+|H|)}$ for some fixed polynomial p. Consequently, by Corollary 23 we may assume that $\|w\| \leq 2^{2^{q(|G|+|H|)}}$ for some fixed polynomial q, and hence the representation size n of w is upper-bounded by $2^{q(|G|+|H|)}$. Thus, for the coNEXP upper bound it only remains to show that $w \in \mathcal{L}(G)$ and $w \notin \mathcal{L}(H)$ can be checked in time polynomial in the n. This is not completely obvious since the word problem for ESCG is PSPACE-complete. In the full version of this paper, we show how this obstacle can be avoided, bringing in a strategy that was used by Huynh [10] in order to show a coNEXP upper bound for the equivalence problem for context-free commutative grammars. ◀

References

1 M. Beck and F. Sottile. Irrational proofs for three theorems of Stanley. *Eur. J. Comb.*, 28(1):403–409, 2007. doi:10.1016/j.ejc.2005.06.003.

2 E. Domenjoud. Solving systems of linear Diophantine equations: An algebraic approach. In *Mathematical Foundations of Computer Science, MFCS*, pages 141–150, 1991. doi:10.1007/3-540-54345-7_57.

3 S. Eilenberg and M.P Schützenberger. Rational sets in commutative monoids. *J. Algebra*, 13(2):173–191, 1969. doi:10.1016/0021-8693(69)90070-2.

4 J. Gallier. Notes on convex sets, polytopes, polyhedra, combinatorial topology, Voronoi diagrams and Delaunay triangulations, 2012. Manuscript available at `http://www.cis.upenn.edu/~jean/gbooks/convexpoly.html`.

5 S. Ginsburg. *The mathematical theory of context free languages.* McGraw-Hill, 1966.

6 S. Ginsburg and E.H. Spanier. Bounded ALGOL-like languages. *T. Am. Math. Soc.*, pages 333–368, 1964. `doi:10.2307/1994067`.

7 C. Haase and P. Hofman. Tightening the complexity of equivalence problems for commutative grammars. In *Symposium on Theoretical Aspects of Computer Science, STACS*, volume 47 of *LIPIcs*, pages 41:1–41:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016. `doi:10.4230/LIPIcs.STACS.2016.41`.

8 C. Haase, B. Nill, and A. Paffenholz. Lecture notes on lattice polytopes. Preliminary version of December 7, 2012, available at `http://polymake.org/polytopes/paffenholz/data/preprints/ln_lattice_polytopes.pdf`.

9 M. Hague and A.W. Lin. Model checking recursive programs with numeric data types. In *Computer Aided Verification, CAV*, volume 6806 of *Lect. Notes Comp. Sci.*, pages 743–759. Springer, 2011. `doi:10.1007/978-3-642-22110-1_60`.

10 D.T. Huynh. The complexity of equivalence problems for commutative grammars. *Information and Control*, 66(1–2):103–121, 1985. `doi:10.1016/S0019-9958(85)80015-2`.

11 D.T. Huynh. A simple proof for the Σ_2^p upper bound of the inequivalence problem for semilinear sets. *Elektron. Inf.verarb. Kybern.*, 22(4):147–156, 1986.

12 T.-D. Huynh. The complexity of semilinear sets. *Elektron. Inf.verarb. Kybern.*, 18(6):291–338, 1982.

13 O.H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *J. ACM*, 25(1):116–133, 1978. `doi:10.1145/322047.322058`.

14 R. Ito. Every semilinear set is a finite union of disjoint linear sets. *J. Comput. Syst. Sci.*, 3(2):221–231, 1969. `doi:10.1016/S0022-0000(69)80014-0`.

15 E. Kopczyński. Complexity of problems of commutative grammars. *Log. Meth. Comput. Sci.*, 11(1), 2015. `doi:10.2168/lmcs-11(1:9)2015`.

16 M. Köppe and S. Verdoolaege. Computing parametric rational generating functions with a primal Barvinok algorithm. *Electr. J. Comb.*, 15(1), 2008.

17 J. Matoušek. *Lectures on discrete geometry.* Graduate texts in mathematics. Springer, 2002. `doi:10.1007/978-1-4613-0039-7`.

18 E.W. Mayr and J. Weihmann. Completeness results for generalized communication-free Petri nets with arbitrary edge multiplicities. In *Reachability Problems (RP'13)*, volume 8169 of *LNCS*, pages 209–221. Springer, 2013. `doi:10.1007/978-3-642-41036-9_19`.

19 A. Paffenholz. Polyhedral geometry and linear optimization. Preliminary version of July, 2010, available at `http://www.mathematik.tu-darmstadt.de/~paffenholz/daten/preprints/ln.pdf`.

20 R. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966. `doi:10.1145/321356.321364`.

21 L. Pottier. Minimal solutions of linear Diophantine systems: Bounds and algorithms. In *Rewriting Techniques and Applications, RTA*, volume 488 of *Lect. Notes Comp. Sci.*, pages 162–173. Springer, 1991. `doi:10.1007/3-540-53904-2_94`.

22 R.T. Rockafellar. *Convex Analysis.* Princeton University Press, 1970.

23 A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1986.

24 J. von zur Gathen and M. Sieveking. A bound on solutions of linear integer equalities and inequalities. *P. Am. Math. Soc.*, 72(1):155–158, 1978. `doi:10.1090/S0002-9939-1978-0500555-0`.

25 V. Weispfenning. The complexity of almost linear Diophantine problems. *J. Symb. Comp.*, 10(5):395–403, 1990. `doi:10.1016/S0747-7171(08)80051-X`.

Revision Notice

This is a revised version. It removes incorrectly stated upper bounds on the cardinalities of the sets of generators in Proposition 4 and Theorems 6 and 7.

Dagstuhl Publishing – October 26, 2016.

Characterizing Classes of Regular Languages Using Prefix Codes of Bounded Synchronization Delay

Volker Diekert[1] and Tobias Walter*†[2]

1 **University of Stuttgart, FMI, Stuttgart, Germany**
diekert@fmi.uni-stuttgart.de
2 **University of Stuttgart, FMI, Stuttgart, Germany**
walter@fmi.uni-stuttgart.de

Abstract

In this paper we continue a classical work of Schützenberger on codes with bounded synchronization delay. He was interested in characterizing those regular languages where the groups in the syntactic monoid belong to a variety **H**. He allowed operations on the language side which are union, intersection, concatenation and modified Kleene-star involving a mapping of a prefix code of bounded synchronization delay to a group $G \in \mathbf{H}$, but no complementation. In our notation this leads to the language classes $\mathrm{SD}_G(A^\infty)$ and $\mathrm{SD}_{\mathbf{H}}(A^\infty)$. Our main result shows that $\mathrm{SD}_{\mathbf{H}}(A^\infty)$ always corresponds to the languages having syntactic monoids where all subgroups are in **H**. Schützenberger showed this for a variety **H** if **H** contains Abelian groups, only. Our method shows the general result for all **H** directly on finite and infinite words. Furthermore, we introduce the notion of *local Rees extensions* which refers to a simple type of classical Rees extensions. We give a decomposition of a monoid in terms of its groups and local Rees extensions. This gives a somewhat similar, but simpler decomposition than in Rhodes' synthesis theorem. Moreover, we need a singly exponential number of operations, only. Finally, our decomposition yields an answer to a question in a recent paper of Almeida and Klíma about varieties that are closed under Rees extensions.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases formal language, synchronization delay, variety, Rees extension

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.129

In memoriam: Marcel-Paul Schützenberger (1920–1996)

1 Introduction

A fundamental result of Schützenberger characterizes the class of star-free languages SF as exactly those languages which are group-free, that is, aperiodic [15]. One usually abbreviates this result by $\mathrm{SF} = \mathbf{AP}$. Schützenberger also found another, but less prominent characterization of SF: the star-free languages are exactly the class of languages which can be defined inductively by finite languages and closure under finite union, concatenation, and the Kleene-star restricted to prefix codes of bounded synchronization delay [17]. This result

* Supported by the German Research Foundation (DFG) under grant DI 435/6-1.
† The authors thank an anonymous referee for the suggestion of the notation *group-controlled star*.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 129; pp. 129:1–129:14

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

is abbreviated by $\mathbf{AP} = \mathrm{SD}$. It is actually stronger than the famous $\mathrm{SF} = \mathbf{AP}$ because $\mathrm{SD} \subseteq \mathrm{SF} \subseteq \mathbf{AP}$ is relatively easy, see [11, Chapter VIII], so $\mathrm{SF} = \mathbf{AP}$ follows from $\mathbf{AP} \subseteq \mathrm{SD}$. The extension $\mathrm{SF} = \mathbf{AP}$ to infinite words is due to Perrin [10]. The result $\mathbf{AP} = \mathrm{SD}$ for infinite words was obtained much later in [5]. It became possible thanks to a "local divisor approach", which also is a main tool in this paper.

Schützenberger did not stop by showing $\mathbf{AP} = \mathrm{SD}$. In retrospective he started a program: in [16] he was able to prove an analogue of $\mathbf{AP} = \mathrm{SD}$ for languages where syntactic monoids have Abelian subgroups, only. In our notation $\mathbf{AP} = \mathrm{SD}$ means $\overline{\mathbf{1}}(A^\infty) = \mathrm{SD}_{\mathbf{1}}(A^\infty)$; and the main result in [16] is "essentially" equivalent to $\overline{\mathbf{Ab}}(A^*) = \mathrm{SD}_{\mathbf{Ab}}(A^*)$. (We write "essentially" because using the structure theory of Abelian groups, a sharper version than $\overline{\mathbf{Ab}}(A^*) = \mathrm{SD}_{\mathbf{Ab}}(A^*)$ is possible.) The proofs [16] use deep results in semigroup theory; and no such result beyond Abelian groups was known so far. Our result generalizes $\overline{\mathbf{Ab}}(A^\infty) = \mathrm{SD}_{\mathbf{Ab}}(A^\infty)$ to every variety $\mathbf{H}$ of finite groups: we show $\overline{\mathbf{H}}(A^\infty) = \mathrm{SD}_{\mathbf{H}}(A^\infty)$. We were able to prove it with much less technical machinery compared to [16]. For example, no knowledge in Krohn-Rhodes theory is required.

Actually, our result is a generalization of $\overline{\mathbf{Ab}}(A^*) = \mathrm{SD}_{\mathbf{Ab}}(A^*)$ [16] and also of $\mathbf{AP}(A^\infty) = \mathrm{SD}(A^\infty)$ [5]. More precisely, we give a characterization of languages which are recognized by monoids where all subgroups belong to $\mathbf{H}$. The characterization uses an inductive scheme starting with all finite subsets of finite words, allows concatenation, finite union, no (!) complementation, but a restricted use of a group-controlled star (resp. group-controlled ω-power). Let us explain the *group-controlled star* in our context. Instead of putting the star above a single language, consider first a disjoint union $K = \bigcup \{K_g \mid g \in G\}$ where G is a finite group and each K_g is regular in A^*. The "group-controlled star", more precisely the "G-controlled star", associates with such a disjoint union the following language:

$$\{u_{g_1} \cdots u_{g_k} \in K^* \mid u_{g_i} \in K_{g_i} \wedge g_1 \cdots g_k = 1 \in G\}\,.$$

Clearly, we obtain a regular language, but without any restriction, allowing such a "group star" yields all regular languages, even in the case of the trivial group. So, the construction is of no interest without a simultaneous restriction. The restriction considered in [16] yields an inductive scheme to define a class $\mathcal{C}$. The restriction says that such a group-controlled star is allowed only over a disjoint union $K = \bigcup \{K_g \mid g \in G\}$ where each K_g already belongs to $\mathcal{C}$ and where K is, in addition, a prefix code of bounded synchronization delay. The initials in "synchronization delay" led to the notation SD; and an indexed version SD_G (resp. $\mathrm{SD}_{\mathbf{H}}$) refers to *synchronization delay* over G (resp. over a finite group in $\mathbf{H}$). Since we also deal with infinite words we apply the same restriction to ω-powers.

Our results give also a new characterization for various other classes. For example, by a result of Straubing, Thérien and Thomas [20], the class of languages, having syntactic monoids where all subgroups are solvable, coincides with $(\mathrm{FO} + \mathrm{MOD})[<]$. Here, $(\mathrm{FO} + \mathrm{MOD})[<]$ means the class of languages defined by the logic $(\mathrm{FO} + \mathrm{MOD})[<]$. Thus, we are able to give a new language characterization: $(\mathrm{FO} + \mathrm{MOD})[<](A^\infty) = \mathrm{SD}_{\mathbf{Sol}}(A^\infty)$.

Moreover, as a sort of byproduct of $\overline{\mathbf{H}} = \mathrm{SD}_{\mathbf{H}}$, we obtain a simple and purely algebraic characterization of the monoids in $\overline{\mathbf{H}}$. Every monoid in $\overline{\mathbf{H}}$ can be decomposed in at most exponentially many iterated Rees extensions of groups in $\mathbf{H}$. The iteration uses only a very restricted version of Rees extensions: *local Rees extensions*. This means we obtain every finite monoid which is not a group as a divisor of a Rees extension between two proper divisors of M, one of them a proper submonoid, the other one a "local divisor".

Our decomposition result is similar to the synthesis theory of Rhodes and Allen [13]. Moreover, it yields a singly exponential bound on the number of operations whereas no such

bound was known by [13]. Finally, using this decomposition, we answer a recent question of Almeida and Klíma [1] concerning varieties which are closed under Rees extensions.

2 Preliminaries

Throughout, A denotes a finite alphabet and A^* is the free monoid over A. It consists of all finite words. The empty word is denoted by 1 as the neutral elements in other monoids or groups. The set of non-empty finite words is A^+; it is the free semigroup over A. By A^ω we denote the set of all infinite words with letters in A. For a set $K \subseteq A^*$, we let $K^\omega = \{u_1u_2\cdots \mid u_i \in K, u_i \text{ non-empty}, i \in \mathbb{N}\} \subseteq A^\omega$. In particular, $K^\omega = (K \setminus \{1\})^\omega$. Since our results concern finite and infinite words, it is convenient to treat finite and infinite words simultaneously. We define $A^\infty = A^* \cup A^\omega$ to be the set of finite or infinite words. Accordingly, a *language* L is a subset of A^∞. We say that L is *regular*, if first, $L \cap A^*$ is regular and second, $L \cap A^\omega$ is ω-regular in the standard meaning of formal language theory. In order to study regular languages algebraically, one considers finite monoids. A *divisor* of a monoid M is a monoid N which is a homomorphic image of a subsemigroup of M. In this case we write $N \preceq M$. A subsemigroup S of M is in our setting a divisor if and only if S is a monoid (but not necessarily a submonoid of M). A *variety* of finite monoids – hence, in Birkhoff's setting: a *pseudovariety* – is a class of finite monoids $\mathbf{V}$ which is closed under finite direct products and under division:

- If I is a finite index set and $M_i \in \mathbf{V}$ for each $i \in I$, then $\prod_{i\in I} M_i \in \mathbf{V}$. In particular, the trivial group $\{1\}$ belongs to $\mathbf{V}$.
- If $M \in \mathbf{V}$ and $N \preceq M$, then $N \in \mathbf{V}$.

Classical formal language theory states "regular" is the same as "recognizable". This means: $L \subseteq A^*$ is regular if and only if its syntactic monoid is finite; $L \subseteq A^\omega$ is regular if and only if its syntactic monoid in the sense of Arnold [2] is finite and, in addition, L is saturated by the syntactic congruence, see eg. [11, 21]. Here we use a notion of recognizability which applies to languages $L \subseteq A^\infty$. Let $\varphi : A^* \to M$ be a homomorphism to a finite monoid M. First, we define a relation $\sim_\varphi$ as follows. If $u \in A^*$ is a finite word, then we write $u \sim_\varphi v$ if v is finite and $\varphi(u) = \varphi(v)$. If $u \in A^\omega$ is an infinite word, then we write $u \sim_\varphi v$ if v is infinite and if there are factorizations $u = u_1u_2\cdots$ and $v = v_1v_2\cdots$ into finite nonempty words such that $\varphi(u_i) = \varphi(v_i)$ for all $i \geq 1$. It is easy to see that $\sim_\varphi$ is not transitive on infinite words, in general. Therefore, we consider its transitive closure $\approx_\varphi$. If $u, v \in A^*$, then we have

$$u \sim_\varphi v \iff u \approx_\varphi v \iff \varphi(u) = \varphi(v).$$

If $\alpha, \beta \in A^\omega$, then we have $\alpha \approx_\varphi \beta$ if and only if there is sequence of infinite words $\alpha_0, \ldots \alpha_k$ such that

$$\alpha = \alpha_0 \sim_\varphi \cdots \sim_\varphi \alpha_k = \beta.$$

We say that $L \subseteq A^\infty$ is *recognizable* by M if there exists a homomorphism $\varphi : A^* \to M$ such that $u \in L$ and $u \sim_\varphi v$ implies $v \in L$. We also say that M or φ recognizes L in this case. The connection to the classical notation is as follows. A regular language $L \subseteq A^\infty$ is recognizable (in our sense) by φ if and only if the syntactic monoids of $L \cap A^*$ and $L \cap A^\omega$ are divisors of M. Another equivalent definition can be given in terms of Wilke algebras [22].

Every variety $\mathbf{V}$ defines a family of regular languages $\mathbf{V}(A^\infty)$ as follows: we let $L \in \mathbf{V}(A^\infty)$ if there exists a monoid $M \in \mathbf{V}$ which recognizes L. Further, we define $\mathbf{V}(A^*) = \{L \subseteq A^* \mid L \in \mathbf{V}(A^\infty)\}$ and $\mathbf{V}(A^\omega) = \{L \subseteq A^\omega \mid L \in \mathbf{V}(A^\infty)\}$. A variety of finite groups is

a variety of finite monoids which contains only groups. Throughout $\mathbf{H}$ denotes a variety of finite groups. Special cases are the varieties

- $\mathbf{1}$: the trivial group $\{1\}$, only.
- $\mathbf{Ab}$: all finite Abelian groups.
- $\mathbf{Sol}$: all finite solvable groups.
- $\mathbf{Sol}_q$: all finite solvable groups where the order is divisible by some power of q.
- $\mathbf{G}$: all finite groups.

According to standard notation $\overline{\mathbf{H}}$ denotes the variety of finite monoids where all subgroups belong to $\mathbf{H}$. It is not completely obvious, but a classical fact [9], that $\overline{\mathbf{H}}$ is indeed a variety. In fact, it is the maximal variety $\mathbf{V}$ such that $\mathbf{V} \cap \mathbf{G} = \mathbf{H}$.

Clearly, $\overline{\mathbf{G}}$ is the class of all finite monoids. The most prominent subclass is $\overline{\mathbf{1}}$: it is the variety of aperiodic monoids $\mathbf{AP}$. The class $\mathbf{AP}(A^\infty) = \overline{\mathbf{1}}(A^\infty)$ admits various other characterizations as subsets of A^∞. For example, it is the class of star-free languages $\mathrm{SF}(A^\infty)$, it is the class of first-order definable languages, and it is the class of definable languages in linear temporal logic over finite or infinite words: $\mathrm{LTL}(A^\infty)$.

Local divisors. Let M be a finite monoid and $c \in M$. Consider the set $cM \cap Mc$ with a new multiplication $\circ$ which is defined as follows:

$$mc \circ cn = mcn.$$

A straightforward calculation shows that $cM \cap Mc$ becomes a monoid with this operation where the neutral element of M_c is c. Thus, the structure $M_c = (cM \cap Mc, \circ, c)$ defines a monoid. We say that M_c is the *local divisor* of M at c. If c is a unit, then M_c is isomorphic to M. If $c = c^2$, then M_c is the standard "local monoid" at the idempotent c.

The important fact is that M_c is always a divisor of M and that $|M_c| < |M|$ as soon as c is not a unit of M. Indeed, the mapping $\lambda_c : \{x \in M \mid cx \in Mc\} \to M_c$ given by $\lambda_c(x) = cx$ is a surjective homomorphism. Moreover, if c is not a unit, then $1 \notin cM \cap Mc$, hence $|M_c| < |M|$. Thus, if M belongs to some variety $\mathbf{V}$, then M_c belongs to the same variety. If M is not a group, then we find some nonunit $c \in M$ and the local divisor M_c is smaller than M. This makes the construction useful for induction. For a survey on the local divisor technique we refer to [6].

Rees extensions. Let N, L be monoids and $\rho : N \to L$ be any mapping. The *Rees extension* $\mathrm{Rees}(\mathrm{N}, \mathrm{L}, \rho)$ is a classical construction for monoids [12, 14], frequently described in terms of matrices. Here, we use an equivalent definition as in [7]. As a set we define

$$\mathrm{Rees}(\mathrm{N}, \mathrm{L}, \rho) = N \cup (N \times L \times N)\,.$$

The multiplication $\cdot$ on $\mathrm{Rees}(\mathrm{N}, \mathrm{L}, \rho)$ is given by

$$\begin{aligned}
n \cdot n' &= nn' && \text{for } n, n' \in N,\\
n \cdot (n_1, m, n_2) \cdot n' &= (nn_1, m, n_2 n') && \text{for } n, n', n_1, n_2 \in N, m \in L,\\
(n_1, m, n_2) \cdot (n_1', m', n_2') &= (n_1, m\rho(n_2 n_1')m', n_2') && \text{for } n_1, n_1', n_2, n_2' \in N, m, m' \in L.
\end{aligned}$$

The neutral element of $\mathrm{Rees}(\mathrm{N}, \mathrm{L}, \rho)$ is $1 \in N$ and $N \subseteq \mathrm{Rees}(\mathrm{N}, \mathrm{L}, \rho)$ is an embedding of monoids. In general, L is not a divisor of $\mathrm{Rees}(\mathrm{N}, \mathrm{L}, \rho)$. The following property holds.

▶ **Lemma 1.** *Let $N \preceq N'$ and $L \preceq L'$. Given $\rho : N \to L$, there exists a mapping $\rho' : N' \to L'$ such that* $\mathrm{Rees}(\mathrm{N}, \mathrm{L}, \rho)$ *is a divisor of* $\mathrm{Rees}(\mathrm{N}', \mathrm{L}', \rho')$.

Proof. First, assume that N (resp. L) is submonoid in N' (resp. L'). Let $\rho' : N' \to L'$ be any function such that $\rho'|_N = \rho$. The mapping $\pi : \mathrm{Rees(N, L, \rho)} \to \mathrm{Rees(N', L', \rho')}$ given by $\pi(n) = n$ and $\pi(n_1, \ell, n_2) = (n_1, \ell, n_2)$ is an injective homomorphism.

Second, let $\varphi : N' \to N$ and $\psi : L' \to L$ be surjective homomorphisms. Let $\rho' : N' \to L'$ be a function such that $\rho'(n) \in \psi^{-1}(\rho(\varphi(n)))$. Let $\pi : \mathrm{Rees(N', L', \rho')} \to \mathrm{Rees(N, L, \rho)}$ be the mapping defined by $\pi(n) = \varphi(n)$ and $\pi(n_1, \ell, n_2) = (\varphi(n_1), \psi(\ell), \varphi(n_2))$. It is clear that π is surjective. It is a homomorphism since

$$\pi((n_1, \ell, n_2) \cdot (n_1', \ell', n_2')) = \pi(n_1, \ell\rho'(n_2 n_1')\ell', n_2') = (\varphi(n_1), \psi(\ell) \underbrace{\psi(\rho'(n_2 n_1'))}_{=\rho(\varphi(n_2 n_1'))} \psi(\ell'), \varphi(n_2'))$$

$$= (\varphi(n_1), \psi(\ell), \varphi(n_2)) \cdot (\varphi(n_1'), \psi(\ell'), \varphi(n_2')) = \pi(n_1, \ell, n_2) \cdot \pi(n_1', \ell', n_2'). \qquad \blacktriangleleft$$

We are mainly interested in the case where N and L are proper divisors of a given finite monoid M. This leads to the notion of local Rees monoids. More precisely, let M be a finite monoid, N be a proper submonoid of M and M_c be a local divisor of M at c where c is not a unit. The *local Rees extension* $\mathrm{LocRees(N, M_c)}$ is defined as the Rees extension $\mathrm{Rees(N, M_c, \rho_c)}$ where ρ_c denotes the mapping $\rho_c : N \to M_c; x \mapsto cxc$.

For a variety $\mathbf{V}$ we define $\mathrm{Rees}(\mathbf{V})$ to be the least variety which contains $\mathbf{V}$ and is closed under taking Rees extensions and $\mathrm{LocRees}(\mathbf{V})$ to be the least variety which contains $\mathbf{V}$ and is closed under local Rees extensions.

2.1 Schützenberger's SD classes

Schützenberger gave a language theoretical characterization of the class of star-free languages $\mathrm{SF}(A^*)$ avoiding complementation, but allowing the star-operation to prefix codes of bounded synchronization delay [17].

A language $K \subseteq A^+$ is called *prefix code* if it is *prefix-free*. That is: $u, uv \in K$ implies $u = uv$. A prefix-free language K is a code since every word $u \in K^*$ admits a unique factorization $u = u_1 \cdots u_k$ with $k \geq 0$ and $u_i \in K$. Note that the empty set $\emptyset$ is considered to be a prefix code. More generally, if $L \subseteq A^+$ is any subset, then $K = L \setminus LA^+$ is a prefix code. A prefix code K has *bounded synchronization delay* if for some $d \in \mathbb{N}$ and for all $u, v, w \in A^*$ we have: if $uvw \in K^*$ and $v \in K^d$, then $uv \in K^*$. Note that the condition implies that for all $uvw \in K^*$ with $v \in K^d$, we have $w \in K^*$, too. If d is given explicitly, K is said to have synchronization delay d. Every subset $B \subseteq A$ (including the empty set) yields a prefix code with synchronization delay 0. If we have $c \in A \setminus B$, then B^*c is a prefix code with synchronization delay 1. If K is any prefix code with (or without) bounded synchronization delay, then K^m is a prefix code for all $m \in \mathbb{N}$, but for $m \geq 2$ it is never of bounded synchronization delay.

Consider a disjoint union $K = \bigcup \{K_g \mid g \in G\}$ of a prefix code K with bounded synchronization delay where G is a finite group and each K_g is regular in A^*. The *G-controlled star* associates with such a disjoint union the following language:

$$\{u_{g_1} \cdots u_{g_k} \in K^* \mid u_{g_i} \in K_{g_i} \wedge g_1 \cdots g_k = 1 \in G\}.$$

Another view of the G-controlled star of K is the following: Let $\gamma_K : K \to G$ be a mapping such that $K_g = \gamma_K^{-1}(g)$ and let $\gamma : K^* \to G$ denote the canonical extension of γ_K to a homomorphism from the free submonoid $K^* \subseteq A^*$ to G, then the G-controlled star of K is exactly the set $\gamma^{-1}(1)$. The generalization to infinite words $\gamma^{-1}(1)^\omega$ is called the G-controlled ω-power. Let $\mathcal{C}$ be a class of languages. We say that $\mathcal{C}$ is closed under G-controlled star (ω-power) if K is a prefix code with bounded synchronization delay, $K_g \in \mathcal{C}$ for all $g \in G$,

then the G-controlled star $\gamma^{-1}(1)$ (ω-power $\gamma^{-1}(1)^\omega$) is in $\mathcal{C}$. For a variety of groups $\mathbf{H}$ we say that $\mathcal{C}$ is closed under $\mathbf{H}$-controlled star (ω-power) if $\mathcal{C}$ is closed under G-controlled star (ω-power) for every group $G \in \mathbf{H}$. By $\mathrm{SD}_G(A^\infty)$ we denote the smallest class of regular languages such that $\emptyset \in \mathrm{SD}_G(A^\infty)$, $\{a\} \in \mathrm{SD}_G(A^\infty)$ for all letters $a \in A$, $\mathrm{SD}_G(A^\infty)$ is closed under finite union and concatenation, i.e., $L, K \in \mathrm{SD}_G(A^\infty)$ implies $L \cup K$ and $(L \cap A^*) \cdot K$ are both in $\mathrm{SD}_G(A^\infty)$, and $\mathrm{SD}_G(A^\infty)$ is closed under G-controlled star and G-controlled ω-power. We also define

$$\mathrm{SD}_G(A^*) = \{L \subseteq A^* \mid L \in \mathrm{SD}_G(A^\infty)\} \quad \text{and} \quad \mathrm{SD}_G(A^\omega) = \{L \subseteq A^\omega \mid L \in \mathrm{SD}_G(A^\infty)\}.$$

Note that for every homomorphism $\gamma : A^* \to G$ we have $\gamma^{-1}(1) \in \mathrm{SD}_G(A^*)$ and $\gamma^{-1}(1)^\omega \in \mathrm{SD}_G(A^\omega)$. This follows because first, A is a prefix code of bounded synchronization delay and second, all finite subsets of A are in $\mathrm{SD}_G(A^*)$.

Unlike the case of star-free sets, the definition of $\mathrm{SD}_G(A^\infty)$ does not use any complementation. By induction: for $L \subseteq A^\infty$ we have $L \in \mathrm{SD}_G(A^\infty)$ if and only if we can write $L = L_1 \cup L_2$ with $L_1 \in \mathrm{SD}_G(A^*)$ and $L_2 \in \mathrm{SD}_G(A^\omega)$. In the special case where $G = \{1\}$ is the trivial group, we also simply write SD instead of $\mathrm{SD}_{\{1\}}$. In this case closure under $\{1\}$-controlled stars (ω-powers) can be rephrased in simpler terms as follows: If $K \in \mathrm{SD}(A^*)$ is a prefix code of bounded synchronization delay, then $K^* \in \mathrm{SD}(A^*)$ and $K^\omega \in \mathrm{SD}(A^\omega)$.

In [16] Schützenberger showed (using a different notation) $\mathrm{SD}_{\mathbf{H}}(A^*) \subseteq \overline{\mathbf{H}}(A^*)$, but the converse only for $\mathbf{H} \subseteq \mathbf{Ab}$, see Proposition 6 for the first inclusion. Our aim is to show $\overline{\mathbf{H}}(A^\infty) \subseteq \mathrm{SD}_{\mathbf{H}}(A^\infty)$ for all $\mathbf{H}$, cf. Theorem 4. We begin with a technical lemma.

▶ **Lemma 2.** *Let $K \subseteq A^+$ be a prefix code of bounded synchronization delay and let $\gamma : K^* \to G$ be a homomorphism such that $\gamma^{-1}(g) \cap K \in \mathrm{SD}_G(A^*)$ for all $g \in G$, then we have $\gamma^{-1}(g) \in \mathrm{SD}_G(A^*)$ for all $g \in G$.*

Proof. For a word $w = u_1 \cdots u_k \in K^*$ we define $P(w) = \{\gamma(u_1 \cdots u_i) \mid 1 \le i \le k\} \subseteq G$ to be the set of prefixes of w in G. By an induction on $|P(w)|$ we construct languages $L(w) \in \mathrm{SD}_G(A^*)$ such that $w \in L(w) \subseteq \gamma^{-1}(\gamma(w))$ and the number $|\{L(w) \mid w \in K^*\}|$ of such languages is finite. The base case $|P(w)| = 0$ implies $g = 1$. We may choose $L(w) = \gamma^{-1}(1)$ and we are done, since $\gamma^{-1}(1) \in \mathrm{SD}_G(A^*)$ by definition. Hence, we may assume $|P(w)| \ge 1$. Let $g_1 = \gamma(u_1)$ and choose i maximal such that $g_1 = \gamma(u_1 \cdots u_i)$. Then we have $u_1 \cdots u_i \in (K \cap \gamma^{-1}(g_1)) \cdot \gamma^{-1}(1)$. Note that $P(w') = g_1^{-1} \cdot \{\gamma(u_1 \cdots u_j) \mid i < j \le k\}$ for $w' = u_{i+1} \cdots u_k$. By choice of i we have $g_1 \notin \{\gamma(u_1 \cdots u_j) \mid i < j \le k\}$ and therefore $|P(w')| = |\{\gamma(u_1 \cdots u_j) \mid i < j \le k\}| < |P(w)|$. By induction there exists $L(w')$ and we let $L(w) = (K \cap \gamma^{-1}(g_1)) \cdot \gamma^{-1}(1) \cdot L(w')$. The number of $|\{L(w) \mid w \in K^*\}|$ is therefore bounded by $\sum_{i=0}^{|G|} |G|^i$ which is less than $|G|^{|G|+1}$. The result follows because we can write $\gamma^{-1}(g) = \bigcup \{L(w) \mid w \in \gamma^{-1}(g)\}$ and this is a finite union. ◀

Clearly, we have for all G: if $K \in \mathrm{SD}_G(A^*)$ is a prefix code of bounded synchronization delay, then K^* and K^ω are both in $\mathrm{SD}_G(A^\infty)$. As a special case, using the prefix code $K = \emptyset$, it holds $K^* = \{1\} \in \mathrm{SD}_G(A^\infty)$. More generally, every finite language is in $\mathrm{SD}_G(A^\infty)$. Note also that for $G' \preceq G$ we have $\mathrm{SD}_{G'}(A^\infty) \subseteq \mathrm{SD}_G(A^\infty)$. In particular, $\bigcup \{\mathrm{SD}_{G_i}(A^\infty) \mid i \in I\} \subseteq \mathrm{SD}_{\prod_{i \in I} G_i}(A^\infty)$ for every finite index set I. This inclusion holds for every divisor of G as observed by the next lemma which can be proved by induction.

▶ **Lemma 3.** $\mathrm{SD}_H(A^\infty) \subseteq \mathrm{SD}_G(A^\infty)$ *holds for $H \preceq G$.*

We will formulate our some of results on the language classes $\mathrm{SD}_G(A^\infty)$ to obtain finer results. However, our main result concerns the language class

$$\mathrm{SD}_{\mathbf{H}}(A^\infty) = \bigcup \{\mathrm{SD}_G(A^\infty) \mid G \in \mathbf{H}\}.$$

▶ **Theorem 4.** *Let* $\mathbf{H}$ *be a variety of finite groups. Then* $\overline{\mathbf{H}}(A^\infty)$ *is the smallest class of languages* $\mathcal{C}$ *closed under finite union, concatenation,* $\mathbf{H}$*-controlled star and* $\mathbf{H}$*-controlled* ω*-power such that* $\mathcal{C}$ *contains all finite languages over* A^**. In other words, it holds* $\overline{\mathbf{H}}(A^\infty) = \mathrm{SD}_{\mathbf{H}}(A^\infty)$*.*

▶ **Corollary 5.** $\mathrm{SD}_{\mathbf{H}}(A^\infty)$ *is closed under complementation and intersection for every variety* $\mathbf{H}$ *of finite groups.*

An algebraic characterization of $\overline{\mathbf{H}}$ in terms of Rees extensions will be given in Theorem 15. The proof of Theorem 4 covers the next two sections.

3 Closure properties of $\mathrm{SD}_{\mathbf{H}}$

In this section we prove $\mathrm{SD}_{\mathbf{H}}(A^\infty) \subseteq \overline{\mathbf{H}}(A^\infty)$. Therefore one has to study the closure properties under the operations given in the definition of $\mathrm{SD}_{\mathbf{H}}(A^\infty)$, that is, one has to show that those operations do not introduce new groups.

The next proposition shows that the $\mathbf{H}$-controlled star does not introduce new groups.

▶ **Proposition 6** ([16])**.** *Let* $K = \bigcup\{K_g \mid g \in G\} \subseteq A^+$ *be a prefix code of bounded synchronization delay where each* K_g *is regular. Then all subgroups in the syntactic monoid of the* G*-controlled star are divisors either of* G *or of the direct product* $\prod_{g\in G} \mathrm{Synt}(K_g)$*.*

We will prove the same for $\gamma^{-1}(1)^\omega$, relying on Proposition 6 as a blackbox. The concept used for transfering the properties to infinite words are Birget-Rhodes expansions [3, 4]. The Birget-Rhodes expansion of a monoid M is the monoid $\mathrm{Exp(M)} = \{(X, m) \mid 1, m \in X \subseteq M\}$. The multiplication on Exp(M) is given as a semi-direct product: $(X, m)\cdot(Y, n) = (X \cup m\cdot Y, m\cdot n)$. Note that M is isomorphic to the submonoid $\{(M, m) \mid m \in M\}$ of Exp(M), that is, M is a divisor of Exp(M). Moreover, the following lemma shows that the Birget-Rhodes expansion has the same groups as M.

▶ **Lemma 7.** *Every subgroup of* Exp(M) *is isomorphic to some group in* M*.*

Proof. Let $G \subseteq \mathrm{Exp(M)}$ be a group contained in Exp(M) and let $(X, e) \in G$ be the unit in G. For every element $(Y, m) \in G$ we have $(X, e)(Y, m) = (X \cup eY, em) = (Y, m)$ and hence, $X \subseteq Y$. Furthermore, $(Y, m)^{|G|} = (Y \cup \cdots, m^{|G|}) = (X, e)$ and we conclude $X = Y$. Thus, $(X, m) \mapsto m$ is an injective embedding of G into M. ◀

The idea behind the Birget-Rhodes expansion is that it stores the seen prefixes in a set.

▶ **Lemma 8.** *Let* $\varphi : A^* \to M$ *be a homomorphism and* $\psi : A^* \to \mathrm{Exp(M)}$ *be the homomorphism given by* $\psi(a) = (\{1, \varphi(a)\}, \varphi(a))$*. Let* $u \in A^*$ *and* $\psi(u) = (X, \varphi(u))$*. For every* $m \in X$ *there exists a prefix* v *of* u *such that* $\varphi(v) = m$*.*

Proof. We will prove this inductively. The statement is true if u is the empty word. Thus, consider $u = va$ for some letter $a \in A$. Let $\psi(v) = (Y, \varphi(v))$, then

$$\psi(u) = \psi(v) \cdot (\{1, \varphi(a)\}, \varphi(a)) = (Y \cup \{\varphi(v), \varphi(v)\varphi(a)\}, \varphi(u)).$$

Inductively, we obtain prefixes of v, and therefore also prefixes of u, for all elements of Y. The only (potentially) new element in X is $\varphi(u)$. This proves the claim. ◀

A special kind of ω-regular languages are *arrow languages*. Let $L \subseteq A^*$ be a language. We define $\overrightarrow{L} = \{\alpha \in A^\omega \mid \text{infinitely many prefixes of } \alpha \text{ are in } L\}$ to be the arrow language

of L. The set of arrow languages is exactly the set of deterministic languages [21]. The Birget-Rhodes expansion can be used to obtain a recognizing monoid for $\overrightarrow{L}$, given a monoid for L. For a related result see [10].

▶ **Proposition 9.** *Let $L \subseteq A^*$ be some regular language and $\varphi : A^* \to M$ be a homomorphism which recognizes L, then $\overrightarrow{L}$ is recognized by* Exp(M).

Proof. Let $\psi : A^* \to \mathrm{Exp(M)}$ be the homomorphism given by $\psi(a) = (\{1, \varphi(a)\}, \varphi(a))$. Let $\alpha \in \overrightarrow{L}$ and $\alpha \sim_\psi \beta$. We show that $\beta \in \overrightarrow{L}$. Let $\alpha = u_1 u_2 \cdots$ and $\beta = v_1 v_2 \cdots$ be factorizations such that $\psi(u_i) = \psi(v_i)$. Since $\alpha \in \overrightarrow{L}$, we may assume that for every i there exists a decomposition $u_i = u_i' u_i''$ such that $u_1 \cdots u_{i-1} u_i' \in L$. By $\psi(u_i) = \psi(v_i)$ and Lemma 8, there exists a decomposition $v_i = v_i' v_i''$ such that $\varphi(u_i') = \varphi(v_i')$. Thus, $u_1 \cdots u_{i-1} u_i' \sim_\varphi v_1 \cdots v_{i-1} v_i'$ and therefore $v_1 \cdots v_{i-1} v_i' \in L$. This implies $\beta \in \overrightarrow{L}$. ◀

▶ **Proposition 10.** *If $L \in \mathrm{SD}_G(A^\infty)$, then all subgroups in* Synt(L) *are a divisor of a direct product of copies of G.*

Proof. We will prove this inductively on the definition of $\mathrm{SD}_G(A^\infty)$. The cases $\emptyset \in \mathrm{SD}_G(A^\infty)$ and $\{a\} \in \mathrm{SD}_G(A^\infty)$ for all letters $a \in A$ are straightforward, as they are recognized by aperiodic monoids. Let L, K be languages, such that their syntactic monoids contain only groups which are divisors of a direct product of G. The language $L \cup K$ is recognized by the direct product of their syntactic monoids which implies the statement. $(L \cap A^*) \cdot K$ is recognized by the Schützenberger product of their syntactic homomorphisms [10] and [8, Proposition 11.7.10]. The Schützenberger product does not introduce new groups [15].

Let $K \subseteq A^+$ be a prefix code of bounded synchronization delay and $\gamma : K^* \to G$ be a homomorphism of the free monoid K^* to the group G such that for all $g \in G$ every subgroup of Synt$(K \cap \gamma^{-1}(g))$ is a divisor of a direct product of copies of G. Proposition 6 implies that every subgroup of Synt$(\gamma^{-1}(1))$ is a divisor of a direct product of copies of G. Note that $\gamma^{-1}(1)^\omega = \overrightarrow{\gamma^{-1}(1)}$ and therefore Proposition 9 and Lemma 7 imply that every subgroup of Synt$(\gamma^{-1}(1)^\omega)$ is a divisor of a direct product of copies of G. ◀

4 The inclusion $\overline{\mathbf{H}}(A^\infty) \subseteq \mathrm{SD}_{\mathbf{H}}(A^\infty)$

We prove that if every subgroup of M is a divisor of G, then every language recognized by M is contained in $\mathrm{SD}_G(A^\infty)$. This result is again finer than just the inequality $\overline{\mathbf{H}}(A^\infty) \subseteq \mathrm{SD}_{\mathbf{H}}(A^\infty)$. The proof works by induction on $|M|$ and on the alphabet and decomposes every $\approx_\varphi$-class into several sets in $\mathrm{SD}_G(A^\infty)$. As a byproduct we obtain a normal form for the languages in $\mathrm{SD}_G(A^\infty)$.

▶ **Proposition 11.** *Let $L \subseteq A^\infty$ be recognized by $\varphi : A^* \to M$ and let G be a group such that every subgroup of M is a divisor of G, then $L \in \mathrm{SD}_G(A^\infty)$. Moreover, L can be written as finite union*

$$L = L_0 \cup \bigcup_{i=1}^{m} L_i \cdot \gamma_i^{-1}(1)^\omega$$

for $L_i \in \mathrm{SD}_G(A^*)$ and $\gamma_i : K_i^* \to G$ for prefix codes $K_i \in \mathrm{SD}_G(A^*)$ of bounded synchronization delay with $\gamma_i^{-1}(g) \cap K_i \in \mathrm{SD}_G(A^*)$ for all $g \in G$. All products in the expressions of L_i are unambiguous.

Proof. Let $[\![w]\!]_\varphi = \{v \in A^\infty \mid w \approx_\varphi v\}$ be the equivalence class of w. Since L is recognized by φ, it holds $L = \cup_{w\in L} [\![w]\!]_\varphi$. Our goal is to construct languages $L(w) \in \mathrm{SD}_G(A^\infty)$ such that

- $w \in L(w) \subseteq [\![w]\!]_\varphi$.
- the number of such languages is finite.
- every word in $L(w)$ starts with the same letter.

In particular, we want to saturate $[\![w]\!]_\varphi$ by sets in $\mathrm{SD}_G(A^\infty)$. The construction of the set $L(w)$ is by induction on $(|M|, |A|)$ with lexicographic order.

If $w = 1$, then we set $L(w) = \{1\}$. This concludes the induction base $|A| = 0$. Let us consider the case that $\varphi(A^*)$ is a group, that is, a divisor of G. Let $K = A$. The set K is a prefix code of synchronization delay 0 and we may choose the homomorphism $\gamma = \varphi$. Note that every subset of A is in $\mathrm{SD}_G(A^*)$. In particular, $K_g = K \cap \gamma^{-1}(g) \in \mathrm{SD}_G(A^*)$ for all $g \in \varphi(A^*)$. This shows $\gamma^{-1}(g) = \varphi^{-1}(g) \in \mathrm{SD}_G(A^*)$ for all $g \in \varphi(A^*)$ by Lemma 2 and Lemma 3. In order to satisfy the third condition let $w = av \in aA^*$ for some $a \in A$ and set $L(w) = a\varphi^{-1}(\varphi(v))$. It is clear that $w \in L(w) \subseteq [\![w]\!]_\varphi$ and $L(w) \in \mathrm{SD}_G(A^*)$ by the above. If $w \in aA^\omega$, then we obtain $w \in a\varphi^{-1}(g)\varphi^{-1}(1)^\omega$ for some $g \in \varphi(A^*)$ by the pigeonhole principle. Thus, we may set $L(w) = a\varphi^{-1}(g)\varphi^{-1}(1)^\omega$. Note that by the definition of $\sim_\varphi$, the inclusion $L(w) \subseteq [\![w]\!]_\varphi$ holds. In particular, these cases include the induction base $|M| = 1$.

In the following we assume that $\varphi(A^*)$ is not a group and therefore there exists a letter $c \in A$ such that $\varphi(c)$ is not a unit. Fix this letter $c \in A$ and set $B = A \setminus \{c\}$. If $w \in B^\infty$, the set $L(w)$ exists by induction. Let $w = uv$ with $u \in B^*$ and $v \in cA^\infty$. By induction we obtain $L(u) \in \mathrm{SD}_G(B^\infty) \subseteq \mathrm{SD}_G(A^\infty)$ and it remains to show $L(v) \in \mathrm{SD}_G(A^\infty)$. Note that the product $L(w) = L(u) \cdot L(v)$ is unambiguous. From now on we may assume $w \in cA^\infty$. Let us first consider the case $w = uv$ with $u \in c(B^*c)^*$ and $v \in B^\infty$, i.e., there are only finitely many occurrences of the letter c in w. By induction, there exists $L(v) \in \mathrm{SD}_G(B^\infty) \subseteq \mathrm{SD}_G(A^\infty)$ and by setting $L(w) = L(u) \cdot L(v)$ it remains to construct $L(u)$.

Consider the alphabet $T = \varphi(B^*) = \{\varphi(u) \mid u \in B^*\}$. Let M_c be the local divisor of M at $\varphi(c)$. Since M_c is a divisor of M, every subgroup of M_c is a divisor of G. Consider the homomorphism $\psi : T^* \to M_c$ given by $\psi(\varphi(u)) = \varphi(cuc)$ and the substitution $\sigma : (B^*c)^\infty \to T^\infty$ with $\sigma(u_1cu_2c\ldots) = \varphi(u_1)\varphi(u_2)\cdots$. Note that

$$\begin{aligned}\psi(\sigma(u_1cu_2c\ldots u_nc)) &= \psi(\varphi(u_1)\varphi(u_2)\cdots\varphi(u_n)) = \varphi(cu_1c)\circ\varphi(cu_2c)\circ\cdots\circ\varphi(cu_nc)\\ &= \varphi(cu_1cu_2c\ldots cu_nc)\end{aligned}$$

and thus $\varphi^{-1}(m) \cap c(B^*c)^* = c\sigma^{-1}(\psi^{-1}(m))$. Since $|M_c| < |M|$, we can apply induction on the monoid size and there exists a language $L(\sigma(u')) \in \mathrm{SD}_G(T^\infty)$ for all $u' \in (B^*c)^*$. We set $L(u) = c\sigma^{-1}(L(\sigma(u')))$ for $u = cu'$. In order to complete the case of finitely many c's, it suffices to show the following claim:

▶ **Claim.** *It is $\sigma^{-1}(K) \in \mathrm{SD}_G(A^\infty)$ for all $K \in \mathrm{SD}_G(T^\infty)$.*

Proof of the Claim: We prove the claim inductively on the definition of SD_G. For $K = \emptyset$, we obtain $\sigma^{-1}(K) = \emptyset \in \mathrm{SD}_G(A^\infty)$. Furthermore,

$$\sigma^{-1}(t) = \bigcup_{v\in B^*, t=\varphi(v)} L(v)c \in \mathrm{SD}_G(A^\infty).$$

Let $L, K \in \mathrm{SD}_G(T^\infty)$. A basic result from set theory yields $\sigma^{-1}(L \cup K) = \sigma^{-1}(L) \cup \sigma^{-1}(K)$. Let $\sigma(v) = w_1w_2$ for some $v \in (B^*c)^*$. Since B^*c is a prefix code, there exists a unique factorization $v = v_1v_2$ with $v_1, v_2 \in (B^*c)^*$ such that $\sigma(v_1) = w_1$ and $\sigma(v_2) = w_2$.

Thus, we conclude $\sigma^{-1}(K \cdot L) = \sigma^{-1}(K) \cdot \sigma^{-1}(L)$. Let now $K \in \mathrm{SD}_G(T^\infty)$ be a prefix code of synchronization delay d. We first show that $\sigma^{-1}(K)$ is a prefix code of bounded synchronization delay. Let $u, uv \in \sigma^{-1}(K)$, then $\sigma(u), \sigma(uv) = \sigma(u)\sigma(v) \in K$ and therefore $\sigma(v) = 1$. This implies $v = 1$ and $\sigma^{-1}(K)$ is a prefix code. We prove that $\sigma^{-1}(K)$ has synchronization delay $d + 1$. The incrementation of the synchronization delay by one comes from the fact that B^*c is not a suffix code, and thus we need another word in B^*c to pose as a left marker. Consider $uvw \in \sigma^{-1}(K)^*$ with $v \in \sigma^{-1}(K)^{d+1}$ and factorize $v = v_1cv_2$ with $v_2 \in \sigma^{-1}(K)^d = \sigma^{-1}(K^d)$. Then $\sigma(uvw) = \sigma(uv_1c)\sigma(v_2)\sigma(w)$, and by $\sigma(v_2) \in K^d$ this implies $\sigma(uv) = \sigma(uv_1c)\sigma(v_2) \in K^*$. Thus, $uv \in \sigma^{-1}(K)^*$. Let $\gamma : K^* \to G$ be some homomorphism and $K_g = K \cap \gamma^{-1}(g) \in \mathrm{SD}_G(T^\infty)$ for all $g \in G$. Inductively, $\sigma^{-1}(K_g) \in \mathrm{SD}_G(A^\infty)$ and $\sigma^{-1}(K) = \bigcup \sigma^{-1}(K_g)$. Let $\gamma' : \sigma^{-1}(K)^* \to G$ be induced by $\gamma'(u) = \gamma(\sigma(u))$. By definition of $\mathrm{SD}_G(A^\infty)$ we obtain $\gamma'^{-1}(1) \in \mathrm{SD}_G(A^\infty)$. However, $u_1 \cdots u_n \in \sigma^{-1}(\gamma^{-1}(1))$ if and only if $\gamma(\sigma(u_1 \cdots u_n)) = 1$. Furthermore, note that $\gamma(\sigma(u_1 \cdots u_n)) = \gamma(\sigma(u_1)) \cdots \gamma(\sigma(u_n)) = \gamma'(u_1) \cdots \gamma'(u_n) = \gamma'(u_1 \cdots u_n)$. Thus, we obtain $\sigma^{-1}(\gamma^{-1}(1)) = \gamma'^{-1}(1) \in \mathrm{SD}_G(A^\infty)$ and $\sigma^{-1}(\gamma^{-1}(1)^\omega) = \gamma'^{-1}(1)^\omega \in \mathrm{SD}_G(A^\infty)$. This concludes the proof of the claim. ◀

At this point we showed the proposition for languages $L \subseteq A^*$.

The last case of the proof is that w contains infinitely many c's, that is, $w = cv$ with $v \in (B^*c)^\omega$. By induction, we know that $\sigma(v) \in L_T \cdot \gamma_T^{-1}(1)^\omega \subseteq [\![\sigma(v)]\!]_\psi$ for some $L_T \in \mathrm{SD}_G(T^*)$ and $\gamma_T : K_T^* \to G$ for some prefix code $K_T \in \mathrm{SD}_G(T^*)$ of bounded synchronization delay with $\gamma_T^{-1}(g) \cap K_T \in \mathrm{SD}_G(T^*)$. By the calculation above, there exists a $\gamma : K^* \to G$ with the usual properties such that $\gamma^{-1}(1) = \sigma^{-1}(\gamma_T^{-1}(1))$. Let $L = \sigma^{-1}(L_T)$ and set $L(w) = cL\gamma^{-1}(1)^\omega$. It remains to show that $cL\gamma^{-1}(1)^\omega \subseteq [\![w]\!]_\varphi$. Let $cu \in cL\gamma^{-1}(1)^\omega$, then $\sigma(u) \in [\![\sigma(v)]\!]_\psi$, that is $\sigma(u) \approx_\psi \sigma(v)$. Since $\approx_\psi$ is the transitive closure of $\sim_\psi$, we show that $\sigma(u) \sim_\psi \sigma(v)$ implies $cu \approx_\varphi cv$ for all $u, v \in (B^*c)^\omega$ which concludes the proof. Now, let $\sigma(u) = \sigma(u_1c)\sigma(u_2c)\cdots$ and $\sigma(v) = \sigma(v_1c)\sigma(v_2c)\cdots$ such that $\psi(\sigma(u_ic)) = \psi(\sigma(v_ic))$. As observed above, this implies $\varphi(cu_ic) = \varphi(cv_ic)$. Thus,

$$\begin{aligned} cu &= (cu_1c)u_2(cu_3c)u_4(c\cdots \sim_\varphi (cv_1c)u_2(cv_3c)u_4(c\cdots \\ &= cv_1(cu_2c)v_3(cu_4c)\cdots \sim_\varphi cv_1(cv_2c)v_3(cv_4c)\cdots \\ &= cv. \end{aligned}$$

This implies the existence of finitely many sets $L(w) \in \mathrm{SD}_G(A^\infty)$ with $w \in L(w) \subseteq [\![w]\!]_\varphi$ in the case of infinitely many c's. ◀

5 Rees extension monoids

We need the fact that every group contained in $\mathrm{Rees}(\mathrm{N}, \mathrm{M}, \rho)$ is contained in N or in M.

▶ **Lemma 12** ([1]). *Let G be a subgroup of $\mathrm{Rees}(\mathrm{N}, \mathrm{M}, \rho)$, then there exists an embedding of G into N or into M.*

Thus, Lemma 12 implies $\mathrm{LocRees}(\mathbf{H}) \subseteq \mathrm{Rees}(\mathbf{H}) \subseteq \mathrm{Rees}(\overline{\mathbf{H}}) \subseteq \overline{\mathbf{H}}$ for any group variety $\mathbf{H}$. We want to prove equality, that is, every monoid which contains only groups in $\mathbf{H}$ is a divisor of an iterated Rees extension of groups in $\mathbf{H}$. However, we are able to prove a stronger statement using only local Rees extensions.

▶ **Lemma 13.** *Let M be a monoid, N be a submonoid of M and $c \in M$. If N and c generate M, then M is a homomorphic image of the local Rees extension $\mathrm{LocRees}(\mathrm{N}, \mathrm{M_c})$.*

Proof. Let $\varphi : \mathrm{LocRees}(\mathrm{N}, \mathrm{M_c}) \to M$ be the mapping given by $\varphi(n) = n$ for $n \in N$ and $\varphi(u, x, v) = uxv$ for $(u, x, v) \in N \times M_c \times N$. Since

$$\begin{aligned}\varphi((u,x,v)(s,y,t)) &= \varphi(u, x \circ cvsc \circ y, t) = \varphi(u, xvsy, t)\\ &= (uxv)(syt) = \varphi(u,x,v)\varphi(s,y,t),\end{aligned}$$

φ is a homomorphism. Obviously, $M = N \cup NM_cN$ and thus φ is surjective. ◀

A *Rees decomposition* of a monoid M is a sequence of monoids $M_1, \ldots, M_k = M$ such that for each $1 \le j \le k$ we have for M_j one of the following:

- M_j is a group which is a divisor of M.
- M_j is a divisor of a local Rees extension of a submonoid M_i of M_j and a local divisor M_ℓ of M_j with $i, \ell < j$.

▶ **Proposition 14.** *A finite monoid M has a Rees decomposition of length at most $2^{|M|} - 1$.*

Proof. We prove the statement with induction on $|M|$. If M is a group, we set $M_1 = M$. This includes the base case $|M| = 1$. If M is not a group, we may choose a minimal generating set of M. Let c be a nonunit of this generating set, then there exists a proper submonoid N of M such that N and c generate M. Since c is not a unit, the local divisor M_c is smaller than M, that is, $|M_c| < |M|$. By induction, there exist Rees decompositions $M'_1, \ldots, M'_{k'} = N$ and $M''_1, \ldots, M''_{k''} = M_c$ with $k', k'' \le 2^{|M|-1} - 1$. Note that every group, which is a divisor of N or M_c also is a divisor of M. Furthermore, M is a divisor of the local Rees extension of $M_{k'} = N$ and $M_{k'+k''} = M_c$ by Lemma 13. Therefore, choosing

- $M_i = M'_i$ for $1 \le i \le k'$
- $M_{i+k'} = M''_i$ for $1 \le i \le k''$
- $M_{k'+k''+1} = M$

leads to such a sequence for M. Since $k' + k'' + 1 \le 2 \cdot (2^{|M|-1} - 1) + 1 = 2^{|M|} - 1$, the bound on k holds. ◀

The inclusion $\overline{\mathbf{H}} \subseteq \mathrm{LocRees}(\mathbf{H})$ is immediate from Proposition 14. This yields

▶ **Theorem 15.** *Let $\mathbf{H}$ be a variety of finite groups. Then $\overline{\mathbf{H}} = \mathrm{LocRees}(\mathbf{H}) = \mathrm{Rees}(\mathbf{H})$.*

In particular, every monoid in $\overline{\mathbf{H}}$ is a divisor of an iterated Rees extension of groups in $\mathbf{H}$ by Lemma 1. We can draw the decomposition as a tree based on the decomposition of M in submonoids and local divisors. We do not describe this formally but content ourselves to give an example.

▶ **Example 16.** Let M be the monoid generated by $\{a, b, \delta, \sigma\}$ with the relations $a^2 = b^2 = ab = ba = 0$, $a\delta = a$, $\delta\sigma = \sigma\delta^2$, $\delta^3 = 1$, $\sigma^2 = 1$ and $d\delta = \delta d$, $d\sigma = \sigma d$ with $d \in \{a, b\}$. The subgroup generated by δ and σ is the symmetric group $\mathfrak{S}_3$; it is solvable but not abelian. The monoid M is syntactic for the language L which is a union of L_a and L_b. The language L_a is the set of all words uav with $uv \in \{\delta, \sigma\}^*$ and the sign of the permutation uv evaluates to -1. The language L_b is the set of all words ubv with $uv \in \{\delta, \sigma\}^*$ and uv evaluates in $\mathfrak{S}_3$ to δ. The decomposition in local Rees extensions from Proposition 14 is depicted in Figure 1. Here $M[a, \sigma, \delta]$ denotes the submonoid generated by $\{a, \sigma, \delta\}$. In particular, this yields

$$M \preceq \mathrm{Rees}(\mathrm{Rees}(\mathfrak{S}_3, \mathrm{Rees}(\mathbb{Z}/2\mathbb{Z}, \{1\}, \rho_1), \rho_2), \mathrm{Rees}(\mathfrak{S}_3, \{1\}, \rho_3), \rho_4)$$

for some $\rho_1, \rho_2, \rho_3, \rho_4$ by Lemma 1.

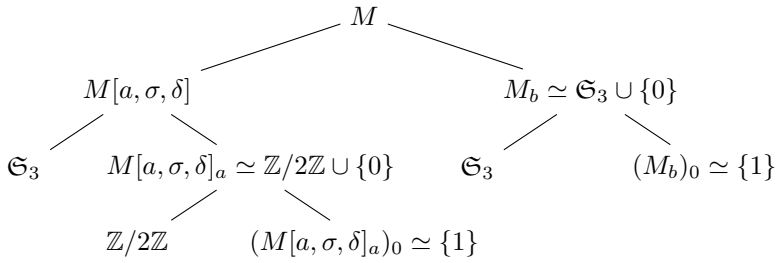


Figure 1 Decomposition tree of the monoid in Example 16.

6 Applications

An application of Proposition 14 is the solution to an open problem of Almeida and Klíma. Let $\mathbf{U}$ and $\mathbf{V}$ be varieties. Let $\mathrm{Rees}(\mathbf{U},\mathbf{V})$ be the variety generated by $\mathrm{Rees}(\mathrm{N},\mathrm{M},\rho)$ for $N \in \mathbf{U}$ and $M \in \mathbf{V}$. Note that in general $\mathrm{Rees}(\mathbf{V}) \neq \mathrm{Rees}(\mathbf{V},\mathbf{V})$. However $\mathrm{Rees}(\mathbf{V})$ can be defined as the limit of this operation. Let $\mathbf{V}_i = \mathrm{Rees}(\mathbf{V}_{i-1},\mathbf{V}_{i-1})$ and $\mathbf{V}_0 = \mathbf{V}$, then

$$\mathrm{Rees}(\mathbf{V}) = \bigcup_{i\in\mathbb{N}} \mathbf{V}_i.$$

The variety $\mathrm{Rees}(\mathbf{U},\mathbf{V})$ has recently been introduced by Almeida and Klíma under the name of *bullet operation* [1]. They defined a variety $\mathbf{V}$ to be *bullet idempotent* if $\mathbf{V} = \mathrm{Rees}(\mathbf{V},\mathbf{V})$ and they asked whether there are varieties apart from $\overline{\mathbf{H}}$ which are bullet idempotent. Using our decomposition above, we prove that the answer to this question is "No".

▶ **Theorem 17.** *Let $\mathbf{V}$ be a bullet idempotent variety and let $\mathbf{H} = \mathbf{V} \cap \mathbf{G}$, then $\mathbf{V} = \overline{\mathbf{H}}$.*

Proof. Since $\overline{\mathbf{H}}$ is the maximal variety with $\overline{\mathbf{H}} \cap \mathbf{G} = \mathbf{H}$, we have $\mathbf{V} \subseteq \overline{\mathbf{H}}$. Let $M \in \overline{\mathbf{H}}$. Inductively, we may assume that every proper divisor of M is in $\mathbf{V}$. If M is a group, then $M \in \mathbf{H}$ and thus $M \in \mathbf{V}$. Thus, there exists an nonunit element $c \in M$ and a proper submonoid N of M such that N and c generate M. By Lemma 13, M is a divisor of $\mathrm{LocRees}(\mathrm{N},\mathrm{M_c})$, and since $N, M_c \in \mathbf{V}$ and $\mathbf{V} = \mathrm{Rees}(\mathbf{V},\mathbf{V})$ we obtain $M \in \mathbf{V}$. ◀

Let $(\mathrm{FO}+\mathrm{MOD}_q)[<]$ be the fragment of first-order sentences which only use first-order quantifiers, modular quantifiers of modulus q and the predicate $<$. Then the following theorem holds.

▶ **Corollary 18.** $(\mathrm{FO}+\mathrm{MOD}_q)[<](A^\infty) = \mathrm{SD}_{\mathbf{Sol}_q}(A^\infty)$

Proof. By [20], see also [19] for a complete treatise, $(\mathrm{FO}+\mathrm{MOD}_q)[<]$ describes the family of all regular languages such that every group in the syntactic monoid is a solvable group of cardinality dividing a power of q, that is the languages in $\mathbf{Sol}_q$. Theorem 4 then implies the stated equality. ◀

The same language class has been described by Straubing with another operation, counting how many prefixes are in a given language, which resembles more closely the counting modulo q [18].

7 Summary

Our main theorem Theorem 4 states $\overline{\mathbf{H}}(A^\infty) = \mathrm{SD}_{\mathbf{H}}(A^\infty)$. An overview over the contributions for $\overline{\mathbf{H}}$ is given in Table 1.

Table 1 Overview of existing and new language characterizations of $\overline{\mathbf{H}}$.

	$\overline{\mathbf{1}}$	$\overline{\mathbf{Ab}}$	$\overline{\mathbf{Sol}}$	$\overline{\mathbf{Sol}_q}$	$\overline{\mathbf{H}}$
finite words	[17]	[16]	[18],**new**	[18],**new**	**new, unless $\mathbf{H} \subseteq \mathbf{Ab}$**
ω-words	[5]	**new**	**new**	**new**	**new, unless $\mathbf{H} = \mathbf{1}$**

As a byproduct we were able to give a simple decomposition of the monoids in $\overline{\mathbf{H}}$ as local Rees extensions and groups in $\mathbf{H}$, using only exponentially many operations.

References

1 Jorge Almeida and Ondřej Klíma. On the irreducibility of pseudovarieties of semigroups. *Journal of Pure and Applied Algebra*, 220(4):1517–1524, 2016. doi:10.1016/j.jpaa.2015.09.015.

2 André Arnold. A syntactic congruence for rational ω-languages. *Theoretical Computer Science*, 39:333–335, 1985.

3 Jean-Camille Birget and John L. Rhodes. Almost finite expansions of arbitrary semigroups. *Journal of Pure and Applied Algebra*, 32(3):239–287, 1984.

4 Jean-Camille Birget and John L. Rhodes. Group theory via global semigroup theory. *Journal of Algebra*, 120(2):284–300, 1989. doi:10.1016/0021-8693(89)90199-3.

5 Volker Diekert and Manfred Kufleitner. Omega-rational expressions with bounded synchronization delay. *Theory Comput. Syst.*, 56:686–696, 2015.

6 Volker Diekert and Manfred Kufleitner. A survey on the local divisor technique. *Theoretical Computer Science*, 610:13–23, 2015. doi:10.1016/j.tcs.2015.07.008.

7 Volker Diekert, Manfred Kufleitner, and Pascal Weil. Star-free languages are Church-Rosser congruential. *Theoretical Computer Science*, 454:129–135, 2012. doi:10.1016/j.tcs.2012.01.028.

8 Volker Diekert and Grzegorz Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.

9 Samuel Eilenberg. *Automata, Languages, and Machines*, volume B. Academic Press, New York and London, 1976.

10 Dominique Perrin. Recent results on automata and infinite words. In *Mathematical foundations of computer science, 1984 (Prague, 1984)*, volume 176 of *Lecture Notes in Comput. Sci.*, pages 134–148. Springer, Berlin, 1984.

11 Dominique Perrin and Jean-Éric Pin. *Infinite words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, Amsterdam, 2004.

12 Jean-Éric Pin. *Varieties of Formal Languages*. North Oxford Academic, London, 1986.

13 John Rhodes and Dennis Allen. Synthesis of the classical and modern theory of finite semigroups. *Advances in Mathematics*, 11(2):238–266, 1973. doi:10.1016/0001-8708(73)90010-8.

14 John L. Rhodes and Benjamin Steinberg. *The* q*-theory of finite semigroups*. Springer Monographs in Mathematics. Springer, 2009.

15 Marcel-Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.

16 Marcel-Paul Schützenberger. Sur les monoides finis dont les groupes sont commutatifs. *Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge*, 8(R-1):55–61, 1974.

17 Marcel-Paul Schützenberger. Sur certaines opérations de fermeture dans les langages rationnels. In *Symposia Mathematica, Vol. XV (Convegno di Informatica Teorica, INDAM, Roma, 1973)*, pages 245–253. Academic Press, 1975.

18 Howard Straubing. Families of recognizable sets corresponding to certain varieties of finite monoids. *Journal of Pure and Applied Algebra*, 15(3):305–318, 1979.

19 Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity.* Birkhäuser, Boston, Basel and Berlin, 1994.

20 Howard Straubing, Denis Thérien, and Wolfgang Thomas. Regular languages defined with generalized quantifiers. *Inform. and Comput.*, 118(2):289–301, 1995.

21 Wolfgang Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V., 1990.

22 Thomas Wilke. An algebraic theory for regular languages of finite and infinite words. *International Journal of Algebra and Computation*, 3(4):447–489, 1993.

An Optimal Dual Fault Tolerant Reachability Oracle*†

Keerti Choudhary

Department of Computer Science and Engineering, IIT Kanpur, Kanpur, India
keerti@cse.iitk.ac.in

Abstract

Let $G = (V, E)$ be an n-vertices m-edges directed graph. Let $s \in V$ be any designated source vertex. We address the problem of reporting the reachability information from s under two vertex failures. We show that it is possible to compute in polynomial time an $O(n)$ size data structure that for any query vertex v, and any pair of failed vertices f_1, f_2, answers in $O(1)$ time whether or not there exists a path from s to v in $G \setminus \{f_1, f_2\}$.

For the simpler case of single vertex failure such a data structure can be obtained using the dominator-tree from the celebrated work of Lengauer and Tarjan [TOPLAS 1979, Vol. 1]. However, no efficient data structure was known in the past for handling more than one failures. We, in addition, also present a labeling scheme with $O(\log^3 n)$-bit size labels such that for any $f_1, f_2, v \in V$, it is possible to determine in poly-logarithmic time if v is reachable from s in $G \setminus \{f_1, f_2\}$ using only the labels of f_1, f_2 and v.

Our data structure can also be seen as an efficient mechanism for verifying double-dominators. For any given $x, y, v \in V$ we can determine in $O(1)$ time if the pair (x, y) is a double-dominator of v. Earlier the best known method for this problem was using dominator chain from which verification of double-dominators of only a single vertex was possible.

1998 ACM Subject Classification G.2.2 Graph Algorithms

Keywords and phrases Fault tolerant, Directed graph, Reachability oracle, Labeling scheme

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.130

1 Introduction

Networks in most real life applications are prone to failures. These failures, though unpredictable, are transient due to some simultaneous repair process that is undertaken in the application. This motivates the research on designing fault tolerant structures for various graph problems. In the past few years, a lot of work has been done in designing fault tolerant structures for various fundamental graph problems, see [7, 9, 14].

We address the problem of building a compact data structure for answering reachability queries from a designated source on vertex failures. The only previous known result for this problem was for single failure. The single fault tolerant reachability is closely related to the notion of *dominators* as follows. Given a directed graph G and a source vertex s, we say that a vertex x dominates a vertex v if every path from s to v contains x. Lengauer and Tarjan [12] introduced a data structure called *dominator tree* which is a tree rooted at s such that for any v in G, the set of ancestors of v in the tree is precisely the set of dominators of v. Thus, for any two vertices x and v in G, v becomes unreachable from s on failure of x if and

* This work was partially supported by Google India PhD Fellowship Award.

† Full version of this article is available at http://www.cse.iitk.ac.in/users/keerti/papers/icalp2016.pdf.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 130; pp. 130:1–130:13

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

only if x is ancestor of v in the dominator tree. A lot of work has been done on computing dominators in optimal and near-optimal time, see [3, 10, 11, 12].

In this work, we focus on building efficient data structures for answering reachability queries upon two vertex failures. One simple solution to this problem is an $O(n)$ space and $O(n)$ query time data structure using the work of Baswana et. al [1] on fault tolerant reachability subgraphs (FTRS). They show that for every $k \geqslant 1$, we can compute in polynomial time a subgraph H of G with at most $2^k n$ edges that preserves the reachability information from s even after k edge/vertex failures. Such a subgraph is referred as a k-FTRS for G. Thus in this case a 2-FTRS serves as our data structure for answering reachability queries. To answer a reachability query after two vertex failures, we can run any standard graph traversal algorithm (BFS/DFS) from s in the 2-FTRS avoiding the failed vertices.

Another extreme solution to this problem is an $O(n^2)$ space data structure with $O(1)$ query time. This is possible by building n dominator trees, one for each graph $G \setminus \{y\}$, where $y \in V$. Now on failure of vertices x, y, a vertex v will be reachable from s if and only if x is not ancestor of v in the dominator tree of $G \setminus \{y\}$. To the best of our knowledge, these are the only known solutions for the dual failure case.

This brings us to the following central question of our work - Is it possible to achieve the best of the above two results? In other words, can we have an oracle of $O(n)$ space that still answers reachability queries in $O(1)$ time? We give an affirmative answer to this question. We also present a labeling scheme for this problem. Our results are summarized as follows:

Oracle: There exists a data structure of $O(n)$ size that given any two failing vertices f_1, f_2 and a query vertex v, takes $O(1)$ time to determine if v is reachable from s in $G \setminus \{f_1, f_2\}$.

Labeling Scheme: There exists a compact labeling scheme for answering reachability queries under two failures. Each vertex stores a label of $O(\log^3 n)$ bits such that for any two failing vertices f_1, f_2 and any destination vertex v, it is possible to determine whether v is reachable from s in $G \setminus \{f_1, f_2\}$ by processing the labels associated with f_1, f_2 and v only.

Our result also implies a data structure for the closely related problem of double dominator verification. A pair of vertices (x, y) is said to be *double-dominator* of a vertex v if each path from s to v contains either x or y, but none of x and y are dominators of v. Using our data structure together with the dominator-tree of Lengauer and Tarjan [12], one can obtain an $O(n)$ space data structure that for any given triplet $x, y, v \in V$ verifies in $O(1)$ time if (x, y) is double-dominator of v. The best previously known result for this could verify double-dominators only for a fixed s, v pair in $O(1)$ time using an $O(n)$ space data structure called dominator chain [16].

1.1 Related work

Demetrescu et al. [7] showed that any oracle for reporting distances from a single source, rather than just the reachability information upon vertex failures in a directed weighted graph must require $\Omega(m)$ space. Nonetheless, for the problem of reporting distances between an arbitrary pair of vertices, they give a construction of $O(n^2 \log n)$ size data structure that for any $u, v, x \in V$ reports the length of the shortest path from u to v avoiding x in constant time. Duan and Pettie [9] extended this result to dual failures by designing a data structure of $O(n^2 \log^3 n)$ space which could answer distance queries after any two vertex failures in an $O(\log n)$ time.

Parter and Peleg [14] addressed the problem of computing a sparse subgraph that preserves the distances from source s on failure of a single vertex. In particular, they show that for any given unweighted graph G we can compute a subgraph H with $O(n^{3/2})$ edges such that

for any two vertices v, x, the distance of v from s in the graph $H \setminus \{x\}$ is the same as that in $G \setminus \{x\}$. They also show that this bound is tight. Parter [13] extended this result to dual failure in undirected graphs by showing an upper bound of $O(n^{5/3})$, and also showed that this bound is tight. Baswana and Khanna [2] showed that for the undirected and single failure case if one is willing to settle for an approximation then there is a subgraph with $O(n \log n)$ edges that preserves the distances up to a multiplicative error of 3. Parter and Peleg [15] improved this result and obtained a subgraph with at most $3n$ edges.

Another closely related problem is the replacement paths problem. In this problem we are given a source s and a target t and for each edge e on the shortest path from s to t the algorithm computes the shortest path from s to t avoiding e. Many variants of this problem were studied along the years. For a recent overview see [17] and references therein.

The questions of finding graph spanners, approximate distance oracles and compact routing schemes that are resilient to f vertex or edge failures in undirected graphs were studied in [8, 5, 4].

1.2 Our Techniques

Consider a reachability tree T rooted at source s. Let v be any vertex in T, and P be the path from s to v in T. Let us first consider the simple case when only a single vertex, say x, fails in G. If x lies on P and v is still reachable from s, then we can find an alternate path consisting of - a prefix of P, followed by a "detour", say D, avoiding P (and x), followed by a suffix of P. This simple decomposition can be used to easily handle reachability queries for one vertex failure. However, in the case of the dual failure, non trivialities arise when the second failing vertex lies on detour D. A natural direction to proceed from here is to define secondary detours which are disjoint from both P as well as D, but handling secondary detours is quite complicated.

So we take an alternative approach in which instead of initially starting with a single tree we begin with two independent trees - T_1 and T_2, defined by Georgiadis and Tarjan [11]. They satisfy the condition that for any v, the path from s to v in two trees intersect only at the dominators of v. This allows us to reduce the problem to the case when exactly one failure is an ancestor of v in T_1, and the other failure is ancestor of v in T_2. Now let P_1, P_2 be the paths from s to v in T_1, T_2, and let f_1, f_2 be the failed vertices lying respectively on P_1, P_2. Then, the structure of an alternate path to v gets simplified as follows: It consists of a prefix of either P_1 or P_2 up to a vertex a (lying respectively before f_1 or f_2) followed by a detour which is disjoint from P_1, P_2 (and f_1, f_2), and followed again by a suffix of P_1 or P_2. Note that the starting and terminating vertices of detour need not lie on the same tree-path.

So, we get a very clean and simple representation of detours. The main challenge lies in how to efficiently search for an appropriate detour avoiding f_1, f_2. The problems arising and how they are tackled is discussed in detail in Section 4.

1.3 Organization of the paper

We describe notations and terminologies in Section 2. In Section 3, we briefly sketch the solution for the single failure case using detours. The overview of the paper is presented in Section 4. In Section 5, we present the reachability oracle for a simpler class of graphs that are 2-vertex strongly connected. In Section 6, we present the oracle for general graphs. The construction of the labeling scheme can be found in the full version of the paper.

2 Preliminaries

Given a directed graph $G = (V, E)$ on $n = |V|$ vertices and $m = |E|$ edges, and a source vertex $s \in V$, the following notations will be used throughout the paper.

- f_1, f_2: A given pair of failed vertices.
- $par_T(x)$: The parent of vertex x in tree T.
- $depth_T(x)$: The depth of vertex x in tree T.
- $\textsc{Path}_T(x, y)$: The path from vertex x to vertex y in tree T.
- $\textsc{Path}_T(\bar{x}, y)$: $\textsc{Path}_T(x, y) \setminus \{x\}$
- $\textsc{Path}_T(x, \bar{y})$: $\textsc{Path}_T(x, y) \setminus \{y\}$
- $\textsc{Path}_T(\bar{x}, \bar{y})$: $\textsc{Path}_T(x, y) \setminus \{x, y\}$
- $idom(x)$: The immediate dominator of vertex x. (See Definition 1).
- T_1, T_2: A pair of independent trees for G rooted at s. (See Definition 2).
- $P[x, y]$: The subpath of path P lying between vertices x, y, assuming x precedes y on P.
- $P::Q$: The path formed by concatenating paths P and Q in G. Here it is assumed that the last vertex of P is the same as the first vertex of Q.

We now define immediate dominators and independent spanning trees which are crucial to our fault tolerant reachability oracle.

▶ **Definition 1** ([12]). A vertex w is said to be the *immediate dominator* of v if (i) w is a dominator of v, and (ii) every dominator of v (other than v itself) is also a dominator of w.

▶ **Definition 2** (Georgiadis and Tarjan [11]). Given a directed graph G and a designated source s, a pair of trees T_1, T_2 rooted at s are said to be *independent spanning trees* if for each $v \neq s$ the paths from s to v in T_1 and T_2 intersect only at the dominators of v. (It was shown by Georgiadis and Tarjan [11] that such a pair of trees can be computed in an $O(m)$ time).

Below we state a few basic properties of dominators in a directed graph.

▶ **Property 1.** *Let T be a reachability tree rooted at s, and y_0, y_1 be vertices such that $y_0 = idom(y_1)$. Then for any $z \in \textsc{Path}_T(\bar{y}_0, y_1)$, $idom(z)$ belongs to $\textsc{Path}_T(y_0, y_1)$.*

▶ **Property 2.** *Let T be a reachability tree rooted at s, and y_1, y_2 be vertices such that y_1 is ancestor of y_2, and $idom(y_1) = idom(y_2)$. Then for any $z \in \textsc{Path}_T(y_1, y_2)$ either y_1 is a dominator of z or $idom(z) = idom(y_1)$.*

For efficient implementation of our oracle, we use the following optimal result by Demaine et. al [6] for answering the range minima queries on tree paths.

▶ **Theorem 3** (Demaine et al. [6]). *A tree T on n vertices and edge weights in the range $[0, n^3]$ can be preprocessed in $O(n \log n)$ time to build a data structure of $O(n)$ size so that given any $u, v \in T$, the edge of smallest weight on the tree path from u to v can be reported in $O(1)$ time.*

The following corollary is immediate from Theorem 3.

▶ **Corollary 4.** *Given a tree, say T, on n vertices, with each vertex assigned an integral weight in range $[0, n^3]$, we can obtain in polynomial time an $O(n)$ size data structure that for any two vertices x, y, outputs in $O(1)$ time the vertex with minimum weight on $\textsc{Path}_T(\bar{x}, y)$.*

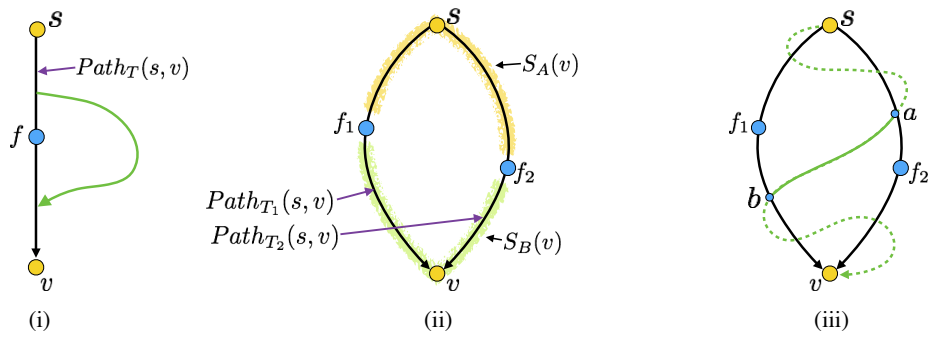


Figure 1 (i) Representation of a path bypassing f in the single vertex failure case; (ii) Representation of sets $S_A(v)$ and $S_B(v)$ when condition $\mathcal{C}$ is satisfied for vertex v; (ii) A path from $a \in S_A(v)$ to $b \in S_B(v)$ when v is reachable from s in $G \setminus \{f_1, f_2\}$.

3 Review of reachability oracle for single failure

In order to understand the dual fault tolerant reachability oracle we first briefly discuss the case of single failure. We here describe an alternative reachability oracle using detours instead of dominator tree. Let T be any reachability tree of G rooted at s, and f, v be respectively the failed and the query vertex. Also assume f is an ancestor of v in T. Notice that if v is reachable from s in $G \setminus \{f\}$, then there must exist a path starting from $\textsc{Path}_T(s, \bar{f})$ and terminating at $\textsc{Path}_T(\bar{f}, v)$ which, except for its endpoints, does not pass through any ancestor of v in T. (See Figure 1(i)). So for each $w \in V$, we can define a detour $D(w)$ to be a path starting from the highest possible ancestor of w in T and terminating at w such that none of the internal vertices of the path pass through an ancestor of w. Now on failure of f it suffices to search whether there exists a vertex lying in $\textsc{Path}_T(\bar{f}, v)$ whose detour starts from an ancestor of f. This can be achieved by assigning to each vertex w a weight equal to the depth of the first vertex on $D(w)$. By doing this the problem of reachability under one vertex failure reduces to the problem of solving range minima on weighted trees, for which already an optimal solution exists. (See Corollary 4).

4 Overview

Let us consider the failure of a pair of vertices f_1, f_2 in G, and let v be the query vertex. Note that if any of the tree paths - $\textsc{Path}_{T_1}(s, v)$ or $\textsc{Path}_{T_2}(s, v)$ is intact, then v will be reachable from s. Also, if both $\textsc{Path}_{T_1}(s, v)$ or $\textsc{Path}_{T_2}(s, v)$ contains a common failed vertex, say f_1, then v will not be reachable from s. This is because then f_1 would be a dominator of v. Thus the non-trivial case is when $\textsc{Path}_{T_1}(s, v)$ contains only f_1 and $\textsc{Path}_{T_2}(s, v)$ contains only f_2, or the vice-versa. So whenever a query vertex v is given to us, we may assume that the following condition is satisfied.

$$\mathcal{C}: f_1 \text{ lies on } \textsc{Path}_{T_1}(s, v) \setminus \textsc{Path}_{T_2}(s, v), \text{ and } f_2 \text{ lies on } \textsc{Path}_{T_2}(s, v) \setminus \textsc{Path}_{T_1}(s, v).$$

Now consider the sets $S_A(v)$ and $S_B(v)$ as defined below. (For a better understanding of these sets see Figure 1(ii)).

- $S_A(v)$: Set of vertices lying either above f_1 on $\textsc{Path}_{T_1}(s, v)$ or above f_2 on $\textsc{Path}_{T_2}(s, v)$.
- $S_B(v)$: Set of vertices lying either below f_1 on $\textsc{Path}_{T_1}(s, v)$ or below f_2 on $\textsc{Path}_{T_2}(s, v)$.

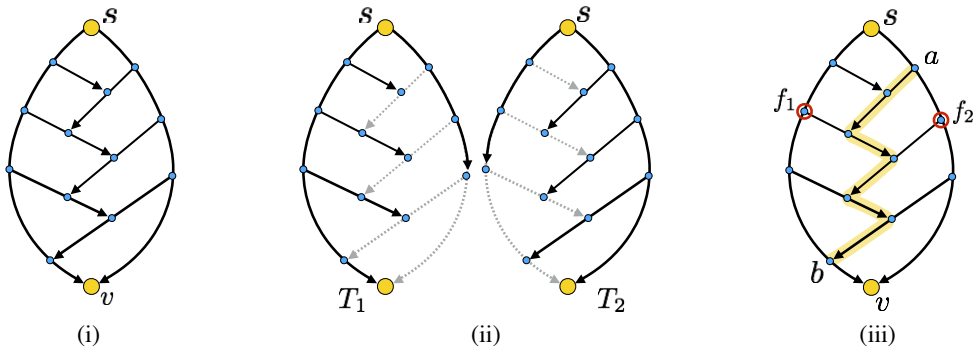


Figure 2 (i) A graph G with in-degree of each vertex bounded by two; (ii) A pair of independent spanning trees T_1 and T_2 for G; (iii) A path P from $a \in S_A(v)$ to $b \in S_B(v)$ in $G \setminus \{f_1, f_2\}$.

It turns out that if v is reachable from s in $G \setminus \{f_1, f_2\}$, then there must exists a path from set $S_A(v)$ to $S_B(v)$ avoiding the vertices of both $\text{PATH}_{T_1}(s, v)$ and $\text{PATH}_{T_2}(s, v)$. This fact is formally stated in the following lemma (refer to the full version of the paper for its proof).

▶ **Lemma 5.** *Given a pair of failed vertices f_1, f_2, a vertex v is reachable from s if and only if G contains a path P satisfying the following conditions. (See Figure 1(iii)).*
C1. *The first and last vertices of P lies respectively in sets $S_A(v)$ and $S_B(v)$.*
C2. *None of the internal vertices of P lies on $\text{PATH}_{T_1}(s, v)$ or $\text{PATH}_{T_2}(s, v)$.*

For simplicity we refer to a path satisfying the conditions *C1* and *C2* stated in the above lemma as an $S_{A,B}(v)$ path. In order to efficiently compute such a path we define a pair of detours $D^1(w)$ and $D^2(w)$ for each vertex $w \in V$ as follows.

- $\text{D}^i(w)$: a path starting from the highest possible ancestor of w in T_i and terminating at w such that none of the internal vertices of the path are ancestor of w in T_1 or T_2.

Note that the detours $D^1(w)$ and $D^2(w)$ can be seen as a simple generalization of the detour $D(w)$ which was defined for the single failure case in Section 3. However, we show that this simple generalization is not sufficient to answer the reachability queries in dual failure. To understand this subtle point consider an $S_{A,B}(v)$ path with a, b as its endpoints. If the endpoint b is equal to v, then P could be simply either $D^1(v)$ or $D^2(v)$. The problem arises when $b \neq v$. This is because if b is an ancestor of v in T_1, then P might contain vertices from $\text{PATH}_{T_2}(s, b)$. (Recall that the internal vertices of P are disjoint from $\text{PATH}_{T_2}(s, v)$, but not necessarily disjoint from $\text{PATH}_{T_2}(s, b)$). So in this case P can neither be $D^1(b)$ nor be $D^2(b)$. For a more clear insight into this consider the graph and its two independent spanning trees in Figure 2. Since in-degree of each vertex in the graph is at most two, $\text{D}^i(w)$ for each $w \in V$ is simply the incoming edge from $par_i(w)$ to w. Thus the path P connecting $S_A(v)$ to $S_B(v)$ is a concatenation of as many detours as there are number of edges in P. Determining whether a concatenation of all these single-edge detours can give us an $S_{A,B}(v)$ path is difficult to achieve in $O(1)$ time.

This shows that a simple generalization of detours from a single tree to two trees is not sufficient. To tackle the problem we extend the notion of detours to 'Parent Detours' and 'Ancestor Detours'. These detours unlike the normal detour terminates at an appropriate ancestor of w in T_1 or T_2. We formally define the parent-detours and ancestor-detours in the following sections, and show how they can be used to solve the problem of dual fault tolerant reachability.

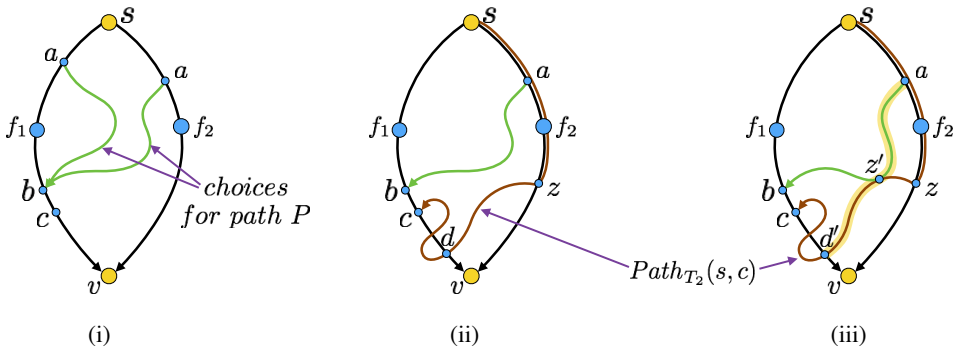


Figure 3 (i) Possibilities for path P when $b \in \text{PATH}_{T_1}(\bar{f_1}, \bar{v})$; (ii) Representation of $\text{PATH}_{T_2}(s, c)$ and $z = LCA(c, v)$ in T_2; (iii) Violation of assumption 2 if $P \cap \text{PATH}_{T_2}(z, c)$ is non-empty.

5 Reachability oracle for 2-vertex strongly connected graphs

In this section we describe an $O(n)$ space and $O(1)$ time reachability oracle for 2-vertex strongly connected graphs. By 2-vertex strongly connectedness we have that on removal of any vertex f ($f \neq s$), all the vertices in $G \setminus \{f\}$ are still reachable from s. Thus each vertex is dominated only by source s and by itself. This implies that for any vertex w, $\text{PATH}_{T_1}(s, w)$ and $\text{PATH}_{T_2}(s, w)$ intersects only at the endpoints s, w.

Consider a query vertex v which is reachable from s in $G \setminus \{f_1, f_2\}$. Let us assume that condition $\mathcal{C}$ is satisfied for v. Let P be any $S_{A,B}(v)$ path, and a, b be respectively the first and last vertices on P. Without loss of generality we can assume that b lies on $\text{PATH}_{T_1}(s, v)$. See Figure 3(i). We make the following additional assumptions.

1. None of the $S_{A,B}(v)$ paths terminates at v (i.e. b cannot be v).
2. b is the lowest vertex on $\text{PATH}_{T_1}(s, v)$ at which an $S_{A,B}(v)$ path terminates.

▶ Remark. The assumption 1 is justified since if $b = v$, then v will be reachable from s using the detours $D^1(v)$ or $D^2(v)$.

We now state a lemma which provides the motivation for defining the parent detours.

▶ **Lemma 6.** *Let a, b, P be as described above, and c be the child of b on $\text{PATH}_{T_1}(s, v)$. Then,*

(i) *Vertex f_2 is an ancestor of c in T_2.*

(ii) *None of the internal vertices of P lie on $\text{PATH}_{T_1}(s, c)$ or $\text{PATH}_{T_2}(s, c)$.*

Proof. Let z denote the LCA of vertices c and v in tree T_2. (See Figure 3(ii)). Consider the path $Q = \text{PATH}_{T_2}(z, c)$. It is easy to see that none of the internal vertices of Q lies on $\text{PATH}_{T_2}(s, v)$. Also, the internal vertices of Q appearing on $\text{PATH}_{T_1}(s, v)$ must lie below c on $\text{PATH}_{T_1}(s, v)$. This is because, by definition of independent spanning trees, $\text{PATH}_{T_1}(s, c)$ and Q can intersect only at the vertices s and c.

We now prove claim 1. Let d be the first point of intersection of Q with $\text{PATH}_{T_1}(c, v)$. (See Figure 3(ii)). Then the internal vertices of $Q[z, d]$ are disjoint from both $\text{PATH}_{T_1}(s, v)$ and $\text{PATH}_{T_2}(s, v)$. Now if z is an ancestor of f_2 in T_2, then $Q[z, d]$ forms an $S_{A,B}(v)$ path, terminating at descendant of b in T_1, thereby violating assumption 2. Hence f_2 must be either same as z or an ancestor of z. This shows that f_2 is an ancestor of c in T_2.

In order to prove claim 2, we first show that P is disjoint from Q. Let us suppose on the contrary, that there exists a vertex, say z', belonging to $P \cap Q$. Also let d' be the first vertex of $Q[z', c]$ lying on $\text{PATH}_{T_1}(c, v)$. (See Figure 3(iii)). Then $P[a, z']::Q[z', d']$ forms an

$S_{A,B}(v)$ path terminating at a descendant of b in T_1. This again violates assumption 2. Thus $P \cap Q = \emptyset$. Now since the internal vertices of P are disjoint from $\textsc{Path}_{T_1}(s,v), \textsc{Path}_{T_2}(s,v)$, they must be disjoint from $\textsc{Path}_{T_1}(s,c)$ and $\textsc{Path}_{T_2}(s,z)::Q = \textsc{Path}_{T_2}(s,c)$, as well. ◀

The above lemma implies that f_1 is an ancestor of c in T_1, and f_2 is an ancestor of c in T_2. Thus $S_A(v) = S_A(c)$. Hence we have the following corollary.

▶ **Corollary 7.** *P is an $S_{A,B}(c)$ path terminating at $par_{T_1}(c)$.*

In order to capture the above fact we define parent-detours for each $w \in V$ which instead of terminating at w terminates at either $par_{T_1}(w)$ or $par_{T_2}(w)$.

- $\text{PD}^i_j(w)$: a path starting from the highest possible ancestor of w in T_i and terminating at $par_{T_j}(w)$ s.t. none of the internal vertices of the path lie on $\textsc{Path}_{T_1}(s,w)$ or $\textsc{Path}_{T_2}(s,w)$.

By above definition of parent-detour it follows that P can be replaced by either $\text{PD}^1_1(c)$ or $\text{PD}^2_1(c)$ depending upon whether it starts from an ancestor of c in T_1 or T_2. Now let x_1 denote the child of f_1 in T_1 lying on $\textsc{Path}_{T_1}(s,v)$, and x_2 denote the child of f_2 in T_2 lying on $\textsc{Path}_{T_2}(s,v)$. Then the parent-detours of vertices x_1, x_2 may not be of any help, since they would terminate at f_1 and f_2. However, the parent-detours of vertices in $S_B(v) \setminus \{x_1, x_2\}$ will suffice to determine whether v is reachable from s or not.

Notice that in above discussion, we observed that P is an $S_{A,B}(c)$ path terminating at $b = par_{T_1}(c)$. This shows that for vertices lying on $\textsc{Path}_{T_1}(\bar{x}_1, v)$, we only need to worry about parent-detours terminating at $par_{T_1}(\cdot)$, i.e. $\text{PD}^1_1(\cdot)$ and $\text{PD}^2_1(\cdot)$. Whereas, for vertices on $\textsc{Path}_{T_2}(\bar{x}_2, v)$, we need to worry about parent-detours terminating at $par_{T_2}(\cdot)$, i.e. $\text{PD}^1_2(\cdot)$ and $\text{PD}^2_2(\cdot)$. We thus have the following lemma.

▶ **Lemma 8.** *Let x_1 and x_2 be as defined above. A vertex v is reachable from s in $G \setminus \{f_1, f_2\}$ if and only if at least one of the following vertices lie in $S_A(v)$.*

(i) *The first vertex of $\text{D}^1(v)$ or $\text{D}^2(v)$.*
(ii) *The first vertex of either $\text{PD}^1_1(w)$ or $\text{PD}^2_1(w)$ for some $w \in \textsc{Path}_{T_1}(\bar{x}_1, v)$.*
(iii) *The first vertex of either $\text{PD}^1_2(w)$ or $\text{PD}^2_2(w)$ for some $w \in \textsc{Path}_{T_2}(\bar{x}_2, v)$.*

5.1 Implementation of the oracle

We first introduce the following notations for detours and parent detours.

- $\beta^i(v)$: $depth_{T_i}$(first vertex on $\text{D}^i(v)$).
- $\gamma^i_j(v)$: $depth_{T_i}$(first vertex on $\text{PD}^i_j(v)$).

Now let f_1, f_2 be a given pair of failed vertices and v be a given query vertex. Our first step is to check if condition $\mathcal{C}$ is satisfied. Recall that this requires only verifying the ancestor-descendant relationship in trees T_1 and T_2. One simple method to achieve this for any given tree T is to perform the pre-order and the post-order traversal of T, and store the vertices in the order they are visited. Now x will be ancestor of y in T if and only if x appears before y in the pre-order traversal, and after y in the post-order traversal.

Algorithm 1 presents the pseudo-code for answering reachability query for a vertex v assuming condition $\mathcal{C}$ is satisfied. This can be explained in words as follows. For $i = 1, 2$, we first check if $D^i(v)$ starts from an ancestor of f_i in T_i or not. This is done by comparing the value of $\beta^i(v)$ with the depth of f_i in T_i. Next we compute the vertices x_1, x_2. Finally for $i, j \in \{1, 2\}$, we compute a vertex $w \in \textsc{Path}_{T_j}(\bar{x}_j, v)$ for which $\gamma^i_j(\cdot)$ is minimum. If $\gamma^i_j(w)$ is less than the depth of f_i in T_i, then it implies that $\text{PD}^i_j(w)$ starts from an ancestor of f_i in T_i, so we return True. If we reach to the end of code, that means we have not been able to find any path for v, so we return False.

Algorithm 1: Oracle for reachability to v in 2-vertex strongly connected graphs.

1 **if** $\beta^1(v) < depth_{T_1}(f_1)$ *or* $\beta^2(v) < depth_{T_2}(f_2)$ **then** Return True;
2 ;
3 $x_1 \leftarrow$ the vertex with minimum depth on $\text{PATH}_{T_1}(\bar{f}_1, v)$;
4 $x_2 \leftarrow$ the vertex with minimum depth on $\text{PATH}_{T_2}(\bar{f}_2, v)$;
5 **foreach** $i, j \in \{1, 2\}$ **do**
6 $\quad w \leftarrow$ a vertex on $\text{PATH}_{T_j}(\bar{x}_j, v)$ for which $\gamma^i_j(\cdot)$ is minimum;
7 $\quad$ **if** $\gamma^i_j(w) < depth_{T_i}(f_i)$ **then** Return True;
8 $\quad$;
9 **end**
10 Return False;

The above oracle can be easily implemented in $O(1)$ time, by having a total of six weight functions – one each for storing the depth of a vertex in trees T_1, T_2, and the other four for storing the values $\gamma^i_j(\cdot)$, for $i, j \in \{1, 2\}$. By doing this the vertices x_1, x_2 can be computed in constant time since they are respectively the vertices with minimum depth on the paths $\text{PATH}_{T_1}(\bar{f}_1, v)$ and $\text{PATH}_{T_1}(\bar{f}_1, v)$. Also Step 4 can be carried out in an $O(1)$ time. We can thus state the following theorem.

▶ **Theorem 9.** *A 2-vertex strongly connected graph on n vertices can be preprocessed in polynomial time for a given source vertex s to build a data structure of $O(n)$ size such that for any query vertex v, and pair of failures f_1, f_2, it takes $O(1)$ time to determine if there exists any path from s to v in $G \setminus \{f_1, f_2\}$.*

6 Reachability oracle for general graphs

In this section we explain the reachability oracle for general graphs. Consider a query vertex u in G. Let $u_0, u_1, ..., u_k$ be the dominators of u with $u_0 = s$ and $u_k = u$. Thus $\text{PATH}_{T_1}(s, u)$ and $\text{PATH}_{T_2}(s, u)$ intersect only at u_i's. (See Figure 4(i)). As in Section 5, we assume that condition $\mathcal{C}$ holds for u, so none of the u_i's can be equal to f_1 or f_2. Now let $i, j \in [1, k]$ be such that $f_1 \in \text{PATH}_{T_1}(\bar{u}_{i-1}, \bar{u}_i)$ and $f_2 \in \text{PATH}_{T_2}(\bar{u}_{j-1}, \bar{u}_j)$. It is easy to see that if $i \neq j$, then u is reachable from s by the path $\text{PATH}_{T_1}(s, u_{i-1})::\text{PATH}_{T_2}(u_{i-1}, u_i)::\text{PATH}_{T_1}(u_i, u)$. (See Figure 4(ii)). Thus we consider the case when $i = j$ [1]. For simplicity, we use symbols, v and $idom(v)$ to respectively denote the vertices u_i and u_{i-1}. Notice that in order to check reachability of u from s, it suffices to check if v is reachable from s in $G \setminus \{f_1, f_2\}$.

We now divide our analysis into various different cases as follows:

Case 1. *There exists an $S_{A,B}(v)$ path terminating at vertex v.*
In this case v will be reachable from s using either of the detours $D^1(v)$ or $D^2(v)$.

Case 2. *There exists a vertex $w \in S_B(v)$ for which $idom(w) \in S_A(v) \setminus \{idom(v)\}$.*
In this case also we can show that v is reachable from s by the following argument. Without loss of generality let us assume that w is an ancestor of v in T_1. Since $idom(w)$ is an

[1] One can verify in $O(1)$ time whether $u_{i-1} = u_{j-1}$ (i.e. if $i = j$) since vertex $u_{i-1} = LCA(f_1, v)$ and vertex $u_{j-1} = LCA(f_2, v)$ in the dominator tree of G.

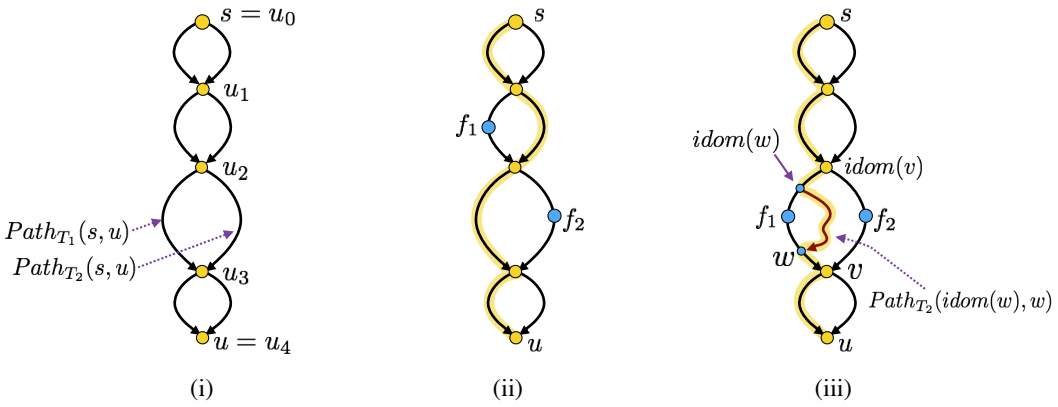


Figure 4 (i) Representation of dominators of u; (ii) A path from s to u (highlighted in yellow) when f_1 lies on $\textsc{Path}_{T_1}(\bar{u}_1, \bar{u}_2)$ and f_2 lies on $\textsc{Path}_{T_2}(\bar{u}_2, \bar{u}_3)$; (iii) A path from s to u (highlighted in yellow) when there exists a vertex $w \in S_B(v)$ for which $idom(w)$ lies in the set $S_A(v) \setminus \{idom(v)\}$.

ancestor of w in T_1, it must lie on $\textsc{Path}_{T_1}(\overline{idom(v)}, \bar{f}_1)$. (See Figure 4(iii)). Consider the path $Q = \textsc{Path}_{T_2}(idom(w), w)$. Note that f_2 cannot lie on Q. This is because otherwise $\textsc{Path}_{T_1}(idom(v), w)$ and $\textsc{Path}_{T_2}(idom(v), f_2)::Q[f_2, w]$ will form two vertex disjoint paths from $idom(v)$ to w, which would violate the fact that $idom(w) \neq idom(v)$. Also f_1 cannot lie on Q, as Q is disjoint from $\textsc{Path}_{T_1}(\overline{idom(w)}, \bar{w})$. Thus v is reachable from s by the path $\textsc{Path}_{T_1}(s, idom(w))::Q::\textsc{Path}_{T_1}(w, v)$.

Case 3. *None of the $S_{A,B}(v)$ path terminates at v, and there does not exist a vertex in $S_B(v)$ whose immediate dominator lies in $S_A(v) \setminus \{idom(v)\}$.*
This is the most non-trivial case of dual fault tolerant reachability oracle. We now provide analysis for this case.

Let us suppose v is reachable from s in $G \setminus \{f_1, f_2\}$. Then without loss of generality we can assume that there exists an $S_{A,B}(v)$ path (say P) terminating at an ancestor of v in T_1. In case there are multiple $S_{A,B}(v)$ paths, then we take P to be that path which terminates at lowest vertex on $\textsc{Path}_{T_1}(\bar{f}_1, \bar{v})$. Let a, b be respectively the first and last vertices on P. By Property 1, we know that $idom(b)$ cannot be an ancestor of $idom(v)$ in T_1. Therefore, $idom(b)$ must be equal to $idom(v)$. This is because $idom(b)$ cannot lie in $S_A(v) \setminus \{idom(v)\}$, and if $idom(b)$ lies in $S_B(v)$ then $P \cap S_B(v)$ will contain both b and $idom(b)$, which would violate the definition of an $S_{A,B}(v)$ path.

Now consider the vertex c which is child of b on $\textsc{Path}_{T_1}(s, v)$. It turns out that in a general graph the parent-detours of c may not be of any help. This is because the analysis for 2-vertex strongly connected graphs crucially exploited the fact that $idom(b) = idom(c) = s$. But in general graphs, if $idom(b)$ is not equal to $idom(c)$, then it can be shown that Lemma 6 no longer holds. To be more precise, we can show that the internal vertices of P might not be disjoint from $\textsc{Path}_{T_2}(s, c)$.

However, the problem can be resolved if we take c to be the first descendant of b on $\textsc{Path}_{T_1}(s, v)$ whose immediate dominator is the same as that of b. This motivates us to define the notion of pseudo-child (and pseudo-parent) as follows.

▶ **Definition 10.** Given a reachability tree T rooted at s, a vertex x is said to be pseudo-parent of y in T (and y is said to be pseudo-child of x) if x is the nearest ancestor of y in T whose immediate dominator is the same as that of y.

Note that in a 2-vertex strongly connected graph, the definition of pseudo-parent and pseudo-child degenerates to normal notion of parent and child. This is because, immediate dominator of all the vertices (other than s) in such graph is equal to s.

We now state a lemma which is an analogue of Lemma 6 for general graphs.

▶ **Lemma 11.** *Let a, b, P be as described above, and c be the pseudo-child of b on $\text{PATH}_{T_1}(s, v)$. Then,*

(i) *Vertex f_2 is an ancestor of c in T_2.*

(ii) *None of the internal vertices of P lie on $\text{PATH}_{T_1}(s, c)$ or $\text{PATH}_{T_2}(s, c)$.*

As a corollary of the above lemma we get that P is an $S_{A,B}(c)$ path terminating at pseudo-parent of c in T_1. We thus define ancestor-detours which are a generalization of parent-detours as follows.

- $\text{AD}^i_j(w)$: a path starting from the highest possible ancestor of w in T_i and terminating at pseudo-parent of w in tree T_j such that none of the internal vertices of the path lie on $\text{PATH}_{T_1}(s, w)$ or $\text{PATH}_{T_2}(s, w)$.

Now let x_1 be the first descendant of f_1 on $\text{PATH}_{T_1}(s, v)$ whose immediate dominator is equal to $idom(v)$. Similarly, let x_2 be the first descendant of f_2 on $\text{PATH}_{T_1}(s, v)$ whose immediate dominator is equal to $idom(v)$. Then the ancestor-detours of x_1, x_2 will not be of any help as they would terminate at either f_1, f_2 or their ancestors. Now as in Section 5, we can argue that ancestor-detours of vertices on $\text{PATH}_{T_1}(\bar{x}_1, v) \cup \text{PATH}_{T_2}(\bar{x}_2, v)$ suffice to answer the reachability query for vertex v. This completes the analysis of the third case.

We thus have the following lemma.

▶ **Lemma 12.** *Let v be a vertex satisfying condition $\mathcal{C}$ such that $f_1 \in \text{PATH}_{T_1}(\overline{idom(v)}, v)$ and $f_2 \in \text{PATH}_{T_1}(\overline{idom(v)}, v)$. Also let x_1 and x_2 be as defined above. Then v is reachable from s in $G \setminus \{f_1, f_2\}$ if and only if either of the following statements holds true.*

(i) [Case 1] *The first vertex of $\text{D}^1(v)$ or $\text{D}^2(v)$ lies in $S_A(v)$.*

(ii) [Case 2] *There exists a vertex $w \in S_B(v)$ for which $idom(w) \in S_A(v) \setminus \{idom(v)\}$.*

(iii) [Case 3] *There exists a vertex $w \in \text{PATH}_{T_1}(\bar{x}_1, v)$ such that $idom(w) = idom(v)$ and the first vertex of either $\text{AD}^1_1(w)$ or $\text{AD}^2_1(w)$ lies in $S_A(v)$.*

(iv) [Case 3] *There exists a vertex $w \in \text{PATH}_{T_2}(\bar{x}_2, v)$ such that $idom(w) = idom(v)$ and the first vertex of either $\text{AD}^1_2(w)$ or $\text{AD}^2_2(w)$ lies in $S_A(v)$.*

6.1 Implementation of the oracle

We now explain the implementation of reachability oracle for general graphs. As in Section 5, we define the following notations.

- $\alpha^i(v)$: $depth_{T_i}(idom(v))$.
- $\beta^i(v)$: $depth_{T_i}(\text{first vertex on } \text{D}^i(v))$.
- $\gamma^i_j(v)$: $depth_{T_i}(\text{first vertex on } \text{AD}^i_j(v))$.

Let f_1, f_2 be a given pair of failed vertices and v be a given query step. We assume that condition $\mathcal{C}$ is satisfied, and failures f_1, f_2 lies respectively on $\text{PATH}_{T_1}(idom(v), v)$ and $\text{PATH}_{T_2}(idom(v), v)$. We first check for $i = 1, 2$, if $D^i(v)$ starts from an ancestor of f_i in T_i or not. This is done by comparing the value of $\beta^i(v)$ with the depth of f_i in T_i.

Next we compute the vertices x_1, x_2 as follows. Recall that x_1 is the highest ancestor of v in $\text{PATH}_{T_1}(\bar{f}_1, v)$ whose immediate dominator is equal to $idom(v)$. So to obtain x_1, we call the range minima query for vertices on $\text{PATH}_{T_1}(\bar{f}_1, v)$ with $\langle \alpha^1(\cdot), depth_{T_1}(\cdot) \rangle$ as the weight function. By comparing the value of $\alpha^1(\cdot)$, it is able to filter out those vertices in

$\textsc{Path}_{T_1}(\bar{f}_1, v)$ whose immediate dominator is at minimum depth in T_1, i.e. it is equal to $idom(v)$. After this it assigns x_1 to be that vertex which has minimum depth in T_1. Vertex x_2 is computed in a similar manner.

Now notice that to find whether there exists a vertex in $S_B(v)$ whose immediate dominator lies in $S_A(v) \setminus \{idom(v)\}$, we only need to restrict ourself to paths $\textsc{Path}_{T_1}(\bar{f}_1, \bar{x}_1)$ and $\textsc{Path}_{T_1}(\bar{f}_1, \bar{x}_1)$. This is because Property 2 implies that immediate dominator of vertices in $\textsc{Path}_{T_1}(x_1, v)$ is either equal to $idom(v)$ or lies in $\textsc{Path}_{T_1}(x_1, v)$ itself. Similarly, for $\textsc{Path}_{T_2}(x_2, v)$. So for $i = 1, 2$, we perform the range minima query to find a vertex, say w, on $\textsc{Path}_{T_i}(\bar{f}_i, \bar{x}_i)$ for which $\alpha^i(w)$ is minimum. If $\alpha^i(w)$ is less than the depth of f_i in T_i, then we report that v is reachable from s.

Finally for $i, j \in \{1, 2\}$, we compute a vertex $w \in \textsc{Path}_{T_j}(\bar{x}_j, v)$ for which $\langle \alpha^i(\cdot), \gamma^i_j(\cdot) \rangle$ is minimum. The term $\alpha^i(\cdot)$ is added in front so that we are able to filter out those vertices whose immediate dominator is equal to $idom(v)$. Now if $\gamma^i_j(w)$ is less than the depth of f_i in T_i, then it implies that $\text{AD}^i_j(w)$ starts from an ancestor of f_i in T_i, so we return True.

If we reach to the end of code, that means we have not been able to find any path for v, so we return False.

Algorithm 2: Oracle for reachability to v in general graphs.

```
1  if β¹(v) < depth_{T1}(f1) or β²(v) < depth_{T2}(f2) then Return True;
2  ;
3  x1 ← a vertex on Path_{T1}(f̄1, v) for which ⟨α¹(·), depth_{T1}(·)⟩ is minimum;
4  x2 ← a vertex on Path_{T2}(f̄2, v) for which ⟨α²(·), depth_{T2}(·)⟩ is minimum;
5  foreach i ∈ {1, 2} do
6  |   w ← a vertex on Path_{Ti}(f̄i, x̄i) for which αⁱ(·) is minimum;
7  |   if αⁱ(w) < depth_{Ti}(fi) then Return True;
8  |   ;
9  end
10 foreach i, j ∈ {1, 2} do
11 |   w ← a vertex on Path_{Tj}(x̄j, v) for which ⟨αⁱ(·), γⁱⱼ(·)⟩ is minimum;
12 |   if γⁱⱼ(w) < depth_{Ti}(fi) then Return True;
13 |   ;
14 end
15 Return False;
```

As in Algorithm 1, we can argue that the Steps 2, 3, 5 and 9, can be implemented in $O(1)$ time. Thus Algorithm 2 takes constant time to answer reachability queries. We thus conclude with the following theorem.

▶ **Theorem 13.** *A directed graph $G = (V, E)$ on n vertices can be preprocessed in polynomial time for a given source vertex $s \in V$ to build a data structure of $O(n)$ size such that for any $f_1, f_2, v \in V$, it takes $O(1)$ time to determine if there exists a path from s to v in $G \setminus \{f_1, f_2\}$.*

Acknowledgements. I am very grateful to my advisor, Prof. Surender Baswana, for many helpful discussions and for reviewing this paper.

References

1. Surender Baswana, Keerti Choudhary, and Liam Roditty. Fault tolerant subgraph for single source reachability: Generic and optimal. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 19-21, 2016 (to appear)*, 2016.
2. Surender Baswana and Neelesh Khanna. Approximate shortest paths avoiding a failed vertex: Near optimal data structures for undirected unweighted graphs. *Algorithmica*, 66(1):18–50, 2013.
3. Adam L. Buchsbaum, Loukas Georgiadis, Haim Kaplan, Anne Rogers, Robert Endre Tarjan, and Jeffery Westbrook. Linear-time algorithms for dominators and other path-evaluation problems. *SIAM J. Comput.*, 38(4):1533–1573, 2008.
4. Shiri Chechik. Fault-tolerant compact routing schemes for general graphs. *Inf. Comput.*, 222:36–44, 2013.
5. Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. f-sensitivity distance oracles and routing schemes. *Algorithmica*, 63(4):861–882, 2012.
6. Erik D. Demaine, Gad M. Landau, and Oren Weimann. On cartesian trees and range minimum queries. *Algorithmica*, 68(3):610–625, 2014.
7. Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37(5):1299–1318, 2008.
8. Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 169–178, 2011.
9. Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *SODA'09: Proceedings of 19th Annual ACM – SIAM Symposium on Discrete Algorithms*, pages 506–515, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.
10. Wojciech Fraczak, Loukas Georgiadis, Andrew Miller, and Robert Endre Tarjan. Finding dominators via disjoint set union. *J. Discrete Algorithms*, 23:2–20, 2013.
11. Loukas Georgiadis and Robert Endre Tarjan. Dominators, directed bipolar orders, and independent spanning trees. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 375–386, 2012.
12. Thomas Lengauer and Robert Endre Tarjan. A fast algorithm for finding dominators in a flowgraph. *ACM Trans. Program. Lang. Syst.*, 1(1):121–141, 1979.
13. Merav Parter. Dual failure resilient BFS structure. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21-23, 2015*, pages 481–490, 2015.
14. Merav Parter and David Peleg. Sparse fault-tolerant BFS trees. In *Algorithms – ESA 2013 – 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, pages 779–790, 2013.
15. Merav Parter and David Peleg. Fault tolerant approximate BFS structures. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1073–1092, 2014.
16. Maxim Teslenko and Elena Dubrova. An efficient algorithm for finding double-vertex dominators in circuit graphs. In *2005 Design, Automation and Test in Europe Conference and Exposition (DATE 2005), 7-11 March 2005, Munich, Germany*, pages 406–411, 2005.
17. Virginia Vassilevska Williams. Faster replacement paths. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1337–1346, 2011.

Graph Minors for Preserving Terminal Distances Approximately – Lower and Upper Bounds*†

Yun Kuen Cheung[1], Gramoz Goranci[2], and Monika Henzinger[3]

1 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
yun.kuen.cheung@univie.ac.at
2 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
gramoz.goranci@univie.ac.at
3 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
monika.henzinger@univie.ac.at

Abstract

Given a graph where vertices are partitioned into k terminals and non-terminals, the goal is to compress the graph (i.e., reduce the number of non-terminals) using minor operations while preserving terminal distances approximately. The distortion of a compressed graph is the maximum multiplicative blow-up of distances between all pairs of terminals. We study the trade-off between the number of non-terminals and the distortion. This problem generalizes the Steiner Point Removal (SPR) problem, in which all non-terminals must be removed.

We introduce a novel black-box reduction to convert any lower bound on distortion for the SPR problem into a super-linear lower bound on the number of non-terminals, with the same distortion, for our problem. This allows us to show that there exist graphs such that every minor with distortion less than 2 / 2.5 / 3 must have $\Omega(k^2)$ / $\Omega(k^{5/4})$ / $\Omega(k^{6/5})$ non-terminals, plus more trade-offs in between. The black-box reduction has an interesting consequence: if the tight lower bound on distortion for the SPR problem is super-constant, then allowing any $\mathcal{O}(k)$ non-terminals will *not* help improving the lower bound to a constant.

We also build on the existing results on spanners, distance oracles and connected 0-extensions to show a number of upper bounds for general graphs, planar graphs, graphs that exclude a fixed minor and bounded treewidth graphs. Among others, we show that any graph admits a minor with $\mathcal{O}(\log k)$ distortion and $\mathcal{O}(k^2)$ non-terminals, and any planar graph admits a minor with $1+\varepsilon$ distortion and $\widetilde{\mathcal{O}}((k/\varepsilon)^2)$ non-terminals.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases Distance Approximating Minor, Graph Minor, Graph Compression, Vertex Sparsification, Metric Embedding

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.131

1 Introduction

Graph compression generally describes a transformation of a *large* graph G into a *smaller* graph G' that preserves, either exactly or approximately, certain features (e.g., distance, cut, flow) of G. Its algorithmic value is apparent, since the compressed graph can be computed in

* A full version of the paper is available at http://arxiv.org/abs/1604.08342.

† The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement no. 340506. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) under grant agreement no. 317532.

a preprocessing step of an algorithm, so as to reduce subsequent running time and memory. Some notable examples are graph spanners, distance oracles and cut/flow sparsifiers.

In this paper, we study compression using minor operations, which has attracted increasing attention in recent years. Minor operations include vertex/edge deletions and edge contractions. It is naturally motivated since it preserves certain structural properties of the original graph, e.g., any minor of a planar graph remains planar, while reducing the size of the graph. We are interested in *vertex sparsification*, where G has a designated subset T of k vertices called the *terminals*, and the goal is to reduce the number of non-terminals in G' while preserving some feature among the terminals. Recent work in this field studied preserving cuts and flows. Our focus here is on preserving terminal distances approximately in a multiplicative sense, i.e., we want that for any terminals t, t', $d_G(t,t') \leq d_{G'}(t,t') \leq \alpha \cdot d_G(t,t')$, for a small *distortion* α. This problem, called *Approximate Terminal Distance Preservation (ATDP) problem*, has natural applications in multicast routing [6] and network traffic optimization [22]. It was also suggested in [15] that to solve the *subset travelling salesman problem*, one can compute a compressed minor with a small distortion as a preprocessing step for algorithms that solve the travelling salesman problem for planar graphs.

ATDP was initiated by Gupta [11], who introduced the related *Steiner Point Removal (SPR) problem*: Given a tree G with both terminals and non-terminals, output a weighted tree G' *with terminals only* which minimizes the distortion. Gupta gave an algorithm that achieves a distortion of 8. Chan et al. [4] observed that Gupta's algorithm returned always a minor of G. For general graphs, Kamma et al. [13] gave an algorithm to construct a minor with distortion $\mathcal{O}(\log^5 k)$. Krauthgamer et al. [15] studied ATDP and showed that every graph has a minor with $\mathcal{O}(k^4)$ non-terminals and distortion 1. It is then natural to ask, for different classes of graphs, what the trade-off between the distortion and the number of non-terminals is. In this paper, for different classes of graphs, and w.r.t. different allowed distortions, we provide lower and upper bounds on the number of non-terminals needed.

Further Related Work. Basu and Gupta [3] showed that for outer-planar graphs, SPR can be solved with distortion $\mathcal{O}(1)$. When randomization is allowed, Englert et al. [9] showed that for graphs that exclude a fixed minor, one can construct a randomized minor for SPR with $\mathcal{O}(1)$ expected distortion. Krauthgamer et al. [15] proved that solving ATDP with distortion 1 for planar graphs needs $\Omega(k^2)$ non-terminals.

Recently, there has been a growing interest on cut/flow vertex sparsifiers [19, 16, 5, 18, 9, 7, 2, 21]; given a capacitated graph G with terminals $T \subset V$, the goal is to find a sparsifier H with $V(H) = T$ preserving all terminal cuts up to a factor $q \geq 1$, i.e. for all $S \subset T$, $\text{mincut}_G(S, T \setminus S) \leq \text{mincut}_H(S, T \setminus S) \leq q \cdot \text{mincut}_G(S, T \setminus S)$. In this setting, there is an equivalence between the construction of vertex cut/flow and distance sparsifiers [20, 9].

A related graph compression is spanners, where the objective is to reduce the number of edges by edge deletions only. We will use a spanner algorithm (e.g., [1]) to derive our upper bound results for general graphs. Although spanner operation enjoys much less freedom than minor operation, proving a lower bound result for it is notably difficult. Assuming the Erdös girth conjecture [10], there are lower bounds that match the best known upper bounds, but the conjecture seems far from being settled [25]. Woodruff [27] showed a lower bound result bypassing the conjecture, but only for *additive* spanners.

Graph	Upper Bound	Lower Bound
General	$\forall q \in \mathbb{N} \quad (2q-1, \mathcal{O}(k^{2+2/q}))$	$(2-\varepsilon, \Omega(k^2))$
General	–	$(2.5-\varepsilon, \Omega(k^{5/4}))$, $(3-\varepsilon, \Omega(k^{6/5}))$
Bounded-treewidth p	$\forall q \in \mathbb{N} \quad (2q-1, \mathcal{O}(p^{1+2/q}k))$	$(1, \Omega(pk))$ †
Excluded-Fixed-Minor	$(\mathcal{O}(1), \widetilde{\mathcal{O}}(k^2)$	–
Planar	$(3, \widetilde{\mathcal{O}}(k^2))$, $(1+\varepsilon, \widetilde{\mathcal{O}}((k/\varepsilon)^2)$	$(1+o(1), \Omega(k^2))$ †

Our Contributions. For various classes of graphs, we show lower and upper bounds on the number of non-terminals needed in the minor for low distortion. The table above summarizes our results (results with † are from [15]); see the full version for more details.

For our lower bound results, we use a novel black-box reduction to convert any lower bound on distortion for the SPR problem into a super-linear lower bound on the number of non-terminals for ATDP with the same distortion. Precisely, we show that given any graph G^* such that solving its SPR problem leads to a minimum distortion of α, we use G^* to construct a new graph G such that every minor of G with distortion less than α must have at least $\Omega(k^{1+\delta(G^*)})$ non-terminals, for some constant $\delta(G^*) > 0$. The lower bound results in the above table are obtained by using for G^* a complete ternary tree of height 2, which was shown that solving its SPR problem leads to minimum distortion 3 [11]. More trade-offs are shown by using for G^* a complete ternary tree of larger heights.

The black-box reduction has an interesting consequence. For the SPR problem on general graphs, there is a huge gap between the best known lower and upper bounds, which are 8 [4] and $\mathcal{O}(\log^5 k)$ [13]; it is unclear what the asymptotically tight bound would be. Our black-box reduction allows us to prove the following result concerning the tight bound: for general graphs, if the tight bound on distortion for the SPR problem is super-constant, then for any constant $C > 0$, even if Ck non-terminals are allowed in the minor, the lower bound will remain super-constant. See Theorem 15 for a formal statement of this result.

We also build on the existing results on spanners, distance oracles and connected 0-extensions to show a number of upper bound results for general graphs, planar graphs and graphs that exclude a fixed minor. Our techniques, combined with an algorithm in Krauthgamer et al. [15], yield an upper bound result for graphs with bounded treewidth. In particular, our upper bound on planar graphs implies that allowing quadratic number of non-terminals, we can construct a deterministic minor with arbitrarily small distortion.

2 Preliminaries

Let $G = (V, E, \ell)$ denote an undirected graph with terminal set $T \subset V$ of cardinality k, where $\ell : E \to \mathbb{R}^+$ is the length function over edges E. A graph H is a *minor* of G if H can be obtained from G by performing a sequence of vertex/edge deletions and edge contractions, but no terminal can be deleted, and no two terminals can be contracted together. In other words, all terminals in G must be *preserved* in H.

Besides the above standard description of minor operations, there is another equivalent way to construct a minor H from G [13], which will be more convenient for presenting some of our results. A partial partition of $V(G)$ is a collection of pairwise disjoint subsets of $V(G)$ (but their union can be a proper subset of $V(G)$). Let $S_1, \cdots, S_m$ be a partial partition of $V(G)$ such that (1) each induced graph $G[S_i]$ is connected, (2) each terminal belongs to exactly one of these partial partitions, and (3) no two terminals belong to the same partial partition. Contract the vertices in each S_i into one single "super-node" in H. For any vertex $u \in V(G)$, let $S(u)$ denote the partial partition that contains u; for any super-node $u \in V(H)$, let $S(u)$ denote the partial partition that is contracted into u. In H, super-nodes u_1, u_2 are adjacent *only if* there exists an edge in G with one of its endpoints in $S(u_1)$ and the other in $S(u_2)$. We denote the super-node that contains terminal t by t as well.

▶ **Definition 1.** The graph $H = (V', E', \ell')$ is an α-distance approximating minor (abbr. α-DAM) of $G = (V, E, \ell)$ if H is a minor of G and for any $t, t' \in T$, $d_G(t, t') \le d_H(t, t') \le \alpha \cdot d_G(t, t')$. H is an (α, y)-DAM of G if H is an α-DAM of G with at most y non-terminals.

We note that the SPR problem is equivalent to finding an $(\alpha, 0)$-DAM. One can also define a randomized version of distance approximating minor:

▶ **Definition 2.** Let π be a probability distribution over minors of $G = (V, E, \ell)$. We call π an α-randomized distance approximating minor (abbr. α-rDAM) of G if for any $t, t' \in T$, $\mathbb{E}_{H\sim\pi}[d_H(t,t')] \leq \alpha \cdot d_G(t,t')$, and for every minor H in the support of π, $d_H(t,t') \geq d_G(t,t')$. Furthermore, we call π an (α, y)-rDAM if π is an α-rDAM of G, and every minor in the support of π has at most y non-terminals.

3 Deterministic and Randomized Lower Bounds

For all the lower bound results, we use a tool in combinatorial design called *Steiner system* (or alternatively, *balanced incomplete block design*). Let $[k]$ denote the set $\{1, 2, \cdots, k\}$.

▶ **Definition 3.** Given a ground set $T = [k]$, an $(s, 2)$-Steiner system (abbr. $(s, 2)$-SS) of T is a collection of s-subsets of T, denoted by $\mathcal{T} = \{T_1, \cdots, T_r\}$, where $r = \binom{k}{2} / \binom{s}{2}$, such that every 2-subset of T is contained in *exactly* one of the s-subsets.

▶ **Lemma 4** ([26])**.** *For any integer $s \geq 2$, there exists an integer M_s such that for every $q \in \mathbb{N}$, the set $[M_s + qs(s-1)]$ admits an $(s, 2)$-SS.*

Our general strategy is to use the following black-box reduction, which proceeds by taking a *small* connected graph G^* as input, and it outputs a *large* graph G which contains many disjoint embeddings of G^*. Here is how it exactly proceeds:

- Let G^* be a graph with $s \geq 2$ terminals and $q \geq 1$ non-terminals. Let k be an integer, as given in Lemma 4, such that the terminal set $T = [k]$ admits an $(s, 2)$-SS $\mathcal{T}$.
- We construct $\mathcal{T}' \subseteq \mathcal{T}$ that satisfies *certain* property depending on the specific problem. For each s-set in $\mathcal{T}'$, we add q non-terminals to the s-set, which altogether form a *group*. The union of vertices in all groups is the vertex set of our graph G. We note that each terminal may appear in many groups, but each non-terminal appears in one group only.
- *Within* each of the groups, we embed G^* in the natural way.

The following two lemmas describe some basic properties of all minors of G output by the black-box above. Their proofs are deferred to the full version.

▶ **Lemma 5.** *Let H be a minor of G. Then for each edge (u_1, u_2) in H, there exists exactly one group R in G such that $S(u_1) \cap R$ and $S(u_2) \cap R$ are both non-empty.*

The above lemma permits us to legitimately define the notion R-edge: an edge (u_1, u_2) in H is an R-edge if R is the unique group that intersects both $S(u_1)$ and $S(u_2)$.

▶ **Lemma 6.** *Suppose that in a minor H of G, (u_1, u_2) is a R_1-edge and (u_2, u_3) is R_2-edge, where $R_1 \neq R_2$. Then R_1 and R_2 intersect, and $S(u_2)$ contains the terminal in $R_1 \cap R_2$.*

We will show that for any minor H with low distortion, at least one of the non-terminals in each group must be retained, and thus H must have at least $|\mathcal{T}'|$ non-terminals. We first present some of our main theorems on lower bounds and then prove them; two more theorems are given in Section 3.3.

▶ **Theorem 7.** *For infinitely many $k \in \mathbb{N}$, there exists a bipartite graph with k terminals which does not have a $(2 - \epsilon, k^2/7)$-DAM, for all $\epsilon > 0$.*

▶ **Theorem 8.** *There exists a constant $c_1 > 0$, such that for infinitely many $k \in \mathbb{N}$, there exists a quasi-bipartite graph with k terminals which does not have an $(\alpha - \epsilon, c_1 k^\gamma)$-DAM, for all $\epsilon > 0$, where α, γ are given in the table below.*

α	2.5	3	10/3	11/3	4	4.2	4.4
γ	5/4	6/5	10/9	11/10	12/11	21/20	22/21

▶ **Theorem 9.** *For infinitely many $k \in \mathbb{N}$, there exists a bipartite graph with k terminals which does not have a $\left(2 - \epsilon, \epsilon^3 k^2/150\right)$-rDAM, for any $1 \geq \epsilon > 0$.*

3.1 Proof of Theorem 7

We start by reviewing the lower bound for SPR problem on stars due to Gupta [11].

▶ **Lemma 10.** *Let $G^* = (T \cup \{v\}, E)$ be an unweighted star with $k \geq 3$ terminals, in which v is the center of the star. Then, every edge-weighted graph only on the terminals T with fewer than $\binom{k}{2}$ edges has distortion at least 2.*

We construct G using the black-box reduction above. Let $k \in \mathbb{N}$ be such that the terminals $T = [k]$ admits a $(3, 2)$-SS, denoted by $\mathcal{T}$. Here, we set $\mathcal{T}' = \mathcal{T}$ and G^* to be the star with 3 terminals, as described in Lemma 10.

By the definition of Steiner system, the shortest path between every pair of terminal t, t' in G is unique, which is the 2-hop path within the group that contains both terminals, i.e., $d_G(t, t') = 2$ for all $t, t' \in T$. Every other simple path between t, t' must pass through an extra terminal, so the length of such simple path is at least 4.

Let H be a minor of G. Suppose that the number of non-terminals in H is less than r, then there exists a group R in which its non-terminal is not retained (which means that it is either deleted, or contracted into a terminal in that group). By Lemma 10, there exists a pair of terminals in that group such that every simple path within R (which means a path comprising of R-edges only) between the two terminals has length at least 4. And every other simple path must pass through an extra terminal (just as in G), so again it has length at least 4. Thus, the distortion of the two terminals is at least 2.

Therefore, every $(2 - \epsilon)$-DAM of G must have $r > k^2/7$ non-terminals.

3.2 Proof of Theorem 8

We will give the proof for the case $\alpha = 2.5$ here, and discuss how to generalize this proof for other distortions. We will first define the notions of *detouring graph* and *detouring cycle*, and then use them to construct the graph G that allows us to show the lower bound.

Detouring Graph and Detouring Cycle. For any $s \geq 3$, let $k \in \mathbb{N}$ be such that the terminal set $T = [k]$ admits an $(s, 2)$-SS. Let $\mathcal{T} = \{T_1, \cdots, T_r\}$ be such an $(s, 2)$-SS. A *detouring graph* has the vertex set $\mathcal{T}$. By the definition of Steiner system, $|T_i \cap T_j|$ is either zero or one. In the detouring graph, T_i is adjacent to T_j if and only if $|T_i \cap T_j| = 1$. Thus, in the detouring graph, it is legitimate to give each edge (T_i, T_j) a *terminal label*, which is the terminal in $T_i \cap T_j$. A *detouring cycle* is a cycle in the detouring graph such that no two neighbouring edges of the cycle have the same terminal label.

Suppose that two edges in the detouring graph have a common vertex, and their terminal labels are different, denoted by t, t'. Then the common vertex must be an s-set in $\mathcal{T}$ containing both t, t'. By the definition of Steiner system, the s-set is uniquely determined.

▶ **Claim 11.** *In the detouring graph, number of detouring cycles of size $\ell \geq 3$ is at most k^ℓ.*

Our key lemma is: for any $L \geq 3$, we can retain $\Omega_s(k^{L/(L-1)})$ vertices in the detouring graph, such that the induced graph on these vertices has *no* detouring cycle of size L or less.

▶ **Lemma 12.** *For any integer $L \geq 3$, given a detouring graph with vertex set $\mathcal{T} = \{T_1, T_2, \cdots, T_r\}$, there exists a subset $\mathcal{T}' \subset \mathcal{T}$ of cardinality $\Omega_s(k^{L/(L-1)})$ such that the induced graph on $\mathcal{T}'$ has no detouring cycle of size L or less.*

Proof. We choose the subset $\mathcal{T}'$ by the following randomized algorithm:

1. Each vertex is picked into $\mathcal{T}'$ with probability $\delta k^{-(L-2)/(L-1)}$, where $\delta = \delta(s) < 1$ is a positive constant which we will derive explicitly later.
2. While (there is a detouring cycle of size L or less in the induced graph of $\mathcal{T}'$)
   Remove a vertex in the detouring cycle from $\mathcal{T}'$

After Step 1, $\mathbb{E}[|\mathcal{T}'|] = r \cdot \delta k^{-(L-2)/(L-1)} \geq \frac{\delta}{2s(s-1)} k^{L/(L-1)}$. Using Claim 11, the expected number of detouring cycles of size L or less is at most $\sum_{\ell=3}^{L} k^{\ell} \cdot (\delta k^{-(L-2)/(L-1)})^{\ell} \leq 2\delta^3 k^{L/(L-1)}$. Thus, the expected number of vertices removed in Step 2 is at most $2\delta^3 k^{L/(L-1)}$. Now, choose $\delta = 1/\sqrt{8s(s-1)}$. By the end of the algorithm,

$$\mathbb{E}[|\mathcal{T}'|] \geq \frac{\delta}{2s(s-1)} k^{L/(L-1)} - 2\delta^3 k^{L/(L-1)} = \Omega(k^{L/(L-1)}).$$ ◀

Construction of G and the Proof. Recall the black-box reduction. Let k be an integer such that $T = [k]$ admits a $(9, 2)$-SS $\mathcal{T}$. By Lemma 12, we choose $\mathcal{T}'$ to be a subset of $\mathcal{T}$ with $|\mathcal{T}'| = \Omega(k^{5/4})$, such that the induced graph on $\mathcal{T}'$ has no detouring cycle of size 5 or less. We choose G^* to be a complete ternary tree of height 2, in which the 9 leaves are the terminals. For each $T_i \in \mathcal{T}'$, we add four non-terminals to T_i, altogether forming a *group*.

The following lemma is a direct consequence that the induced graph on $\mathcal{T}'$ has no detouring cycle of size 5 or less.

▶ **Lemma 13.** *For any two terminals t, t' in the same group, let R denote the group. Then, in any minor H of G, every simple path from t to t' either comprises of R-edges only, or it comprises of edges from at least 5 groups other than R.*

Proof. Proof of Theorem 8 Let H be a $(2.5 - \epsilon)$-DAM of G, for some $\epsilon > 0$. Suppose that there exists a group such that all its non-terminals are not retained in H. By [11], there exists a pair of terminals t, t' in that group such that every simple path between t and t', which comprises of edges of that group only, has length at least $3 \cdot d_G(t, t')$.

By Lemma 13 and Lemma 6, any other simple path P between t and u passes through at least 4 other terminals, say they are t_a, t_b, t_c, t_d in the order of the direction from t to t'. We denote this path by $P := t \to t_a \to t_b \to t_c \to t_d \to t'$, by ignoring the non-terminals along the path. Between every pair of consecutive terminals in P, the length is at least 2. Thus, the length of P is at least 10. Since $d_G(t, t') \leq 4$, the length of P is at least $2.5 \cdot d_G(t, t')$.

Thus, the length of *every* simple path from t to t' in H is at least $2.5 \cdot d_G(t, t')$, a contradiction. Therefore, at least one non-terminal in each group is retained in H. As there are $\Omega(k^{5/4})$ groups, we are done. ◀

For the other results in Theorem 8, we follow the above proof almost exactly, with the following modifications. Set $s = 3^h$ for some $h \geq 2$, and set G^* to be a complete ternary tree with height h, in which the leaves are the terminals. Let α_h be a lower bound on the distortion for the SPR problem on G^*. Apply Lemma 12 with some integer $h < L \leq \lceil \alpha_h h \rceil$. Following the above proof, attaining a distortion of $\min\left\{\frac{L}{h}, \alpha_h\right\} - \epsilon$ needs $\Omega(k^{L/(L-1)})$ non-terminals.

The last puzzle we need is the values of α_h. Chan et al. [4] proved that for complete binary trees of height h, $\lim_{h\to+\infty} \alpha_h = 8$, but they did not give explicit values of α_h. We apply their ideas to complete ternary tree of height h, to obtain explicit values for $h \leq 5$, which are used to prove all the results in Theorem 8. The explicit values are $\alpha_2 = 3$, $\alpha_3 = \alpha_4 = 4$ and $\alpha_5 = 4.4$. We discuss the details for computing these values in the full version.

3.3 Full Generalization of Theorem 8, and its Interesting Consequence

Indeed, we can set G^* as *any* graph. In our above proofs we used a tree for G^* because the only known lower bounds on distortion for the SPR problem are for trees. If one can find a graph G^* (either by a mathematical proof, or by computer searches) such that its distortion for the SPR problem is at least α, applying the black-box reduction with this G^*, and reusing the above proof show that there exists a graph G with k terminals such that attaining a distortion of $\alpha - \epsilon$ needs $\Omega(k^{1+\delta(G^*)})$ non-terminals, for some $\delta(G^*) > 0$.

▶ **Theorem 14.** *Let G^* be a graph with s terminals, and the distance between any two terminals is between 1 and β. Suppose the distortion for the* SPR *problem on G^* is at least α. Then, for any positive integer $\max\{2, \lceil\beta\rceil\} \leq L \leq \lceil\alpha\beta\rceil$, there exists a constant $c_4 := c_4(s) > 0$, such that for infinitely many $k \in \mathbb{N}$, there exists a graph with k terminals which does not have a $\left(\min\{L/\beta, \alpha\} - \epsilon, c_4 k^{L/(L-1)}\right)$-DAM, for all $\epsilon > 0$.*

The above theorem has an interesting consequence. For the SPR problem on general graphs, the best known lower bound is 8, while the best known upper bound is $\mathcal{O}(\log^5 k)$ [13]. There is a huge gap between the two bounds, and it is not clear where the tight bound locates in between. Suppose that the tight lower bound on SPR is super-constant. Then for any positive constant α, there exists a graph G^*_α with $s(\alpha)$ terminals and some non-terminals, such that the distortion is larger than α. By Theorem 14, G^*_α can be used to construct a family of graphs with k terminals, such that to attain distortion α, the number of non-terminals needed is super-linear in k. Recall that in SPR, no non-terminal can be retained. In other words, Theorem 14 implies that: *if retaining no non-terminal will lead to a super-constant lower bound on distortion, then having the power of retaining any linear number of non-terminals will not improve the lower bound to a constant.*

Formally, we define the following generalization of SPR problem. Let LSPR_y denote the problem that for an input graph with k terminals, find a DAM with at most yk non-terminals so as to minimize the distortion; the SPR problem is equivalent to LSPR_0.

▶ **Theorem 15.** *For general graphs,* SPR *has super-constant lower bound on distortion if and only if for any constant $y \geq 0$, LSPR_y has super-constant lower bound on distortion.*

3.4 Proof of Theorem 9

In this subsection we give a lower bound for rDAM. The strategy we follow will be very similar to that of Theorem 7. In fact, one can view it as a randomized version of that proof. We start with the following lemma, which generalizes the deterministic SPR lower bound of Gupta in Lemma 10 to randomized minors.

▶ **Lemma 16.** *Let $G^* = (T \cup \{v\}, E)$ be an unweighted star with $k \geq 3$ terminals, in which v is the center of the star. Then, for every probability distribution over minors of G^* with vertex set T, there exists a terminal pair with distortion at least $2(1 - 1/k)$.*

We now continue with the construction of our input graph. For some constant $s \geq 3$ and some integer k, we construct a $(s, 2)$-SS of the terminal set T and denote it by

$\mathcal{T} = \{T_1, \ldots, T_r\}$, where $r = \binom{k}{2}/\binom{s}{2} \geq 2\binom{k}{2}/s^2$. Similarly to the proof of Theorem 7, we apply the black-box reduction with $\mathcal{T}' = \mathcal{T}$, and set G^* as a star with c_1 terminals, to generate a bipartite graph G. For any constant $c_1 > 0$, we define the family of minors $\mathcal{L} := \{H : H \text{ is a minor of } G \text{ and } |V(H)| < \binom{k}{2}/c_1\}$.

▶ **Claim 17.** *Let π be any probability distribution over $\mathcal{L}$. There exists a non-terminal of G that is involved in an edge contraction with probability at least $1 - s^2/2c_1$ under π.*

Proof. Proof of Theorem 9 Let v be the non-terminal from Claim 17 and let T_i be its corresponding set of size c_1. Invoking Lemma 16 and using conditional expectations, we get that there exists a terminal pair $(t, t') \in T_i$ such that

$$\frac{\mathbb{E}_\pi[d_H(t,t')]}{d_G(t,t')} \geq \frac{\mathbb{E}_\pi\left[d_H(t,t') \mid v \text{ is contracted}\right] \cdot \mathbb{P}_\pi\left[v \text{ is contracted}\right]}{d_G(t,t')}$$
$$\geq 2\left(1 - \frac{1}{s}\right)\left(1 - \frac{s^2}{2c_1}\right) \geq 2 - \left(\frac{2}{s} + \frac{s^2}{c_1}\right),$$

which can be made arbitrarily close to 2 by setting s and c_1 sufficiently large. To be precise, given any $\epsilon > 0$, by setting $s = 5/\epsilon$ and $c_1 = 2s^2/\epsilon$, the distortion is at least $2 - \epsilon$. ◀

4 Minor Construction for General Graphs

In this section we give minor constructions that present numerous trade-offs between the distortion and size of DAMs. Our results are obtained by combining the work of Coppersmith and Elkin [8] on sourcewise distance preservers with the well-known notion of spanners.

Given an undirected graph $G = (V, E, \ell)$ with terminals T, we let $\Pi_{u,v}$ denote the shortest path between u and v in G. Without loss of generality, we assume that for any pair of vertices (u, v), the shortest path connecting u and v is *unique*. This can be achieved by slightly perturbing the original edge lengths of G (see [8]).

For a graph G, let $N_G(u)$ denote the vertices incident to u in G. We say that two paths Π and Π' branch at a vertex $u \in V(\Pi) \cap V(\Pi')$ iff $|N_{\Pi \cup \Pi'}(u)| > 2$. We call such a vertex u a *branching* vertex. Let $\mathcal{P}$ denote the set of shortest paths corresponding to every pair of vertices in G. We review the following result proved in [8, Lemma 7.5].

▶ **Lemma 18.** *Any pair of shortest paths $\Pi, \Pi' \in \mathcal{P}$ has at most two branching vertices.*

▶ **Definition 19** (Terminal Path Cover)**.** Given $G = (V, E, \ell)$ with terminals T, a set of shortest paths $\mathcal{P}' \subset \mathcal{P}$ is an $(\alpha, f(k))$-terminal path cover (abbr. $(\alpha, f(k))$-TPc) of G if

1. $\mathcal{P}'$ covers the terminals, i.e. $T \subseteq V(H)$, where $H = \bigcup_{\Pi \in \mathcal{P}'} E(\Pi)$,
2. $|\mathcal{P}'| \leq f(k)$ and $\forall t, t' \in T$, $d_G(t,t') \leq d_H(t,t') \leq \alpha \cdot d_G(t,t')$.

We remark that the endpoints of the shortest paths in $\mathcal{P}'$ are not necessarily terminals. Now we give a simple algorithm generalizing the one presented by Krauthgamer et al. [15].

Algorithm 1 MinorSparsifier (graph G, terminals T, $(\alpha, f(k))$-TPc $\mathcal{P}'$ of G)

1: Set $H = \emptyset$. Then add all shortest paths from the path cover $\mathcal{P}'$ to H.
2: **while** there exists a degree two non-terminal v incident to edges (v, u) and (v, w) **do**
3: Contract the edge (u, v), then set the length of edge (u, w) to $d_H(u, w)$.
4: **return** H

▶ **Lemma 20.** *For a given graph $G = (V, E, \ell)$ with terminals $T \subset V$ and an $(\alpha, f(k))$-TPc $\mathcal{P}'$ of G,* MinorSparsifier$(G,T,\mathcal{P}')$ *outputs an $(\alpha, f(k)^2)$-DAM of G.*

A trivial *exact* terminal path cover for any k-terminal graph is to take the union of all terminal shortest paths, which we refer to as the $(1, \mathcal{O}(k^2))$-TPc $\mathcal{P}'$ of G. Krauthgamer et al. [15] used this $(1, \mathcal{O}(k^2))$-TPc to construct an $(1, \mathcal{O}(k^4))$-DAM. Here, we study the question of whether increasing the distortion slightly allows us to obtain a cover of size $o(k^2)$. We answer this question positively, by reducing it to the well-known spanner problem. Let $q \geq 1$ be an integer and let $G = (V, E, \ell)$ be an undirected graph. A q-spanner of G is a subgraph $S = (V, E_S, \ell)$ such that $\forall u, v \in V,\ d_G(u, v) \leq d_S(u, v) \leq q \cdot d_G(u, v)$. We refer to q and $|E_S|$ as the *stretch* and *size* of spanner S, respectively.

▶ **Lemma 21** ([1]). *Let $q \geq 1$ be an integer. Any graph $G = (V, E, \ell)$ admits a $(2q-1)$-spanner S of size $\mathcal{O}(|V|^{1+1/q})$.*

Given a graph $G = (V, E, \ell)$ with terminals T, we compute the complete graph $Q_T = (T, \binom{T}{2}, d_G|T)$, where $d_G|T$ denotes the distance metric of G restricted to the point set T (In other words, for any pair of terminals $t, t' \in T$, the weight of the edge connecting them in Q_T is given by $w_{Q_T}(t, t') = d_G(t, t')$). Recall that all shortest paths in G are unique.

Using Lemma 21, we construct a $(2q-1)$-spanner S of size $\mathcal{O}(k^{1+1/q})$ for Q_T. Observe that each edge of S corresponds to an unique (terminal) shortest path in G since S is a subgraph of Q_T. Thus, the set of shortest paths corresponding to edges of S form a $(2q-1, \mathcal{O}(k^{1+1/q}))$-TPc $\mathcal{P}'$ of G. Using $\mathcal{P}'$ with Lemma 20 gives the following theorem.

▶ **Theorem 22.** *Let $q \geq 1$ an integer. Any graph $G = (V, E, \ell)$ with $T \subset V$ admits a $(2q-1, \mathcal{O}(k^{2+2/q}))$-DAM.*

We mention two trade-offs from the above theorem. When $q = 2$, we get an $(3, \mathcal{O}(k^3))$-DAM. When $q = \log k$, we get an $(\mathcal{O}(\log k), \mathcal{O}(k^2))$-DAM. These are new distortion-size trade-offs.

The above method allows us to have improved guarantees for bounded treewidth graphs.

▶ **Theorem 23.** *Let $q \geq 1$ be an integer. Any graph $G = (V, E, \ell)$ with treewidth at most p, $T \subset V$ and $k \geq p$ admits a $(2q-1, \mathcal{O}(p^{1+2/q}k))$-DAM.*

5 Minor Construction for Graphs that Exclude a Fixed Minor

In this section we give improved guarantees for distance approximating minors for special families of graphs. Specifically, we show that graphs that exclude a fixed minor admit an $(\mathcal{O}(1), \widetilde{\mathcal{O}}(k^2))$-DAM. This family of graphs includes, among others, the planar graphs.

The reduction to spanner in Section 4 does not consider the structure of Q_T, which is inherited from the input graph. We exploit this structure, together with the use of the randomized Steiner Point Removal Problem, which is equivalent to finding an $(\alpha, 0)$-rDAM.

We will make use the following theorem due to Englert et al. [9, Theorem 14].

▶ **Theorem 24** ([9]). *Let $\alpha = \mathcal{O}(1)$. Given a graph that excludes a fixed minor $G = (V, E, \ell)$ with $T \subset V$, there is a probability distribution π over its minors $H = (T, E', \ell')$, such that $\forall\ t, t' \in T,\ \mathbb{E}_{H\sim\pi}[d_H(t, t')] \leq \alpha \cdot d_G(t, t')$ and for all H in support of π, $d_H(t, t') \geq d_G(t, t')$.*

Given a graph G that excludes a fixed minor, any minor H of G only on the terminals also excludes the fixed minor. Thus H has $\mathcal{O}(k)$ edges [23]. This leads to the corollary below.

▶ **Corollary 25.** *Let $\alpha = \mathcal{O}(1)$. Given a graph that excludes a fixed minor $G = (V, E, \ell)$ with $T \subset V$ and Q_T as defined in Section 4, there exists a probability distribution π over subgraphs $H = (T, E', \ell')$ of Q_T, each having at most $\mathcal{O}(k)$ edges, such that for all $t, t' \in T,\ \mathbb{E}_{H\sim\pi}[d_H(t, t')] \leq \alpha \cdot d_{Q_T}(t, t')$.*

Proof. Let π be the distribution over minors of G from Theorem 24, then every minor in its support is clearly a subgraph of Q_T with $\mathcal{O}(k)$ edges. Since during the construction of these minors we may assume that $\forall (t,t') \in E',\ \ell'(t,t') = d_G(t,t')$, the corollary follows. ◀

▶ **Lemma 26.** *Given a graph that excludes a fixed minor $G = (V, E, \ell_G)$ with $T \subset V$, and Q_T as defined in Section 4, there exists an $\mathcal{O}(1)$-spanner S of size $\mathcal{O}(k \log k)$ for Q_T.*

Proof. Let π be the probability distribution over subgraphs H from Corollary 25. Set $S = \emptyset$. First, we sample independently $q = 3 \log k$ subgraphs $H_1, \ldots, H_q$ from π. We then add the edges from all these subgraphs to the graph S, i.e., $E_S = \bigcup_{i=1}^{q} E_{H_i}$. Fix an edge (t,t') from Q_T and a subgraph H_i. By Corollary 25 and the Markov inequality, $\mathbb{P}[d_{H_i}(t,t') \geq 2\alpha \cdot d_{Q_T}(t,t')] \leq 2^{-1}$, and hence

$$\mathbb{P}[d_S(t,t') \geq 2\alpha \cdot d_{Q_T}(t,t')] = \prod_{i=1}^{q} \mathbb{P}[d_{H_i}(t,t') \geq 2\alpha \cdot d_{Q_T}(t,t')] \leq 2^{-q} = k^{-3}.$$

Applying union bound overall all edges from Q_T yields

$$\mathbb{P}[\text{there exists an edge } (t,t') \in Q_T \text{ s.t. } d_S(t,t') \geq 2\alpha \cdot d_{Q_T}(t,t')] \leq k^2 \cdot k^{-3} = k^{-1}.$$

Hence, for all edges (t,t') from Q_T, with probability at least $1 - 1/k$, we preserve the shortest path distance between t and t' up to a factor of $2\alpha = \mathcal{O}(1)$ in S. Since S is a subgraph of Q_T, this implies that there exists a $\mathcal{O}(1)$-spanner S of size $\mathcal{O}(k \log k)$ for Q_T. ◀

Similar to the last section, the set of shortest paths corresponding to edges of S is an $(\mathcal{O}(1), \mathcal{O}(k \log k))$-TPc $\mathcal{P}'$ of G. Using $\mathcal{P}'$ with Lemma 20 gives the following theorem.

▶ **Theorem 27.** *Any graph that excludes a fixed minor $G = (V, E, \ell)$ with $T \subset V$ admits an $(\mathcal{O}(1), \widetilde{\mathcal{O}}(k^2))$-DAM.*

6 Minor Construction for Planar Graphs

In this section, we show that for planar graphs, we can improve the constant guarantee bound on the distortion to 3 and $1 + \varepsilon$, respectively, without affecting the size of the minor. Our work builds on existing techniques used in the context of approximate distance oracles, thereby bypassing our previous spanner reduction. Both results use essentially the same ideas and rely heavily on the fact that planar graphs admit separators with special properties.

We say that a graph $G = (V, E, \ell)$ admits a λ-separator if there exists a set $R \subseteq V$ whose removal partitions G into connected components, each of size at most λn, where $1/2 \leq \lambda < 1$. Lipton and Tarjan [17] showed that every planar graph has a 2/3-separator R of size $\mathcal{O}(\sqrt{n})$. Later on, Gupta et al. [12] and Thorup [24] independently observed that one can modify their construction to obtain a 2/3-separator R, with the additional property that R consists of vertices belonging to shortest paths from G (note that this R is not guaranteed to be small). We briefly review the construction of such *shortest path separators*.

Let $G = (V, E, \ell)$ be a triangulated planar graph (the triangulation is guaranteed by adding infinity edge lengths among the missing edges). Further, let us fix an arbitrary shortest path tree A rooted at some vertex r. Then, it can be inferred from the work of Lipton and Tarjan [17] that there always exists a non-tree edge $e = \{u, v\}$ of A such that the fundamental cycle $\mathcal{C}$ in $A \cup \{e\}$, formed by adding the non-tree edge e to A, gives a 2/3-separator for G. Because A is a tree, the separator will consist of two paths from the $\mathrm{lca}(u, v)$ to u and v. We denote such paths by P_1 and P_2, respectively. Both paths are

shortest paths as they belong to A. We will show how to use such separators to obtain terminal path covers. Before proceeding, we give the following preprocessing step.

Given a planar graph $G = (V, E, \ell)$ with $T \subset V$, the algorithm MinorSparsifier(G, T, $\mathcal{P}'$) with $\mathcal{P}'$ being the $(1, \mathcal{O}(k^2))$-TPc of G, produces an $(1, \mathcal{O}(k^4))$-DAM for G. Thus, without loss of generality, we may assume that G has at most $\mathcal{O}(k^4)$ vertices.

Stretch-3 Guarantee. When solving a graph problem, it is often that the problem can be more easily solved for simpler graph instances, e.g., trees. Driven by this, it is desirable to reduce the problem from arbitrary graphs to one or several tree instances, possibly allowing a small loss in the quality of the solution. Along the lines of such an approach, Gupta et al. [12] gave the following definition in the context of shortest path distances.

▶ **Definition 28** (Forest Cover)**.** Given a graph $G = (V, E, \ell)$, a forest cover (with stretch α) of G is a family $\mathcal{F}$ of subforests $\{F_1, F_2, \ldots, F_k\}$ of G such that for every $u, v \in V$, there is a forest $F_i \in \mathcal{F}$ such that $d_G(u, v) \leq d_{F_i}(u, v) \leq \alpha \cdot d_G(u, v)$.

If we restrict our attention to planar graphs, Gupta et al. [12] used shortest path separators (as described above) to give a divide-and-conquer algorithm for constructing forest covers with small guarantees on the stretch and size. Here, we slightly modify their construction for our purpose. Before proceeding to the algorithm, we give the following useful definition.

▶ **Definition 29.** Let t be a terminal and let P be a shortest path in G. Then $t^P_{\min}$ denotes the vertex of P that minimizes $d_G(t, p)$, for all $p \in P$, breaking ties lexicographically.

Algorithm 2 ForestCover (planar graph G, terminals T)

1: **if** $|V(G)| \leq 1$ **then return** $V(G)$
2: Compute a 2/3-separator $\mathcal{C}$ consisting of shortest paths P_1 and P_2 for G.
3: **for** $i = 1, 2$ **do**
4: Contract P_i to a single vertex p_i and compute a shortest path tree L_i from p_i.
5: Expand back the contracted edges in L_i to get the tree L'_i.
6: **for** every terminal $t \in T$ **do**
7: Add $t^{P_i}_{\min}$ as a terminal in the tree L'_i.
8: Let (G_1, T_1) and (G_2, T_2) be the resulting connected graphs from $G \setminus \mathcal{C}$, where T_1 and T_2 are disjoint subsets of the terminals T induced by $\mathcal{C}$.
`// Note that all distances involving terminals from` $\mathcal{C}$ `are taken care of.`
9: **return** $\bigcup_{i=1}^{2} L'_i \cup \bigcup_{i=1}^{2}$ ForestCover(G_i, T_i).

Algorithm 3 PlanarTPc-1 (planar graph G, terminals T)

1: Set $\mathcal{P}' = \emptyset$. Set $\mathcal{F} =$ ForestCover(G, T).
2: **for** every forest $F_i \in \mathcal{F}$ **do**
3: Let R_i be the terminal set of F_i and let $\mathcal{P}'_i$ be the (trivial) $(1, \mathcal{O}(k^2))$-TPc of F_i;
4: Compute $F'_i =$ MinorSparsifier$(F_i, R_i, \mathcal{P}'_i)$.
5: Add the shortest paths corresponding to the edges of F'_i to $\mathcal{P}'$.
6: **return** $\mathcal{P}'$

▶ **Theorem 30** ([12], Theorem 5.1)**.** *Given a planar graph $G = (V, E, \ell)$ with $T \subset V$,* ForsetCover(G, T) *produces a stretch-3 forest cover with $\mathcal{O}(\log |V|)$ forests.*

We note that the original construction does not consider terminal vertices, but this does not worsen neither the stretch nor the size of the cover. The only difference here is that we need to add at most k new terminals to each forest compared to the original number of terminals in the input graph. This modification affects our bounds only by a constant factor.

▶ **Lemma 31.** *Given a planar graph $G = (V, E, \ell)$ with $T \subset V$,* PlanarTPc-1(G, T) *produces an $(3, \mathcal{O}(k \log k))$-*TPc *$\mathcal{P}'$ for G.*

Proof. We first review the following simple fact, whose proof can be found in [15].

▶ **Fact 32.** *Given a forest $F = (V, E, \ell)$ with terminals $T \subset V$ and $\mathcal{P}'$ being the (trivial) $(1, \mathcal{O}(k^2))$-TPc of F, the procedure* MinorSparsifier$(F, T, \mathcal{P}')$ *outputs an $(1, k)$-DAM.*

Let us proceed with the analysis. Observe that from the Preprocessing Step our input graph G has at most $\mathcal{O}(k^4)$ vertices. Thus, applying Theorem 30 on G gives a stretch-3 forest cover $\mathcal{F}$ of size $\mathcal{O}(\log k)$. In addition, recall that all shortest paths are unique in G.

Next, let F_i by any forest from $\mathcal{F}$. By construction, we note that each tree belonging to F_i has the nice property of being a concatenation of a given shortest path with another shortest path tree. We will exploit this in order to show that every edge of the minor F_i' for F_i corresponds to the (unique) shortest path between its endpoints in G.

To this end, let $e' = (u, v)$ be an edge of F_i' that does not exist in F_i. Since F_i' is a minor of F_i, we can map back e' to the path $\Pi_{u,v}$ connecting u and v in F_i. Because of the additional terminals $u_{\min}^{P_i}$ added to F_i, we claim that $\Pi_{u,v}$ is entirely contained either in some shortest path tree L_j or some shortest path separator P_j. Using the fact that subpaths of shortest paths are shortest paths, we conclude that the length of the path $\Pi_{u,v}$ (or equivalently, the length of edge e') corresponds to the unique shortest path connecting u and v in G. The same argument is repeatedly applied to every such edge of F_i'.

By construction we know that F_i has at most $2k$ terminals. Using Fact 32 we get that F_i' contains at most $4k$ edges. Since there are $\mathcal{O}(\log k)$ forests, we conclude that the terminal path cover $\mathcal{P}'$ consists of $\mathcal{O}(k \log k)$ shortest paths. The stretch guarantee follows directly from that of cover $\mathcal{F}$, since F_i' exactly preserves all distances between terminals in F_i. ◀

▶ **Theorem 33.** *Any planar graph $G = (V, E, \ell)$ with $T \subset V$ admits a $(3, \widetilde{\mathcal{O}}(k^2))$-*DAM.

Indeed, there is another distance oracle that yields better distortion $(1 + \varepsilon)$ (see [24, 14]). Similar to the above, we prove the following theorem; its proof is deferred to the full version.

▶ **Theorem 34.** *Any planar graph $G = (V, E, \ell)$ with $T \subset V$ admits an $(1 + \varepsilon, \widetilde{\mathcal{O}}((k/\varepsilon)^2)$-*DAM.

7 Discussion and Open Problems

We note that there remain gaps between some of the best upper and lower bounds, e.g., for general graphs and distortion $3 - \epsilon$, the lower bound is $\Omega(k^{6/5})$, while for distortion 3, our upper bound is $\mathcal{O}(k^3)$. Improving the bounds is an interesting open problem.

Our techniques for showing upper bounds rely heavily on the spanner reduction. For planar graphs, Krauthgamer et al. [15] showed that to achieve distortion $1 + o(1)$, $\Omega(k^2)$ non-terminals are needed; we bypass the spanner reduction to construct an $(1 + \varepsilon, \widetilde{\mathcal{O}}(k/\varepsilon)^2)$-DAM, which is tight up to a poly-logarithmic factor. It is an interesting open question on whether similar guarantees can be achieved for general graphs.

Acknowledgements. The authors thank Veronika Loitzenbauer and Harald Räcke for the helpful discussions.

References

1. Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.
2. Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards (1 + *eps*)-approximate flow sparsifiers. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 279–293, 2014. doi:10.1137/1.9781611973402.20.
3. Amitabh Basu and Anupam Gupta. Steiner point removal in graph metrics. http://www.ams.jhu.edu/∼abasu9/papers/SPR.pdf, 2008.
4. T.-H. Chan, Donglin Xia, Goran Konjevod, and Andrea Richa. A tight lower bound for the steiner point removal problem on trees. In *9th International Workshop on Approximation, Randomization, and Combinatorial Optimization*, APPROX'06/RANDOM'06, pages 70–81, Berlin, Heidelberg, 2006. Springer-Verlag. doi:10.1007/11830924_9.
5. Moses Charikar, Tom Leighton, Shi Li, and Ankur Moitra. Vertex sparsifiers and abstract rounding algorithms. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 265–274, 2010. doi:10.1109/FOCS.2010.32.
6. Yang-Hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *SIGMETRICS*, pages 1–12, 2000.
7. Julia Chuzhoy. On vertex sparsifiers with steiner nodes. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19-22, 2012*, pages 673–688, 2012. doi:10.1145/2213977.2214039.
8. Don Coppersmith and Michael Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM J. Discrete Math.*, 20(2):463–501, 2006. doi:10.1137/050630696.
9. Matthias Englert, Anupam Gupta, Robert Krauthgamer, Harald Räcke, Inbal Talgam-Cohen, and Kunal Talwar. Vertex sparsifiers: New results from old techniques. *SIAM J. Comput.*, 43(4):1239–1262, 2014. doi:10.1137/130908440.
10. Paul Erdös. Extremal problems in graph theory. *Theory of Graphs and its Applications (Proc. Symposium Smolenice)*, 1963.
11. Anupam Gupta. Steiner points in tree metrics don't (really) help. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 220–227, 2001. URL: http://dl.acm.org/citation.cfm?id=365411.365448.
12. Anupam Gupta, Amit Kumar, and Rajeev Rastogi. Traveling with a pez dispenser (or, routing issues in MPLS). *SIAM J. Comput.*, 34(2):453–474, 2004. doi:10.1137/S0097539702409927.
13. Lior Kamma, Robert Krauthgamer, and Huy L. Nguyen. Cutting corners cheaply, or how to remove steiner points. *SIAM J. Comput.*, 44(4):975–995, 2015.
14. Ken-ichi Kawarabayashi, Philip N. Klein, and Christian Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *Automata, Languages and Programming – 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 135–146, 2011. doi:10.1007/978-3-642-22006-7_12.
15. Robert Krauthgamer, Huy L Nguyen, and Tamar Zondiner. Preserving terminal distances using minors. *SIAM Journal on Discrete Mathematics*, 28(1):127–141, 2014.

16 Frank Thomson Leighton and Ankur Moitra. Extensions and limits to vertex sparsification. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 47–56, 2010. doi:10.1145/1806689.1806698.

17 Richard J. Lipton and Robert Endre Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, 1979. doi:10.1137/0136016.

18 Konstantin Makarychev and Yury Makarychev. Metric extension operators, vertex sparsifiers and lipschitz extendability. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 255–264, 2010. doi:10.1109/FOCS.2010.31.

19 Ankur Moitra. Approximation algorithms for multicommodity-type problems with guarantees independent of the graph size. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009, October 25-27, 2009, Atlanta, Georgia, USA*, pages 3–12, 2009. doi:10.1109/FOCS.2009.28.

20 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 255–264, 2008. doi:10.1145/1374376.1374415.

21 Harald Räcke, Chintan Shah, and Hanjo Täubig. Computing cut-based hierarchical decompositions in almost linear time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 227–238, 2014. doi:10.1137/1.9781611973402.17.

22 Michael Scharf, Gordon T. Wilfong, and Lisa Zhang. Sparsifying network topologies for application guidance. In *IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, Ottawa, ON, Canada, 11-15 May, 2015*, pages 234–242, 2015.

23 Andrew Thomason. An extremal function for contractions of graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 95:261–265, 3 1984.

24 Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004. doi:10.1145/1039488.1039493.

25 Rephael Wenger. Extremal graphs with no C^4's, C^6's, or C^{10}'s. *J. Comb. Theory, Ser. B*, 52(1):113–116, 1991.

26 Richard M. Wilson. An existence theory for pairwise balanced designs, III: proof of the existence conjectures. *J. Comb. Theory, Ser. A*, 18(1):71–79, 1975.

27 David P. Woodruff. Lower bounds for additive spanners, emulators, and more. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 389–398, 2006.

Distance Labeling Schemes for Trees

Stephen Alstrup*[1], Inge Li Gørtz†[2], Esben Bistrup Halvorsen[3], and Ely Porat[4]

1 **University of Copenhagen, Copenhagen, Denmark**
`s.alstrup@di.ku.dk`
2 **Technical University of Copenhagen, Copenhagen, Denmark**
`inge@dtu.dk`
3 **University of Copenhagen, Copenhagen, Denmark**
`esben@bistruphalvorsen.dk`
4 **Bar-Ilan University, Bar Ilan, Israel**
`porately@cs.biu.ac.il`

Abstract

We consider distance labeling schemes for trees: given a tree with n nodes, label the nodes with binary strings such that, given the labels of any two nodes, one can determine, by looking only at the labels, the distance in the tree between the two nodes.

A lower bound by Gavoille et al. [Gavoille et al., J. Alg., 2004] and an upper bound by Peleg [Peleg, J. Graph Theory, 2000] establish that labels must use $\Theta(\log^2 n)$ bits[1]. Gavoille et al. [Gavoille et al., ESA, 2001] show that for very small approximate stretch, labels use $\Theta(\log n \log\log n)$ bits. Several other papers investigate various variants such as, for example, small distances in trees [Alstrup et al., SODA, 2003].

We improve the known upper and lower bounds of exact distance labeling by showing that $\frac{1}{4}\log^2 n$ bits are needed and that $\frac{1}{2}\log^2 n$ bits are sufficient. We also give $(1+\varepsilon)$-stretch labeling schemes using $\Theta(\log n)$ bits for constant $\varepsilon > 0$. $(1+\varepsilon)$-stretch labeling schemes with polylogarithmic label size have previously been established for doubling dimension graphs by Talwar [Talwar, STOC, 2004].

In addition, we present matching upper and lower bounds for distance labeling for caterpillars, showing that labels must have size $2\log n - \Theta(\log\log n)$. For simple paths with k nodes and edge weights in $[1, n]$, we show that labels must have size $(k-1)/k \log n + \Theta(\log k)$.

1998 ACM Subject Classification E.1 Distributed data structures, E.1 Graphs and networks, G.2.2 Graph Theory

Keywords and phrases Distributed computing, Distance labeling, Graph theory, Routing, Trees

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.132

1 Introduction

A *distance labeling scheme* for a given family of graphs assigns *labels* to the nodes of each graph in the family such that, given the labels of two nodes in the graph and no other information, it is possible to determine the shortest distance between the two nodes. The labels are assumed to be composed of bits, and the goal is to make the worst-case label size as

* Research partly supported by the FNU project AlgoDisc – Discrete Mathematics, Algorithms, and Data Structures.
† Research partly supported by the Danish Research Council (DFF 1323-00178) and the Danish Research Council under the Sapere Aude Program (DFF 4005-00267).
[1] Throughout this paper we use log for $\log_2$.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 132; pp. 132:1–132:16


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

Table 1 Results presented in this paper. $\varepsilon > 0$ is a constant.

Problem	Lower bound	Upper bound
Exact, general trees	$\frac{1}{4}\log^2 n$	$\frac{1}{2}\log^2 n$
$(1+\varepsilon)$-stretch, general trees	$\Theta(\log n)$	
Caterpillars	$2\log n - \Theta(\log\log n)$	
Weighted paths, k nodes, weights in $[1,n]$	$\frac{k-1}{k}\log n + \Theta(\log k)$	

small as possible. Labeling schemes are also called *implicit representation of graphs* [64, 71]. The problem of finding implicit representations with small labels for specific families of graphs was introduced in the 1960s [16, 17], and efficient labeling schemes were introduced in [44, 57]. Distance labeling for general graphs has been considered since the 1970/80s [40, 72], and later for various restricted classes of graphs and/or approximate distances, often tightly related to distance oracle and routing problems, see e.g. [7]. This paper focuses on distance labels for the well studied case of trees.

Exact distances. In [61] Peleg presented an $O(\log^2 n)$ bits distance labeling scheme for general unweighted trees. In [39] Gavoille et al. proved that distance labels for unweighted binary trees require $\frac{1}{8}\log^2 n - O(\log n)$ bits and presented a scheme with $1/(\log 3 - 1)\log n \approx 1.7\log n$ bits. This paper presents a scheme of size $\frac{1}{2}\log^2 n + O(\log n)$ and further reduces the gap by showing that $\frac{1}{4}\log^2 n - O(\log n)$ bits are needed. Our upper bound is a somewhat straightforward application of a labeling scheme for nearest common ancestors [8, 9].

Approximate distances. Let $\text{dist}_T(x,y)$ denote the shortest distance between nodes x, y in a tree T. An r-additive approximation scheme returns a value $\text{dist}'_T(x,y)$, where $\text{dist}_T(x,y) \leq \text{dist}'_T(x,y) \leq \text{dist}_T(x,y)+r$. An s-stretched approximation scheme returns a value $\text{dist}'_T(x,y)$, where $\text{dist}_T(x,y) \leq \text{dist}'_T(x,y) \leq \text{dist}_T(x,y)\cdot s$. For trees of height h Gavoille et al. [32, theorem 4] gave a 1-additive $O(\log n \log h)$ bit labeling scheme. However, using an extra bit in the label for the node depth modulo 2, it is easy to see that any 1-additive scheme can be made exact. Gavoille et al. [32] also gave upper and lower bounds of $\Theta(\log\log n \log n)$ bits for $(1+1/\log n)$-stretched distance. This paper presents a scheme of size $\Theta(\log n)$ for $(1+\varepsilon)$-stretch for constant $\varepsilon > 0$. Labeling schemes for $(1+\varepsilon)$-stretch with polylogarithmic size label have previously been given for graphs of doubling dimension [65] and planar graphs [67].

Distances in caterpillars and paths. Labeling schemes for caterpillars have been studied for various queries, e.g., adjacency [15]. Here we present upper and lower bounds showing that distance labeling caterpillars requires $2\log n - \Theta(\log\log n)$ bits. The upper bound is constructed by reduction to the case of *weighted paths* with $k > 1$ nodes and positive integer edge weights in $[1,n]$, for which we give upper and lower bounds showing that labels must have size $\frac{k-1}{k}\log n + \Theta(\log k)$.

1.1 Related work

Distances in trees with small height. It is known that, for unweighted trees with bounded height h, labels must have size $\Theta(\log n \log h)$. The upper bound follows from [32, Theorem 2]

and the lower bound from [39, Section 3][2]. In [45] distance labeling for various restricted class of trees, including trees with bounded height, is considered, and in [66] another distance labeling scheme for unweighted trees using $O(\log n \log h)$ bits is given.

Small distances in trees. Distances in a tree between nodes at distance at most k can be computed with labels of size $\log n + O(k\sqrt{\log n})$ [46]. In [5] it is shown that size $\log n + \Theta(\log \log n)$ are needed for labeling schemes supporting both parent and sibling queries. More generally, [5] shows that, using labels of size $\log n + O(\log \log n)$, the distance between two nodes can be determined if it is at most k for some constant k, which is optimal for $k > 1$. In [33, 34] further improvements are given for small distances in trees. For $k = 1$, corresponding to adjacency testing, there is a sequence of papers that improve the second order term, recently ending with [6] which establishes that $\log n + \Theta(1)$ bits are sufficient.

Various other cases for trees. Distance labeling schemes for various other cases have been considered, e.g., for weighted trees [32, 39, 61], dynamic trees [52], and a labeling scheme variation with extra free lookup [50, 51].

Exact and approximate distances in graphs. Distance labeling schemes for general graphs [7, 39, 40, 64, 70, 72] and various restricted graphs exist, e.g., for bounded tree-width, planar and bounded degree [39], distance-hereditary [36], bounded clique-width [22], some non-positively curved plane [19], interval [37] and permutation graphs [14]. Approximate distance labeling schemes, both additive and stretched, are also well studied; see e.g., [18, 26, 32, 35, 39, 41, 42, 53, 61, 69]. An overview of distance labeling schemes can be found in [7].

1.2 Second order terms for labeling schemes are theoretically well studied

Chung's solution in [20] gives labels of size $\log n + O(\log \log n)$ for adjacency labeling in trees, which was improved to $\log n + O(\log^* n)$ in FOCS'02 [13] and in [15, 20, 29, 30, 47] to $\log n + \Theta(1)$ for various special cases. Finally it was improved to $\log n + \Theta(1)$ for general trees in FOCS'15 [6].

A recent STOC'15 paper [11] improves label size for adjacency in general graphs from $n/2 + O(\log n)$ [44, 56] to $n/2 + O(1)$ almost matching an $(n-1)/2$ lower bound [44, 56].

Likewise, the second order term for ancestor relationship is improved in a sequence of STOC/SODA papers [2, 4, 12, 30, 31] (and [1, 47]) to $\Theta(\log \log n)$, giving labels of size $\log n + \Theta(\log \log n)$.

Somewhat related, *succinct data structures* (see, e.g., [24, 27, 28, 58, 59]) focus on the space used in addition to the information theoretic lower bound, which is often a lower order term with respect to the overall space used.

1.3 Labeling schemes in various settings and applications

Using labeling schemes to compute locally and distributed, it is possible to avoid costly access to large global tables. Such properties are used, e.g., in XML search engines [2], network routing and distributed algorithms [23, 25, 68, 69], dynamic and parallel settings [21, 52], graph representations [44], and other applications [48, 49, 60, 61, 62].

[2] We thank Gavoille for pointing this out.

Trees have always been a subject of special interest in the routing literature (see e.g. [3, 10, 55, 73]). As pointed out by Leighton [54], many networks are simply trees, and, moreover, routing in general graphs may be done using only the edges of a spanning tree of the graph. In [68] a $O(\log n)$ labeling routing scheme for tree is given. This e.g. combined with one of the results in this paper makes it possible using labels of size $O(\log n)$ not only to route packets in trees, but also estimate the distance to its destination.

Various computability requirements are sometimes imposed on labeling schemes [2, 44, 48]. This paper assumes the RAM model.

2 Preliminaries

Trees. Given nodes u, v in a rooted tree T, u is an *ancestor* of v and v is a *descendant* of u, if u is on the unique path from v to the root. For a node u of T, denote by T_u be the subtree of T consisting of all the descendants of u (including itself). The *depth* of u is the number of edges on the unique simple path from u to the root of T. The nearest common ancestor (NCA) of two nodes is the unique common ancestor with largest depth. Let $T[u, v]$ denote the nodes on the simple path from u to v in T. The variants $T(u, v]$ and $T[u, v)$ denote the same path without the first and last node, respectively. The distance between u and v is the number $\text{dist}(u, v) = |T(u, v]|$. We set $\text{distroot}(v) = \text{dist}(v, r)$, where r is the root of T. A *caterpillar* is a tree whose non-leaf nodes form a path, called the *spine*.

Heavy-light decomposition (From [63]). Let T be a rooted tree. The nodes of T are classified as either *heavy* or *light* as follows. The root r of T is light. For each non-leaf node v, pick one child w where $|T_w|$ is maximal among the children of v and classify it as heavy; classify the other children of v as light. The *apex* of a node v is the nearest light ancestor of v. By removing the edges between light nodes and their parents, T is divided into a collection of *heavy paths*. Any given node v has at most $\log n$ light ancestors (see [63]), so the path from the root to v goes through at most $\log n$ heavy paths.

Bit strings. A bit string s is a member of the set $\{0, 1\}^*$. We denote the length of a bit string s by $|s|$, the ith bit of s by s_i, and the concatenation of two bit strings s, s' by $s \circ s'$. We say that s_1 is the most significant bit of s and $s_{|s|}$ is the least significant bit.

Labeling schemes. An *distance labeling scheme* for trees of size n consists of an *encoder* e and a *decoder* d. Given a tree T, the encoder computes a mapping $e_T : V(T) \to \{0, 1\}^*$ assigning a *label* to each node $u \in V(T)$. The decoder is a mapping $d : \{0, 1\}^* \times \{0, 1\}^* \to \mathbb{Z}^+$, where $\mathbb{Z}^+$ denotes the positive integers, such that, given any tree T and any pair of nodes $u, v \in V(T)$, $d(e(u), e(v)) = \text{dist}(u, v)$. Note that the decoder does not know T. The *size* of a labeling scheme is defined as the maximum label size $|e_T(u)|$ over all trees T and all nodes $u \in V(T)$. If, for all trees T, the mapping e_T is injective we say that the labeling scheme assigns *unique* labels.

3 Distances on weighted paths

In this section we study the case of paths with k nodes and integral edge weights in $[1, n]$. The solution to this problem will later be used to establish the upper bound for caterpillars.

3.1 Upper Bound

▶ **Theorem 1.** *There exist a distance labeling scheme for paths with k nodes and positive integral edge weights in $[1, n]$ with labels of size $\frac{k-1}{k}\log n + O(\log k)$.*

Proof. We begin by considering the family of paths with k nodes, integral edge weights and diameter $< n$. We shall prove that there exists a distance labeling scheme for this family with labels of size $\frac{k-1}{k}\log n + \log k + \mathrm{O}(\log\log k)$.

So consider such a path, and root it in one of its end nodes, denoted v_0. Denote the nodes on the path $v_0, \ldots, v_{k-1}$ in order. Let $d_i = \mathrm{distroot}(v_i)$ and note that, by assumption, $d_i < n$ for all i. We will let the label for v_i store the number $d_i + x$ for some $x < n$ that allows us to represent $d_i + x$ compactly. Since we use the same x for all nodes, we can easily compute the distance between any pair of nodes v_i, v_j as $|(d_i + x) - (d_j + x)|$.

Since we choose $x < n$, the largest number stored in a label will be $d_k + x < 2n$, which can be represented with *exactly* $L = \lceil \log(2n) \rceil$ bits. Divide those L bits to $k + 1$ segments, whereof k have $\ell = \lfloor L/k \rfloor$ bits and the last segment contains the remaining bits. The first segment, segment 0, will contain the ℓ least significant bits, segment 1 the following ℓ bits and so on. We will choose x such that the representation of $d_i + x$ has 0s in all the bits in the i'th segment. If we manage to do so, we will be able to encode each $d_i + x$ with $L - \ell + \lceil \log k \rceil$ bits. Indeed, we can use exactly $\lceil \log k \rceil$ bits to represent i, and the next $L - \ell$ bits to represent $d_i + x$ where we skip the i'th segment. Prefixing with a string in the form $0^{\lceil \log \lceil \log k \rceil \rceil}1$, we get a string from which we can determine the number of bits needed to write $\lceil \log k \rceil$ and therefrom the numbers i and $d_i + x$. We use this string as the label for v_i. The label length is $L - \ell + \lceil \log k \rceil + \lceil \log \lceil \log k \rceil \rceil + 1 = \frac{k-1}{k}\log n + \log k + O(\log\log k)$.

It remains to show that there exist a number $x < n$ as described. In the following we shall, as in the above, represent numbers $< 2n$ with L bits that are divided into $k + 1$ segments whereof the first k have size ℓ. For $i < k$ and $y < 2n$, let $a(i, y)$ be a function which returns a number z with the following properties:

(i) In z, all bits outside segment i are 0.
(ii) $z + y$ has only 0s in segment i.

This function is constructed as follows. If y only has 0s in segment i, let $a(i, y) = 0$. Otherwise take the representation of y, zero out all bits outside segment i, reverse the bits in segment i and add v to the resulting number, where v has a 1 in the least significant bit of segment i and 0s in all other positions.

Note that from (i) it follows that adding z to any number will not change bits in less significant positions than segment i. We can now scan through the nodes $v_0, \ldots v_{k-1}$, increasing x by adding bits to x in more and more significant positions (in non-overlapping segments), as follows:

- Set $x = 0$.
- For $i = 1 \ldots, k - 1$, set $x = x + a(i, x + d_i)$.

After iteration i we have that $x + d_i$ in segment i only has 0s, and in the following iterations, 1s are only added to x in more significant bit positions, meaning that $d_i + x$ continues to have only 0s in segment i. Since the segments are non-overlapping, we end up with $x < n$.

For the more general family of paths with k nodes and edge weights in $[1, n]$, we simply note that the diameter of any path in this family is at most kn. Using the above result thus immediately gives us a labeling scheme with labels of size $\frac{k-1}{k}\log n + O(\log k)$. ◀

3.2 Lower bound

▶ **Theorem 2.** *Labeling scheme for distances on weighted paths with k nodes and edge weights $[1, n]$ require $\frac{k-1}{k}\log n + \Omega(\log k)$ bits.*

Proof. Let $\mathcal{F}$ denote the family of paths with k nodes and integral edge weights in $[1, n]$. We can construct all the members of $\mathcal{F}$ by selecting $(k-1)$ different edge weights in the range $[1, n]$, skipping the paths which have already been constructed by the reverse sequence of edge weights. With this construction we will at most skip half of the paths, and hence $|\mathcal{F}| \geq \frac{1}{2}n^{k-1}$. Let the worst-case label size of an optimal distance labeling scheme for such paths have length L. The number of different labels with length at most L is $N = 2^{L+1} - 1$. We can uniquely represent each of the paths in $\mathcal{F}$ with the collection of their labels, and hence $|\mathcal{F}| \leq \binom{N}{k}$. Thus, we have found that $\frac{1}{2}n^{k-1} \leq \binom{N}{k}$. Since $\binom{N}{k} \leq (Ne/k)^k$, it follows that $\frac{k-1}{k}\log n \leq \log N - \log k + O(1)$ and hence that $L \geq \frac{k-1}{k}\log n + \log k - O(1)$ as desired. ◀

Combining Theorem 2 with Theorem 1 we see that distance labels for paths of k nodes with integral weights in $[1, n]$ must have length $\frac{k-1}{k}\log n + \Theta(\log k)$.

4 Distances in caterpillars

4.1 Upper bound

▶ **Theorem 3.** *There exist a distance labeling scheme for caterpillars with worst-case label size $2\log n - \log\log n + O(\log\log\log n)$.*

Proof. We will start by giving a simple $2\log n$ bits scheme and then improve it. The simple solution assigns two numbers to each node. The nodes on the spine save distroot and the number 0. The nodes not on the spine save their parent's distroot and a number that is unique among their siblings. The second number is required to distinguish siblings, and hence determine if the distance between two nodes is 0 or 2. The worst-case label size for this solution is $2\log n + O(1)$.

To improve the solution, we split up the nodes on the spine into two groups: (1) nodes with $> \frac{n}{k}$ leaves and (2) nodes with $\leq \frac{n}{k}$ leaves, for some parameter k to be chosen later. We add the root to the first group no matter what. Note that the first group can contain at most k nodes.

As before, all nodes save two numbers: distroot and the number 0 for spine nodes or a number to distinguish siblings. The idea is to reduce label size with $\log k$ bits by using fewer bits for the first number for nodes in the first group and for the second number for nodes in the second group.

The nodes in the first group form a path with at most k nodes and edge weights in $[1, n]$ (where each weight corresponds to the distance between the nodes in the original graph). The algorithm from Theorem 1 will add a number x, which is less than the diameter, which again is less than n, to the numbers representing the root distances of the nodes. Using this technique, we can, as seen in the proof of Theorem 1, encode the (modified) distroots of the nodes in the first group with only $\frac{k-1}{k}\log n + \log k + O(\log\log k)$ bits. This gives labels of size $\frac{2k-1}{k}\log n + \log k + O(\log\log k)$ for non-spine nodes whose parents are in the first group.

We will also add x to the distroots of nodes in the second group, but since $x < n$ this will not change the label size by more than a single bit. For non-spine nodes whose parents are in the second group, we need at most $\log n - \log k + O(1)$ bits for the second number, giving a total label size of $2\log n - \log k + O(1)$.

Finally, since the two numbers that form a label now have different lengths, we need an additional $O(\log\log k)$ bits to determine when one number ends and the next begins. Indeed, it wil be possible to split up labels into their components if we know the number of bits used to write $\lceil \log k \rceil$, and we represent this number with $O(\log\log k)$ bits.

Setting $k = \frac{\log n}{2\log\log n}$, we now see that our worst-case label size is the maximum of

$$2\log n - \log k + O(\log\log k) = 2\log n - \log\log n + O(\log\log\log n)$$

and

$$\begin{aligned}\frac{2k-1}{k}\log n + \log k + O(\log\log k) &= 2\log n - 2\log\log n + \log\log n + O(\log\log\log n)\\ &= 2\log n - \log\log n + O(\log\log\log n).\end{aligned}$$

This proves the theorem. ◀

It is worth noting that the ability to distinguish nodes plays a significant part in the label size. Indeed, if the two nodes given as input to the decoder could always be assumed to be distinct, then a distance labeling scheme for caterpillars with worst-case label size $\log n + O(1)$ would be possible.

4.2 Lower bound

We present a technique that counts tuples of labels that are known to be distinct and compares the result to the number of tuples one can obtain with labels of size L. The technique may have applications to distance labeling for other families of graphs.

▶ **Theorem 4.** *For any $n \geq 4$, any distance labeling scheme for the family of caterpillars with at most n nodes has a worst-case label size of at least $2\lfloor\log n\rfloor - \lfloor\log\lfloor\log n\rfloor\rfloor - 4$.*

Proof. Set $k = \lfloor\log n\rfloor$ and $m = 2^k$. Let $(i_1, \ldots, i_k)$ be a sequence of k numbers from the set $\{1, \ldots, m/2\}$ with the only requirement being that $i_1 = 1$. Now consider, for each such sequence, the caterpillar whose main path has length $m/2$ and where, for $t = 1, \ldots, k$, the node in position i_t has $\lfloor m/2k\rfloor$ leaf children (not on the main path). We shall refer to these children as the *t'th group*. Note that two disjoint groups of children may be children of the same node if $i_t = i_s$ for some s, t. Each of these caterpillars has $m/2 + k\lfloor m/2k\rfloor \leq m \leq n$ nodes.

Suppose that σ is a distance labeling scheme for the family of caterpillars, and consider one of the caterpillars defined above. Given distinct nodes u, v not on the main path, their distance will be $\mathrm{dist}(u, v) = |i_s - i_t| + 2$, where i_s and i_t are the positions on the main path of the parents of u and v, respectively. In particular, if $s = 1$, so that $i_s = 1$, then $\mathrm{dist}(u, v) = i_t + 1$. Thus, if σ has been used to label the nodes of the caterpillar, the number i_t for a child in the t'th group can be uniquely determined from its label together with the label of any of the children from the first group. It follows that any k-tuple of labels $(l_1, \ldots, l_k)$ where l_t is a label of a child in the t'th group uniquely determines the sequence $(i_1, \ldots, i_k)$. In particular, k-tuples of labels from distinct caterpillars must be distinct. Of course, k-tuples of labels from the same caterpillar must also be distinct, since labels are unique in a distance labeling scheme.

Now, there are $(m/2)^{k-1}$ choices for the sequence $(i_1, \ldots, i_k)$, and hence there are $(m/2)^{k-1}$ different caterpillars in this form. For each of these, there are $\lfloor m/2k\rfloor^k$ different choices of k-tuples of labels. Altogether, we therefore have $(m/2)^{k-1}\lfloor m/2k\rfloor^k$ distinct k-tuples of labels. If the worst-case label size of σ is L, then we can create at most $(2^{L+1} - 1)^k$

distinct k-tuples of labels, so we must have $(m/2)^{k-1}\lfloor m/2k \rfloor^k \leq (2^{L+1}-1)^k$. From this it follows that

$$\begin{aligned} L &\geq \lfloor \frac{k-1}{k}(\log m - 1) + \log\lfloor m/2k \rfloor \rfloor \\ &\geq \lfloor \frac{(k-1)^2}{k} + k - \log k \rfloor - 2 \\ &\geq 2k - \lfloor \log k \rfloor - 4 \\ &= 2\lfloor \log n \rfloor - \lfloor \log \lfloor \log n \rfloor \rfloor - 4. \end{aligned}$$

◀

5 Exact distances in trees

5.1 Upper bound

Let u, v be nodes in a tree T and let w be their nearest common ancestor. We then have

$$\operatorname{dist}(u,v) = \operatorname{distroot}(u) - \operatorname{distroot}(v) + 2\operatorname{dist}(w,v)\,. \tag{1}$$

If $w = u$ so that u is an ancestor of v, then the above equation is just a difference of distroots, which can be stored for each node with $\log n$ bits. The same observation clearly holds if $w = v$.

Assume now that $w \notin \{u, v\}$ so that u and v are not ancestors of each other. Consider the heavy-light decomposition [63] described in the preliminaries. At least one of the nodes u and v must have an ancestor which is a light child of w. Assume that it is v. Now, v has at most $\log n$ light ancestors. Saving the distance to all of them together with distroot gives us sufficient information to compute the distance between u and v using equation (1). This is the idea behind Theorem 6 below.

By examining the NCA labeling scheme from [8, 9], we see that it can easily be extended as follows.

▶ **Lemma 5** ([8, 9]). *There exists an NCA labeling scheme of size $O(\log n)$. For any two nodes u, v the scheme returns the label of $w = \operatorname{nca}(u,v)$ as well as:*

- *which of u and v (if any) have a light ancestor that is a child of w; and*
- *the number of light nodes on the path from the root to w and from w to u and v, respectively.*

▶ **Theorem 6.** *There exists a distance labeling scheme for trees with worst-case label size $\frac{1}{2}\log^2 n + O(\log n)$.*

Proof. We use $O(\log n)$ bits for the extended NCA labeling in Lemma 5 and for distroot. Using (1) it now only remains to efficiently represent, for each node, the distance to all its light ancestors.

We consider the light ancestors of a node v encountered on the path from the root to v. The distance from v to the root is at most $n-1$ and can therefore be encoded with *exactly* $\lceil \log n \rceil$ bits (by adding leading zeros if needed). By construction of the heavy-light decomposition, the next light node on the path to v will be the root of a subtree of size at most $n/2$, meaning that the distance from v to that ancestor is at most $n/2 - 1$ and can be encoded with *exactly* $\lceil \log n \rceil - 1$ bits. Continuing this way, we encode the i'th light ancestor on the path from the root to v with exactly $\lceil \log n \rceil - i$ bits. When we run out of light ancestors, we concatenate all the encoded distances, resulting in a string of length at most

$$\lceil \log n \rceil + (\lceil \log n \rceil - 1) + \cdots + 2 + 1 = \frac{1}{2}\lceil \log n \rceil^2 + \frac{1}{2}\lceil \log n \rceil\,.$$

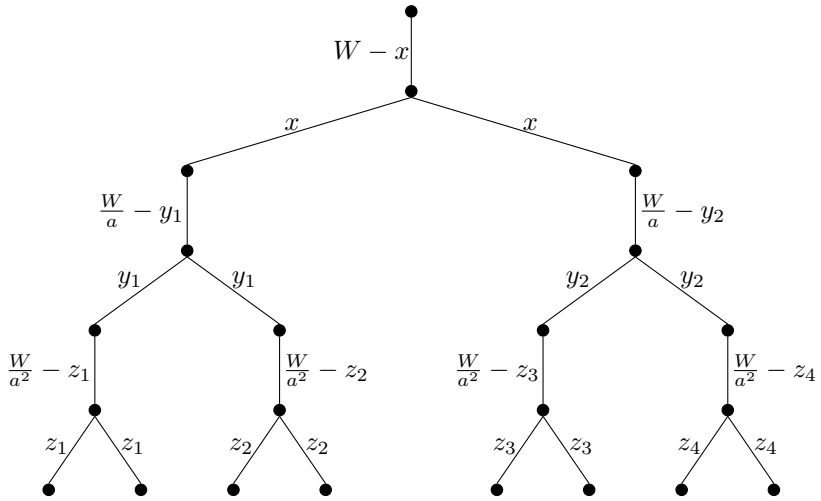


Figure 1 An (h, W, a)-tree, where $h = 3$. We require that $x < W$, $y_1, y_2 < W/a$ and $z_1, \ldots, z_4 < W/a^2$.

We can use $O(\log n)$ extra bits to encode n and to separate all sublabels from each other. The decoder can now determine $\lceil \log n \rceil$ and split up the entries in the list of distances. When applying formula (1), it can then determine the distance between v and w by adding together the relevant distances in the list of light ancestors, using the fact from Lemma 5 that it knows the number of light ancestors from v to w. ◀

5.2 Lower bound

In the case of general trees, Gavoille et al [39] establish a lower bound of $\frac{1}{8}\log^2 n - O(\log n)$ using an ingenious technique where they apply a distance labeling scheme to a special class of trees called (h, M)-trees[3]. The following uses a generalization of (h, M)-trees to improve their ideas and leads to a lower bound of $\frac{1}{4}\log^2 n - O(\log n)$.

(h, W, a)-trees. We begin with some definitions. For integers $h, W \geq 0$ and a number $a \geq 1$ such that W/a^i is integral for all $i = 0, \ldots, h$, an (h, W, a)-*tree* is a rooted binary tree T with edge weights in $[0, W]$ that is constructed recursively as follows. For $h = 0$, T is just a single node. For $h = 1$, T is a claw (i.e. a star with three edges) with edge weights $x, x, W - x$ for some $0 \leq x < W$ rooted at the leaf node of the edge with weight $W - x$. For $h > 1$, T consists of an $(1, W, a)$-tree whose two leaves are the roots of two $(h - 1, W/a, a)$-trees T_0, T_1. We shall denote an (h, W, a)-tree constructed in this way by $T = \langle T_0, T_1, x \rangle$ An example for $h = 3$ can be seen in Figure 1. Note that the case $a = 1$ simply corresponds to the (h, W)-trees defined in [39].

It is easy to see that an (h, W, a)-tree has 2^h leaves and $3 \cdot 2^h - 2$ nodes. Further, it is straightforward to see that, if u, v are leaves in an (h, W, a)-tree $T = \langle T_0, T_1, x \rangle$, then

$$\mathrm{dist}_T(u, v) = \begin{cases} 2W\frac{a^{-1} - a^{-h}}{1 - a^{-1}} + 2x, & \text{if } u \in T_0 \text{ and } v \in T_1, \text{ or vice versa,} \\ \mathrm{dist}_{T_i}(u, v), & \text{if } u, v \in T_i \text{ for some } i = 0, 1. \end{cases} \tag{2}$$

[3] Note that their exposition has some minor errors as pointed out (and corrected) in [43].

Leaf distance labeling schemes. In the following we shall consider *leaf distance labeling schemes* for the family of (h, W, a)-trees: that is, distance labeling schemes where only the leaves in a tree need to be labeled, and where only leaf labels can be given as input to the decoder. Since an ordinary distance labeling scheme obviously can be used only for leaves, any lower bound on worst-case label sizes for a leaf distance labeling scheme is also a lower bound for an ordinary distance labeling scheme. We denote by $g(h, W, a)$ the smallest number of labels needed by an optimal leaf distance labeling scheme to label all (h, W, a)-trees.

▶ **Lemma 7.** *For all $h \geq 1$ and $W \geq 2$, $g(h, W, a)^2 \geq W g(h-1, W^2/a^2, a^2)$.*

Proof. Fix an optimal leaf distance labeling scheme σ which produces exactly $g(h, W, a)$ distinct labels for the family of (h, W, a)-trees. For leaves u and v in an (h, W, a)-tree, denote by $l(u)$ and $l(v)$, respectively, the labels assigned by σ. For $x = 0, \ldots, W-1$, let $S(x)$ be the set consisting of pairs of labels $(l(u), l(v))$ for all leaves $u \in T_0$ and $v \in T_1$ in all (h, W, a)-trees $T = \langle T_0, T_1, x\rangle$.

The sets $S(x)$ and $S(x')$ are disjoint for $x \neq x'$, since every pair of labels in $S(x)$ uniquely determines x due to (2). Letting $S = \bigcup_{x=0}^{W-1} S(x)$, we therefore have $|S| = \sum_{x=0}^{W-1} |S(x)|$. Since S contains pairs of labels produced by σ from leaves in (h, W, a)-trees , we clearly also have $|S| \leq g(h, W, a)^2$, and hence it only remains to prove that $|S| \geq W g(h-1, W^2/a^2, a^2)$, which we shall do by showing that $|S(x)| \geq g(h-1, W^2/a^2, a^2)$ for all x.

The goal for the rest of the proof is therefore to create a leaf distance labeling scheme for $(h-1, W^2/a^2, a^2)$-trees using only labels from the set $S(x)$ for some fixed x. So let x be given and consider an $(h-1, W^2/a^2, a^2)$-tree T'. Let $V = W/a$. From T' we shall construct an $(h-1, V, a)$-tree $\phi_i(T')$ for $i = 0, 1$ such that every leaf node v in T' corresponds to nodes $\phi_i(v)$ in $\phi_i(T')$ for $i = 0, 1$. The trees $\phi_i(T')$ are defined as follows. If $h = 1$, so that T' consists of a single node, then $\phi_i(T') = T'$ for $i = 0, 1$. If $h > 1$, then T' is in the form $T' = \langle T'_0, T'_1, y\rangle$ for some $0 \leq y < V^2$. We can write y in the form $y = y_0 + y_1 V$ for uniquely determined y_0, y_1 with $0 \leq y_0, y_1 < V$. For $i = 0, 1$, we recursively define $\phi_i(T') = \langle \phi_i(T'_0), \phi_i(T'_1), y_i\rangle$. Thus, $\phi_i(T')$ is an $(h-1, V, a)$-tree that is similar to T' but where we replace the top edge weight y by edge weights y_i and, recursively, do the same for all $(h-2, V^2/a^2, a^2)$-subtrees. Note also that the corresponding edge weight $V^2 - y$ in T' automatically is replaced by the edge weight $V - y_i$ in $\phi_i(T')$ in order for $\phi_i(T')$ to be an $(h-1, V, a)$-tree.

Denote by $\phi_i(v)$ the leaf in $\phi_i(T')$ corresponding to the leaf v in T'.

Consider now the (h, W, a)-tree $T = \langle \phi_0(T'), \phi_1(T'), x\rangle$. Every leaf v in T' corresponds to the leaves $\phi_0(v), \phi_1(v)$ in T where $\phi_i(v) \in \phi_i(T')$ for $i = 0, 1$. Using formula (2) for the distances in T', it is straightforward to see that

$$\mathrm{dist}_{T'}(u, v) = \big(\mathrm{dist}_{\phi_0(T')}(\phi_0(u), \phi_0(v)) \bmod (2V)\big) + V\, \mathrm{dist}_{\phi_1(T')}(\phi_1(u), \phi_1(v)). \tag{3}$$

We can now apply the leaf distance labeling scheme σ to T and obtain a label for each leaf node in T. In particular, the pair of leaves $(\phi_0(v), \phi_1(v))$ corresponding to a node v in T' will receive a pair of labels. We use this pair to label v in T', whereby we have obtained a labeling of the leaves in T' with labels from $S(x)$. Using the formula in (3) we can construct a decoder that can compute the distance between two nodes in T' using these labels alone, and hence we have obtained a leaf distance labeling scheme for $(h-1, V^2, a^2)$-trees using only labels from $S(x)$ as desired. ◀

▶ **Lemma 8.** *For all $h \geq 1$ and $W \geq 2$, $g(h, W, a) \geq \frac{W^{h/2}}{a^{h(h-1)/4}}$.*

Proof. The proof is by induction on h. For $h = 1$ we note that an $(0, W, a)$-tree has only one node, so that $g(0, W^2/a^2, a^2) = 1$. Lemma 7 therefore yields $g(1, W, a)^2 \geq W$ from which

it follows that $g(1,W,a) \geq \sqrt{W}$. The lemma therefore holds for $h=1$. Now let $h>1$ and assume that the lemma holds for $h-1$. Lemma 7 and the induction hypothesis now yield

$$g(h,W,a)^2 \;\geq\; Wg(h-1,W^2/a^2,a^2) \;\geq\; W\frac{(W^2/a^2)^{(h-1)/2}}{a^{2(h-1)(h-2)/4}} \;=\; \frac{W^h}{a^{h(h-1)/2}}$$

from which it follows that $g(h,W,a) \geq \frac{W^{h/2}}{a^{h(h-1)/4}}$ as desired. ◀

The previous lemma implies that any (leaf and hence also ordinary) distance labeling scheme for (h,W,a)-trees must have labels with worst-case length at least $\frac{h}{2}(\log W - \frac{h-1}{2}\log a) = \frac{1}{2}h\log W - \frac{1}{4}h^2\log a + \frac{1}{4}h\log a$. Since the number of nodes in such a tree is $n = 3\cdot 2^h - 2$, it follows that $h = \log(n+2) - \log 3$, and hence that $\log n - 2 \leq h \leq \log n$ for sufficiently large n. From this we see that the worst case label length is at least

$$\frac{1}{2}\log n\log W - \frac{1}{4}\log n(\log n - 1)\log a - \log W - \frac{1}{2}\log a.$$

In the case where $a=1$, we retrieve the bound of $\frac{1}{2}\log n\log W - \log W$ obtained in [38]. It seems that larger values of a only makes the above result weaker, but the the real strength of the above becomes apparent when we switch to the unweighted version of (h,W,a)-trees, in which we replace weighted edges by paths of similar lengths. Note that a distance labeling scheme for the family of unweighted (h,W,a)-trees can be used as a distance labeling scheme for the weighted (h,W,a)-trees, and hence any lower bound in the weighted version automatcially becomes a lower bound in the unweighted version.

The number of nodes n in an *unweighted* (h,W,a)-tree is upper bounded by

$$n \leq 2W + 2\cdot 2W/a + 2^2\cdot 2W/a^2 + \cdots + 2^{h-1}\cdot 2W/a^{h-1} + 1$$

In the case $a=2$, we get $n \leq 2Wh+1$.

▶ **Theorem 9.** *Any distance labeling scheme for unweighted $(h,W,2)$-trees, and hence also for general trees, has a worst-case label size of at least $\frac{1}{4}\log^2 n - O(\log n)$.*

Proof. Choose the largest integer h with $2\cdot 2^h h + 1 \leq n$, and note that we must have $h \geq \log n - O(\log\log n)$. Set $W = 2^h$ and consider the family of $(h,W,2)$-trees, which is a subfamily of the family of trees with n nodes. From Lemma 8 it therefore follows that the worst-case label length is

$$\frac{1}{2}h\log W - \frac{1}{4}h^2 + \frac{1}{4}h = \frac{1}{4}h^2 + \frac{1}{4}h = \frac{1}{4}\log^2 n + \frac{1}{4}\log n - O(\log\log n).$$ ◀

6 Approximate distances in trees

In this section we present a $(1+\varepsilon)$-stretch distance labeling schemes with labels of size $O(\log n)$.

▶ **Theorem 10.** *For constant $\varepsilon > 0$, $(1+\varepsilon)$ stretch distance labeling schemes for trees use $\Theta(\log n)$ bits.*

Proof. As in the case of exact distances, we will create labels of size $O(\log n)$ bits that contain the extended NCA labels from Lemma 5 as well as distroot. We will also be using the formula in (1). However we can not afford to store exact distance to each apex ancestor. Even directly storing an 2-approximate distance to each apex ancestor would require $\log n\log\log n$ bits. Instead we show how to compactly represent all the $(1+\varepsilon)$-approximate distances to

light ancestors for a node using a total of $O(\log n)$ bits, and show how to use this to obtain a $(1+2\varepsilon)$-approximation.

Let $w = \text{nca}(u,v)$ and assume $w \notin \{u,v\}$, since otherwise we can compute the exact distance using only distroot. Suppose we know a $(1+\varepsilon)$-approximation α of $\text{dist}(w,v)$ for some $\varepsilon \geq 0$. That is,

$$\text{dist}(w,v) \leq \alpha \leq (1+\varepsilon)\,\text{dist}(w,v). \tag{4}$$

Define $\tilde{d} = \text{distroot}(u) - \text{distroot}(v) + 2\alpha$. First we show that $\tilde{d}$ is a $(1+2\varepsilon)$-approximation of $\text{dist}(u,v)$. Next we show how to represent all the $(1+\varepsilon)$-approximate distances to light ancestors for a node using a total of $O(\log n)$ bits. Together with formula (1), these two facts prove that we can compute $(1+2\varepsilon)$-stretch distances between any pair of nodes with labels of size $O(\log n)$. To prove the theorem, we can then simple replace ε by $\frac{1}{2}\varepsilon$.

To see that $\tilde{d}$ is a $(1+2\varepsilon)$-approximation of $\text{dist}(u,v)$, first note that

$$\tilde{d} = \text{distroot}(u) - \text{distroot}(v) + 2\alpha \geq \text{distroot}(u) - \text{distroot}(v) + 2\,\text{dist}(w,v) = \text{dist}(u,v).$$

For the other inequality, note that

$$\begin{aligned}
\tilde{d} &= \text{distroot}(u) - \text{distroot}(v) + 2\alpha \\
&\leq \text{distroot}(u) - \text{distroot}(v) + 2(1+\epsilon)\,\text{dist}(w,v) \\
&= \text{distroot}(u) - (\text{distroot}(v) - \text{dist}(w,v)) + (1+2\epsilon)\,\text{dist}(w,v) \\
&= \text{distroot}(u) - \text{distroot}(w) + (1+2\epsilon)\,\text{dist}(w,v) \\
&= \text{dist}(u,w) + (1+2\epsilon)\,\text{dist}(w,v) \\
&\leq (1+2\epsilon)\,(\text{dist}(u,w) + \text{dist}(w,v)) \\
&= (1+2\epsilon)\,\text{dist}(u,v)\ .
\end{aligned}$$

It now only remains to show that we can compactly store all the approximate distances α to light ancestors using $O(\log n)$ bits space.

We use a heavy light path decomposition of the tree. For each node v we can save a 2 approximate distance to all its k proper light ancestors as follows. Let S be a binary string initially with k zeros. Before each 0 we now insert 1s such that, if we have j 1s in total from the beginning of S and to the i'th 0, then the distance to the ith light ancestor a of v satisfies that $2^{j-1} \leq \text{dist}(v,a) \leq 2^j$. This is the same as traversing the tree bottom-up from v and, for each light node encountered on the way, adding a 0 and each time the distance doubles adding a 1. The number of 0s equal the number of light nodes which is at most $\log n$, and the number of 1s is also limited by $\log n$ since n is the maximum distance in the tree. In total the length of S is at most $2\log n$.

Using the $O(\log n)$ bits label from Lemma 5 we can tell if one node is an ancestor of another, and if not which one has a light ancestor a that is a child of their nearest common ancestor w. In addition, we can determine the total number i of light ancestors up to a. This means that we can compute j, and hence the 2-approximation $j-1$, as the number of 1's in S until the i'th 0.

We have now obtained a 2-approximation with labels of size $O(\log n)$. We can improve this to a $(1+\varepsilon)$-approximation by setting a 1 in S each time the distance increases with $1+\varepsilon$ rather than 2. This will increase the label size with a constant factor $\frac{1}{\log(1+\varepsilon)}$.

This proves that there is a $(1+\varepsilon)$-stretch distance labeling scheme with $O(\log n)$ bit label length. To complete the proof of the theorem, we note that, given any $(1+\varepsilon)$-stretch distance scheme, we can always distinguish nodes (since identical nodes have distance 0), which means that we always need at least n different labels, and hence labels of size at least $\log n$ bits. ◀

References

1 S. Abiteboul, S. Alstrup, H. Kaplan, T. Milo, and T. Rauhe. Compact labeling scheme for ancestor queries. *SIAM J. Comput.*, 35(6):1295–1309, 2006. doi:10.1137/S0097539703437211.

2 S. Abiteboul, H. Kaplan, and T. Milo. Compact labeling schemes for ancestor queries. In *Proc. of the 12th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 547–556, 2001.

3 N. Alon, F .R. K. Chung, and R. L. Graham. Routing permutations on graphs via matchings. In *Proc. of the 25th Annual ACM Symp. on the Theory of Computing (STOC)*, pages 583–591, 1993.

4 S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. In *Proc. of the 14th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 689–698, 2003. URL: http://dl.acm.org/citation.cfm?id=644108.644220.

5 S. Alstrup, P. Bille, and T. Rauhe. Labeling schemes for small distances in trees. *SIAM J. Discrete Math.*, 19(2):448–462, 2005. See also SODA'03. doi:10.1137/S0895480103433409.

6 S. Alstrup, S. Dahlgaard, and M. B. T. Knudsen. Optimal induced universal graphs and labeling schemes for trees. In *Proc. 56th Annual Symp. on Foundations of Computer Science (FOCS)*, 2015.

7 S. Alstrup, C. Gavoile, E. B. Halvorsen, and H. Petersen. Simpler, faster and shorter labels for distances in graphs. In *Proc. of the 27th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2016.

8 S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe. Nearest common ancestors: A survey and a new algorithm for a distributed environment. *Theory of Computing Systems*, 37(3):441–456, 2004. doi:10.1007/s00224-004-1155-5.

9 S. Alstrup, E. B. Halvorsen, and K. G. Larsen. Near-optimal labeling schemes for nearest common ancestors. In *Proc. of the 25th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 972–982, 2014. doi:10.1137/1.9781611973402.72.

10 S. Alstrup, J. Holm, K. de Lichtenberg, and M. Thorup. Direct routing on trees. In *Proc. of the 9th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 342–349, 1998.

11 S. Alstrup, H. Kaplan, M. Thorup, and U. Zwick. Adjacency labeling schemes and induced-universal graphs. In *Proc. of the 47th Annual ACM Symp. on Theory of Computing (STOC)*, 2015.

12 S. Alstrup and T. Rauhe. Improved labeling schemes for ancestor queries. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2002.

13 S. Alstrup and T. Rauhe. Small induced-universal graphs and compact implicit graph representations. In *Proc. 43rd Annual Symp. on Foundations of Computer Science (FOCS)*, pages 53–62, 2002.

14 F. Bazzaro and C. Gavoille. Localized and compact data-structure for comparability graphs. *Discrete Mathematics*, 309(11):3465–3484, 2009. doi:10.1016/j.disc.2007.12.091.

15 N. Bonichon, C. Gavoille, and A. Labourel. Short labels by traversal and jumping. In *Structural Information and Communication Complexity*, pages 143–156. Springer, 2006. Include proof for binary trees and caterpillars.

16 M. A. Breuer. Coding the vertexes of a graph. *IEEE Trans. on Information Theory*, IT–12:148–153, 1966.

17 M. A. Breuer and J. Folkman. An unexpected result on coding vertices of a graph. *J. of Mathemathical analysis and applications*, 20:583–600, 1967.

18 V. D. Chepoi, F. F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs. In *24st Annual ACM*

Symp. on Computational Geometry (SoCG), pages 59–68, 2008. doi:10.1145/1377676.1377687.

19 V. D. Chepoi, F. F. Dragan, and Y. Vaxès. Distance and routing labeling schemes for non-positively curved plane graphs. *J. of Algorithms*, 61(2):60–88, 2006. doi:10.1016/j.jalgor.2004.07.011.

20 F. R. K. Chung. Universal graphs and induced-universal graphs. *J. of Graph Theory*, 14(4):443–454, 1990.

21 E. Cohen, H. Kaplan, and T. Milo. Labeling dynamic XML trees. *SIAM J. Comput.*, 39(5):2048–2074, 2010. doi:10.1137/070687633.

22 B. Courcelle and R. Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discrete Applied Mathematics*, 131:129–150, 2003. doi:10.1016/S0166-218X(02)00421-3.

23 L. J. Cowen. Compact routing with minimum stretch. *J. of Algorithms*, 38:170–183, 2001. See also SODA'91.

24 Y. Dodis, M. Pătraşcu, and M. Thorup. Changing base without losing space. In *Proc. of the 42nd Annual ACM Symp. on Theory of Computing (STOC)*, pages 593–602, 2010.

25 T. Eilam, C. Gavoille, and D. Peleg. Compact routing schemes with low stretch factor. *J. of Algorithms*, 46(2):97–114, 2003. doi:10.1016/S0196-6774(03)00002-6.

26 M. Elkin, A. Filtser, and O. Neiman. Prioritized metric structures and embedding. In *Proc. of the 47th Annual ACM Symp. on Theory of Computing (STOC)*, pages 489–498, 2015.

27 A. Farzan and J. I. Munro. Succinct encoding of arbitrary graphs. *Theoretical Computer Science*, 513:38–52, 2013. doi:10.1016/j.tcs.2013.09.031.

28 A. Farzan and J. I. Munro. A uniform paradigm to succinctly encode various families of trees. *Algorithmica*, 68(1):16–40, 2014. doi:10.1007/s00453-012-9664-0.

29 P. Fraigniaud and A. Korman. On randomized representations of graphs using short labels. In *Proc. of the 21st Annual Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 131–137, 2009. doi:10.1145/1583991.1584031.

30 P. Fraigniaud and A. Korman. Compact ancestry labeling schemes for XML trees. In *Proc. of the 21st annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 458–466, 2010.

31 P. Fraigniaud and A. Korman. An optimal ancestry scheme and small universal posets. In *Proc. of the 42nd Annual ACM Symp. on Theory of Computing (STOC)*, pages 611–620, 2010. doi:10.1145/1806689.1806773.

32 C. Gavoille, M. Katz, N. Katz, C. Paul, and D. Peleg. Approximate distance labeling schemes. In *Proc. of the 9th Annual European Symp. on Algorithms (ESA)*, pages 476–488, 2001.

33 C. Gavoille and A. Labourel. Distributed relationship schemes for trees. In *18th International Symp. on Algorithms and Computation (ISAAC)*, pages 728–738, 2007. doi:10.1007/978-3-540-77120-3_63.

34 C. Gavoille and A. Labourel. On local representation of distances in trees. In *Proc. of the 26th Annual ACM Symp. on Principles of Distributed Computing (PODC)*, pages 352–353, 2007. doi:10.1145/1281100.1281169.

35 C. Gavoille and O. Ly. Distance labeling in hyperbolic graphs. In *16th Annual International Symp. on Algorithms and Computation (ISAAC)*, pages 1071–1079, 2005. doi:10.1007/11602613_106.

36 C. Gavoille and C. Paul. Distance labeling scheme and split decomposition. *Discrete Mathematics*, 273(1-3):115–130, 2003.

37 C. Gavoille and C. Paul. Optimal distance labeling for interval graphs and related graphs families. *SIAM J. Discrete Math.*, 22(3):1239–1258, 2008. doi:10.1137/050635006.

38 C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. In *Proc. of the 12th Annual ACM-SIAM Symp. on Discrete algorithms (SODA)*, pages 210–219, 2001. URL: `http://dl.acm.org/citation.cfm?id=365411.365447`.

39 C. Gavoille, D. Peleg, S. Pérennes, and R. Raz. Distance labeling in graphs. *J. of Algorithms*, 53(1):85–112, 2004. See also SODA'01. `doi:10.1016/j.jalgor.2004.05.002`.

40 R. L. Graham and H. O. Pollak. On embedding graphs in squashed cubes. In *Lecture Notes in Mathematics*, volume 303. Springer-Verlag, 1972.

41 A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 534–543, 2003. `doi:10.1109/SFCS.2003.1238226`.

42 A. Gupta, A. Kumar, and R. Rastogi. Traveling with a pez dispenser (or, routing issues in mpls). *SIAM J. on Computing*, 34(2):453–474, 2005. See also FOCS'01.

43 E. B. Halvorsen. Labeling schemes for trees – overview and new results. Master's thesis, University of Copenhagen, 2013. Available at `http://esben.bistruphalvorsen.dk`.

44 S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. *SIAM J. Disc. Math.*, pages 596–603, 1992. See also STOC'88.

45 M. Kao, X. Li, and W. Wang. Average case analysis for tree labelling schemes. *Theor. Comput. Sci.*, 378(3):271–291, 2007. `doi:10.1016/j.tcs.2007.02.066`.

46 H. Kaplan and T. Milo. Short and simple labels for distances and other functions. In *7nd Work. on Algo. and Data Struc.*, 2001.

47 H. Kaplan, T. Milo, and R. Shabo. A comparison of labeling schemes for ancestor queries. In *Proc. of the 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2002.

48 M. Katz, N. A. Katz, A. Korman, and D. Peleg. Labeling schemes for flow and connectivity. *SIAM J. Comput.*, 34(1):23–40, 2004. See also SODA'02. `doi:10.1137/S0097539703433912`.

49 A. Korman. Labeling schemes for vertex connectivity. *ACM Trans. Algorithms*, 6(2):39:1–39:10, 2010. `doi:10.1145/1721837.1721855`.

50 A. Korman and S. Kutten. Labeling schemes with queries. *CoRR*, abs/cs/0609163, 2006. URL: `http://arxiv.org/abs/cs/0609163`.

51 A. Korman and S. Kutten. Labeling schemes with queries. In *SIROCCO*, pages 109–123, 2007.

52 A. Korman and D. Peleg. Labeling schemes for weighted dynamic trees. *Inf. Comput.*, 205(12):1721–1740, 2007. `doi:10.1016/j.ic.2007.08.004`.

53 R. Krauthgamer and J. R. Lee. Algorithms on negatively curved spaces. In *47th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 119–132, 2006. `doi:10.1109/FOCS.2006.9`.

54 F. T. Leighton. Methods for message routing in parallel machines. In *Proc. of the 24 Annual ACM Symp. on the Theory of Computing (STOC)*, pages 77–96, 1992.

55 M. Mitzenmacher. Constant time per edge is optimal on rooted tree networks. In *Proc. of the 8th Annual ACM Symp. on parallel algorithms and Architectures (SPAA'96)*, pages 162–169, 1996.

56 J. W. Moon. On minimal n-universal graphs. *Proc. of the Glasgow Mathematical Association*, 7(1):32–33, 1965.

57 J. H. Müller. *Local structure in graph classes*. PhD thesis, Georgia Institute of Technology, 1988.

58 J. I. Munro, R. Raman, V. Raman, and S. Srinivasa Rao. Succinct representations of permutations and functions. *Theor. Comput. Sci.*, 438:74–88, 2012. `doi:10.1016/j.tcs.2012.03.005`.

59 M. Pătraşcu. Succincter. In *Proc. 49th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 305–313, 2008.

60 D. Peleg. Informative labeling schemes for graphs. In *Proc. 25th Symp. on Mathematical Foundations of Computer Science*, pages 579–588, 2000.

61 D. Peleg. Proximity-preserving labeling schemes. *J. Graph Theory*, 33(3):167–176, 2000.

62 N. Santoro and R. Khatib. Labeling and implicit routing in networks. *The computer J.*, 28:5–8, 1985.

63 D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. of Computer and System Sciences*, 26(3):362–391, 1983. doi:10.1016/0022-0000(83)90006-5.

64 J. P. Spinrad. *Efficient Graph Representations*, volume 19 of *Fields Institute Monographs*. AMS, 2003.

65 K. Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proc. of the 36th Annual ACM Symp. on Theory of Computing (STOC)*, pages 281–290, 2004. doi:10.1145/1007352.1007399.

66 M. Tang, J. Yang, and G. Zhang. A compact distance labeling scheme for trees of small depths. In *International Conference on Scalable Computing and Communications / Eighth International Conference on Embedded Computing, ScalCom-EmbeddedCom*, pages 455–458, 2009. doi:10.1109/EmbeddedCom-ScalCom.2009.87.

67 M. Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004. See also FOCS'01. doi:10.1145/1039488.1039493.

68 M. Thorup and U. Zwick. Compact routing schemes. In *Proc. of the 13th Annual ACM Symp. on Parallel Algorithms and Architectures*, SPAA'01, pages 1–10, 2001. doi:10.1145/378580.378581.

69 M. Thorup and U. Zwick. Approximate distance oracles. *J. of the ACM*, 52(1):1–24, 2005. See also STOC'01.

70 O. Weimann and D. Peleg. A note on exact distance labeling. *Inf. Process. Lett.*, 111(14):671–673, 2011. doi:10.1016/j.ipl.2011.04.006.

71 Wikipedia. Implicit graph – wikipedia, the free encyclopedia, 2013. [Online; accessed 15-February-2014]. URL: http://en.wikipedia.org/w/index.php?title=Implicit_graph&oldid=585232203.

72 P. M. Winkler. Proof of the squashed cube conjecture. *Combinatorica*, 3(1):135–139, 1983. doi:10.1007/BF02579350.

73 L. Zhang. Optimal bounds for matching routing on trees. In *Proc. of the 8th Annual ACM-SIAM Symposium on discrete algorithms (SODA)*, pages 445–453, 1997.

Near Optimal Adjacency Labeling Schemes for Power-Law Graphs*

Casper Petersen[1], Noy Rotbart[2], Jakob Grue Simonsen†[3], and Christian Wulff-Nilsen[4]

1 **University of Copenhagen, Copenhagen, Denmark**
cazz@di.ku.dk
2 **University of Copenhagen, Copenhagen, Denmark**
noyro@di.ku.dk
3 **University of Copenhagen, Copenhagen, Denmark**
simonsen@di.ku.dk
4 **University of Copenhagen, Copenhagen, Denmark**
koolooz@di.ku.dk

Abstract

An adjacency labeling scheme labels the n nodes of a graph with bit strings in a way that allows, given the labels of two nodes, to determine adjacency based only on those bit strings. Though many graph families have been meticulously studied for this problem, a non-trivial labeling scheme for the important family of power-law graphs has yet to be obtained. This family is particularly useful for social and web networks as their underlying graphs are typically modelled as power-law graphs. Using simple strategies and a careful selection of a parameter, we show upper bounds for such labeling schemes of $\tilde{O}(\sqrt[\alpha]{n})$ for power law graphs with coefficient α, as well as nearly matching lower bounds. We also show two relaxations that allow for a label of logarithmic size, and extend the upper-bound technique to produce an improved distance labeling scheme for power-law graphs.

1998 ACM Subject Classification E.1 Distributed data structures

Keywords and phrases Labeling schemes, Power-law graphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.133

1 Introduction

A body of work on large, real-world networks deals with the difficulties of storing them and effectively resolving queries on them; examples of techniques are compression [14, 13] and dissemination of the underlying graphs of these networks over several machines [35, 52, 54]. A different approach to storing information about the graph is to disseminate the structural information of the graph to its vertices and store it locally. This *peer-to-peer* strategy allows inferring the graph's local topology using only local information stored in each vertex without using costly access to large, global data structures. In particular, it can be useful to address privacy concerns and ensure a high survivability rate [18].

We posit that a useful tool for such a peer-to-peer strategy is the notion of a *labeling scheme*: an algorithm that assigns a bit string–a *label*–to each vertex so that a query between

* The full version of this paper is available at http://arxiv.org/abs/1502.03971.
† Jakob Grue Simonsen partially supported by the Danish Council for Independent Research Sapere Aude grant "Complexity via Logic and Algebra" (COLA).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 133; pp. 133:1–133:15


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

any two vertices can be deduced solely from their respective labels. Labeling schemes are extremely well-studied in the algorithmic literature [8, 17, 20, 27, 32, 30, 37, 38, 40, 41, 49]; the main objective is to minimize the *maximum label size*: the maximum number of bits used in a label of any vertex. Among their applications are XML search engines [25], mapping services [1], and internet routing [43].

One class of graphs extensively used for modelling real-world networks is *power-law graphs*: roughly, n-vertex graphs where the number of vertices of degree k is proportional to n/k^α for some positive α. Power-law graphs (also called scale-free graphs in the literature) have been used to model the Internet AS-level graph [50, 4], and many other types of network (see, e.g., [24, 47] for overviews). The adequacy of fit of power-law graph models to actual data, as well as the empirical correctness of the conjectured mechanisms giving rise to power-law behaviour, have been subject to criticism (see, e.g., [2, 24]). In spite of such criticism, and because their degree distribution affords a reasonable approximation of the degree distribution of many networks, the class of power-law graphs remains a popular tool in network modelling. In this paper, we perform the first theoretical and practical study of adjacency labeling schemes for classes of graphs whose statistical properties–in particular their *degree distribution*–more closely resemble that of real-world networks.

1.1 Our contributions

Our contributions are:

A discrete and simple characterisation of power-law graphs

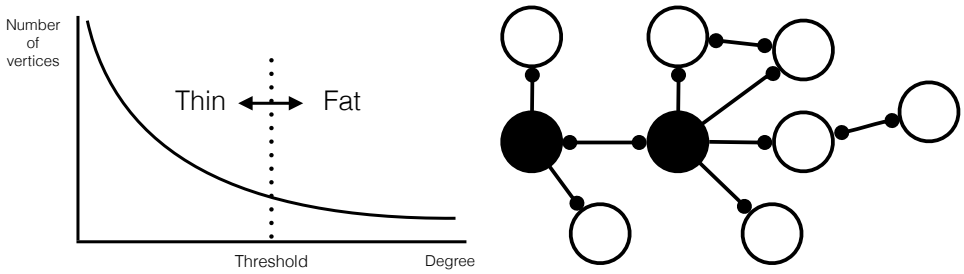
An n-vertex graph is power-law if the number of its vertices of degree k is proportional to n/k^α for some positive α. To solidify this somewhat vague definition, numerous probabilistic and deterministic definitions of power-law graphs are given in the literature. In Sec. 3, we define and prove useful properties for two simple families of graphs, $\mathcal{P}_h$ and $\mathcal{P}_l$, where $\mathcal{P}_h$ contains and $\mathcal{P}_l$ is contained by the standard definitions of power-law graphs in the literature, including recent ones [16]. We use $\mathcal{P}_h$ and $\mathcal{P}_l$ to study upper and lower bounds respectively.

An $O(\sqrt[\alpha]{n}(\log n)^{1-1/\alpha})$ adjacency labeling scheme

In Sec. 4, we describe our labeling scheme, which is based on two ideas: (i) a labeling *strategy* that partitions the vertices of G into high ("fat") and low degree ("thin") vertices based on a threshold degree, and (ii) a threshold *prediction* that depends only on the coefficient α of a power-law curve fitted to the degree distribution of G. These ideas are illustrated in Figure 1. Using the same ideas, we get an asymptotically near-tight $O(\sqrt{n \log n})$ adjacency labeling scheme for sparse graphs. As real-world power-law graphs have $2 \leq \alpha \leq 3$ and rarely exceed 10^{10} vertices, this implies labels of the order of $10^4 - 10^5$ bits. That, and the simplicity of our labeling scheme suggests that our labeling schemes may be appealing in practice. To stress this point, we offer an experimental evaluation of our labeling scheme in the full version of the paper.

A lower bound of $\Omega(\sqrt[\alpha]{n})$ for any adjacency labeling scheme

In Sec. 5, We use our restrictive subclass of power-law graphs and show that it requires label size $\Omega(\sqrt[\alpha]{n})$ for n-vertex graphs. This lower bound shows that our upper bound above is asymptotically optimal, bar a $(\log n)^{1-1/\alpha}$ factor. By the connections between adjacency labeling schemes and universal graphs, we also obtain upper and lower bounds for induced



(a) demonstrates the threshold assignment

(b) demonstrates the label assignment, in which fat (black) nodes do not store adjacency to thin (white) nodes

Figure 1 Two illustrations of the main idea.

universal graphs for power-law graphs. We also show, in Sec. 6, two scenarios in which this lower bound can be bypassed.

A $o(n)$ distance labeling scheme

In Sec. 7, we demonstrate the usefulness of our strategy to arrive at a $o(n)$ distance labeling scheme for power-law graphs. Our labeling scheme is designed to outperform competing labeling schemes for small distances, in accordance with Chung and Lu's findings [22] on the small expected diameter of power-law graphs.

1.2 Related work

Adjacency labeling schemes for key graph families are by now well understood. General graphs require a label size of $n/2 + O(1)$ [48, 8], while trees, planar graphs, and bounded degree graphs enjoy labels of logarithmic size [9, 30, 3]. Adjacency labeling schemes are also tightly coupled with the graph-theoretic concept of induced universal graphs, in which one aims to find the smallest N where there exist a graph of N vertices which contains all graphs of a particular graph family $\mathcal{F}_n$ of n vertices as induced subgraphs. It was shown [36] that an $f(n) \log n$ adjacency labeling scheme for $\mathcal{F}_n$ constructs an induced universal graph for this family of $2^{f(n)}$ vertices. In the context of sparse graphs, a body of work on universal graphs[1] for this family was investigated both by Babai et al. [11] and by Alon and Asodi [5].

Routing labeling schemes for power-law graphs have been investigated by Brady and Cowen [17], and by Chen et al. [21]. Labeling schemes for properties other than adjacency have been investigated for various classes of graphs, e.g., distance [32], and flow [37]. Dynamic labeling schemes were studied by Korman and Peleg [38, 40, 41] and recently by Dahlgaard et. al [27]. Experimental evaluations for some labeling schemes for various properties on general graphs have been performed by Caminiti et. al [20], Fischer [29] and Rotbart et. al [49].

In the context of distributed graph computing systems, a somewhat related paradigm of computation is the vertex centric computing model "think like a vertex". In this model each vertex exchanges messages only with nearby vertices, to improve locality and simplify the design and implementation of such systems. Among the numerous systems proposed are Pregel [45], Power-Graph [35] and GraphLab [44]. For a recent survey on the topic see [46].

[1] A graph that contains each graph from the graph family as a subgraph, not necessarily induced.

2 Preliminaries

Throughout the paper we consider n-vertex, undirected graphs. For a real $c > 0$, a graph is *c-sparse* if it has at most cn edges and *sparse* if it is c-sparse for some constant c. For $0 < c \leq n-1$, the set of c-sparse graphs with n vertices is denoted by $\mathcal{S}_{c,n}$. If $\mathcal{F}$ is a set of graphs, $\mathcal{F}_n$ denotes the subset of graphs in $\mathcal{F}$ having exactly n vertices. The degree of a vertex v in a graph is denoted by $\Delta(v)$, and for non-negative integers k, the set of vertices in a graph G of degree k is denoted by V_k. The length of a binary string $x \in \{0,1\}^*$ is denoted by $|x|$.

Let $\mathcal{F}$ be a set of graphs. An *adjacency labeling scheme* (from hereon just *labeling scheme*) for $\mathcal{F}$ is a pair consisting of an *encoder* and a *decoder*. The encoder is an algorithm that receives $G \in \mathcal{F}$ as input and outputs a bit string $\mathcal{L}(v) \in \{0,1\}^*$, called the *label* of v, for each vertex v in G. The decoder is an algorithm that receives any two labels $\mathcal{L}(v), \mathcal{L}(u)$ as input and outputs `true` if u and v are adjacent in G and `false` otherwise. Note that the graph G is not an input to the decoder. The *size* of a labeling scheme is the map $\text{size} : \mathbb{N} \longrightarrow \mathbb{N}$ such that $\text{size}(n)$ is the maximum length of any label assigned by the encoder to any vertex in any graph $G \in \mathcal{F}_n$. The *degree distribution* of a graph $G = (V, E)$ is the mapping $\text{ddist}_G(k) : \mathbb{N}_0 \longrightarrow \mathbb{Q}$ defined by $\text{ddist}_G(k) := \frac{|V_k|}{n}$.

3 Defining Power-Law Graphs

In the literature *power-law* graphs are usually defined as the class of n vertex graphs G such that $\text{ddist}_G(k)$ is proportional to $k^{-\alpha}$ for some real number $\alpha > 1$. Ideally, and ignoring rounding, $\text{ddist}_G(k) = Ck^{-\alpha}$ for all k for constant C. As the degree distribution of a graph must be a probability distribution, we have $\sum_{k=1}^{\infty} Ck^{-\alpha} = C\sum_{k=1}^{\infty} k^{-\alpha} = 1$, hence $C = 1/\zeta(\alpha)$ where ζ is the Riemann zeta function. However, in the literature, concessions are usually made that relax the restrictions on $\text{ddist}_G(k)$, for example that the power-law property need only hold for high-degree vertices ("above a cutoff"), or that $\text{ddist}_G(k)$ is only *approximately* equal to $Ck^{-\alpha}$, with some approximation error that falls off with n. To ensure that our results hold for all these variations of power-law graphs, we define two families of graphs $\mathcal{P}_h$ and $\mathcal{P}_l$ with $\mathcal{P}_l \subsetneq \mathcal{P}_h$. Family $\mathcal{P}_h$ is rich enough to contain the graphs whose degree distribution is approximately, or perfectly, power-law distributed, and our upper bound on the label size for our labeling scheme holds for any graph in $\mathcal{P}_h$. Family $\mathcal{P}_l$ is used to show our lower bound and is restrictive enough that most definitions of power-law graph occurring in the literature will contain it.

In the following, let $i_1 = \Theta(\sqrt[\alpha]{n})$ be the smallest integer such that $\lfloor Cn/i_1^{\alpha} \rfloor \leq 1$, and let $C' \geq (\frac{C}{\alpha-1} + \frac{i_1}{\sqrt[\alpha]{n}} + 5)^{\alpha} + \frac{C}{\alpha-1}$ be a constant; we shall use C' in the remainder of the paper.

▶ **Definition 1.** Let $\alpha > 1$ be a real number and let $\chi : \mathbb{N} \to \mathbb{N}$ be a function. $\mathcal{P}_{h,\chi,\alpha}$ is the family of graphs G such that if $n = |V(G)|$ then for all integers k between $\chi(n)$ and $n-1$, $\sum_{i=k}^{n-1} |V_i| \leq C'(\frac{n}{k^{\alpha-1}})$. We shall usually suppress χ and α, writing merely $\mathcal{P}_h$.

The function χ captures the notion of a cutoff as defined in [24, Sec. 3.1]; the intuition is that for an n-vertex graph the power-law distribution need only apply for nodes of degree higher than $\chi(n)$, rather than for all degrees. Setting $\chi(n) = 1$ corresponds to the case where the entire range of degrees follows a power-law distribution, hence even for small values of $\chi(n)$, $\mathcal{P}_h$ morally contains all graphs with power-law degree distribution. We will later prove upper bounds that hold for *all* χ bounded from above by some function; in particular for the upper bound for adjacency labeling schemes, the bound holds for $\chi(n)$ as high as $\sqrt[\alpha]{n/\log n}$.

The class $\mathcal{P}_l$ contains graphs where the number of vertices of degree k must be $C\frac{n}{k^\alpha}$ rounded either up or down and the number of vertices of degree k is non-increasing with k. Note that the function $k \mapsto C\frac{1}{k^\alpha}$ is strictly decreasing.

▶ **Definition 2.** Let $\alpha > 1$ be a real number and let $C = 1/\zeta(\alpha)$ where ζ is the Riemann zeta function. $\mathcal{P}_{l,\alpha}$ is the set of graphs $G = (V, E)$ such that

1. $\lfloor Cn \rfloor - i_1 - 1 \leq |V_1| \leq \lceil Cn \rceil$,
2. $\lfloor C\frac{n}{2^\alpha} \rfloor \leq |V_2| \leq \lceil C\frac{n}{2^\alpha} \rceil + 1$,
3. for every i with $3 \leq i \leq n$: $|V_i| \in \{\lfloor C\frac{n}{i^\alpha} \rfloor, \lceil C\frac{n}{i^\alpha} \rceil\}$, and
4. for every i with $2 \leq i \leq n-1$: $|V_i| \geq |V_{i+1}|$.

We usually suppress α, writing just $\mathcal{P}_l$.

Note that we allow slightly more noise in the sizes of V_1 and V_2 than in the remaining sets; without it, it seems tricky to prove a better lower bound than $\Omega(\sqrt[\alpha+1]{n})$ on label sizes.

We show the following properties of $\mathcal{P}_l$.

▶ **Proposition 1.** *The maximum degree in an n-vertex graph in $\mathcal{P}_l$ is at most $\left(\frac{C}{\alpha-1} + 2\right)\sqrt[\alpha]{n} + i_1 + 3 = \Theta(\sqrt[\alpha]{n})$.*

Proof. Let $n > 0$ be an integer and let $k' = \lfloor \sqrt[\alpha]{n} \rfloor$. Furthermore, let $S_{k'} = \sum_{i=1}^{k'} |V_i|$, that is $S_{k'}$ is the number of vertices of degree at most k'. Let $S^-_{k'} = (\sum_{i=1}^{k'} \lfloor Cni^{-\alpha} \rfloor) - i_1 - 1$. Then $S_{k'} \geq S^-_{k'}$. We now bound $S^-_{k'}$ from below. For every i with $1 \leq i \leq k'$,

$$\begin{aligned}
S^-_{k'} + k' &= -i_1 - 1 + \sum_{i=1}^{k'} \left(\lfloor Cni^{-\alpha} \rfloor + 1 \right) \geq -i_1 - 1 + \sum_{i=1}^{k'} Cni^{-\alpha} = -i_1 - 1 + Cn \sum_{i=1}^{k'} i^{-\alpha} \\
&\geq n \left(1 - C \sum_{i=k'+1}^{\infty} i^{-\alpha} \right) - i_1 - 1 \geq n \left(1 - C \int_{k'}^{\infty} x^{-\alpha} dx \right) - i_1 - 1 \\
&= n \left(1 - C \left[\frac{1}{\alpha - 1} x^{-\alpha+1} \right]_{\infty}^{k'} \right) - i_1 - 1 = n \left(1 - \frac{C}{\alpha - 1} \left(\lceil \sqrt[\alpha]{n} \rceil \right)^{-\alpha+1} \right) - i_1 - 1 \\
&\geq n \left(1 - \frac{C}{\alpha - 1} \left(\sqrt[\alpha]{n} \right)^{-\alpha+1} \right) - i_1 - 1 = n - \frac{Cn}{\alpha - 1} n^{-1+\frac{1}{\alpha}} - i_1 - 1 \\
&= n - \frac{C}{\alpha - 1} \sqrt[\alpha]{n} - i_1 - 1,
\end{aligned}$$

giving $S_{k'} \geq S^-_{k'} \geq n - \frac{C}{\alpha-1}\sqrt[\alpha]{n} - \lceil \sqrt[\alpha]{n} \rceil - i_1 - 1$. There are thus at most $\frac{C}{\alpha-1}\sqrt[\alpha]{n} + \lfloor \sqrt[\alpha]{n} \rfloor + i_1 + 1$ vertices of degree strictly more than $k' = \lceil \sqrt[\alpha]{n} \rceil$. Since for every $1 \leq i \leq n-1$: $|V_i| \geq |V_{i+1}|$, it follows that the maximum degree of any graph in $\mathcal{P}_l$ is at most $\left(\frac{C}{\alpha-1} + 2\right)\sqrt[\alpha]{n} + i_1 + 3$. ◀

▶ **Proposition 2.** *For $\alpha > 2$, all graphs in $\mathcal{P}_l$ are sparse.*

Proof. By Proposition 1, the maximum degree of an n-vertex graph in $\mathcal{P}_l$ graph is at most $k' \triangleq \left(\frac{C}{\alpha-1} + 2\right)\sqrt[\alpha]{n} + i_1 + 3$, whence the total number of edges is at most $\frac{1}{2}\sum_{k=1}^{k'} k|V_k|$. By definition, $|V_k| \leq \lceil \frac{Cn}{k^\alpha} \rceil \leq \frac{Cn}{k^\alpha} + 1$ for $k \neq 2$ and $|V_2| \leq \lceil \frac{Cn}{2^\alpha} \rceil + 1$, and thus

$$\begin{aligned}
\frac{1}{2} \sum_{k=1}^{k'} k|V_k| &\leq 1 + \frac{1}{2} \sum_{k=1}^{k'} k \left(\frac{Cn}{k^\alpha} + 1 \right) \leq 1 + \frac{k'(k'+1)}{4} + Cn \sum_{k=1}^{\infty} k^{-\alpha+1} \\
&= O(n^{2/\alpha}) + Cn\zeta(\alpha - 1) = O(n).
\end{aligned}$$

◀

▶ **Proposition 3.** *For any χ and $\alpha > 1$, $\mathcal{P}_{l,\alpha} \subseteq \mathcal{P}_{h,\chi,\alpha}$.*

Proof. Let $d = \lfloor (\frac{C}{\alpha-1} + 2)\sqrt[\alpha]{n} + i_1 + 3 \rfloor$. For any graph in $\mathcal{P}_l$ with n vertices and for any k, $|V_k| \leq Ck^{-\alpha}n + 1$ and by Proposition 1, $|V_k| = 0$ when $k > d$.

Let k be an arbitrary integer between $\chi(n)$ and $n-1$. We need to show that $\sum_{i=k}^{n-1} |V_i| \leq C'(\frac{n}{k^{\alpha-1}})$. It suffices to show this for $k \leq d$. We have:

$$\begin{aligned}
\sum_{i=k}^{n-1} |V_i| &\leq \sum_{i=k}^{d} (Ci^{-\alpha}n + 1) = d - k + 1 + Cn \sum_{i=k}^{d} i^{-\alpha} \\
&\leq \left(\frac{C}{\alpha - 1} + \frac{i_1}{\sqrt[\alpha]{n}} + 5 \right) \sqrt[\alpha]{n} + Cn \int_k^d x^{-\alpha} dx \\
&\leq \left(\frac{C}{\alpha - 1} + \frac{i_1}{\sqrt[\alpha]{n}} + 5 \right) \sqrt[\alpha]{n} + Cn \left[\frac{1}{\alpha - 1} x^{-\alpha+1} \right]_{\infty}^{k} \\
&\leq \left(\left(\frac{C}{\alpha - 1} + \frac{i_1}{\sqrt[\alpha]{n}} + 5 \right) \left(\frac{\sqrt[\alpha]{n} d^{\alpha-1}}{n} \right) + \frac{C}{\alpha - 1} \right) nk^{-\alpha+1} \\
&\leq \left(\frac{C}{\alpha - 1} + \frac{i_1}{\sqrt[\alpha]{n}} + 5 \right) \left(\frac{C}{\alpha - 1} + \frac{i_1}{\sqrt[\alpha]{n}} + 5 \right)^{\alpha-1} nk^{-\alpha+1} + \left(\frac{C}{\alpha - 1} \right) nk^{-\alpha+1} \\
&\leq C' nk^{-\alpha+1},
\end{aligned}$$

as desired. ◀

3.1 Comparison to other deterministic models

Numerous probabilistic and deterministic definitions of power-law graphs are given in the literature. A recent deterministic model, called shifted power-law distribution [28] has recently proven to capture a vast number of such definitions, both in theory and experimentally in [16]. We show that our definition of $\mathcal{P}_h$ contains graphs that adhere to the model, which is defined as follows. Let $c_1 > 0$ be a constant. A graph G is *power-law bounded* for parameters $\alpha > 1$ and $t \geq 0$ if for every integer $d \geq 0$, the number of vertices of G of degree in $[2^d, 2^{d+1})$ is at most

$$c_1 n (t+1)^{\alpha-1} \sum_{i=2^d}^{2^{d+1}-1} (i+t)^{-\alpha}.$$

As experimentally verified in [16], the value of t is typically very small. If $t = O(1)$, the bound above becomes $O(n \sum_{i=2^d}^{2^{d+1}-1} i^{-\alpha})$. In this case, our family $\mathcal{P}_h(\chi, \alpha)$ is rich enough to contain these power-law bounded graphs for sufficiently large C' and any choice of χ and α. This follows since for any power-law bounded graph with n vertices and any integer k between 1 and $n-1$, $\sum_{i=k}^{n-1} |V_i| = O(\sum_{d=\lfloor \lg k \rfloor}^{\lceil \lg(n-1) \rceil} n \sum_{i=2^d}^{2^{d+1}-1} i^{-\alpha}) = O(\frac{n}{k^{\alpha-1}})$. Thus our upper bound also applies to power-law bounded graphs. It is possible to extend our upper bound to super-constant t where the bound is stronger the smaller t is; we omit the details. Conversely, our family $\mathcal{P}_l$ is restrictive enough that $\mathcal{P}_l$ is contained in the family of power-law bounded graphs when $t = O(1)$, and the lower bound we derive thus also holds in that setting.

4 The Labeling Schemes

We now construct algorithms for labeling schemes for c-sparse graphs and for the family $\mathcal{P}_h$. Both labeling schemes partition vertices into *thin* vertices which are of low degree and *fat*

vertices of high degree. The *degree threshold* for the scheme is the lowest possible degree of a fat vertex. We start with c-sparse graphs.

▶ **Theorem 3.** *There is a* $\sqrt{2cn\log n} + 2\log n + 1$ *labeling scheme for* $\mathcal{S}_{c,n}$.

Proof. Let $G = (V, E)$ be an n-vertex c-sparse graph. Let $\tau(n)$ be the degree threshold for n-vertex graphs; we choose $\tau(n)$ below. Let k denote the number of fat vertices of G, and assign each fat vertex a unique identifier between 1 and k. Each thin vertex is given a unique identifier between $k+1$ and n.

For a $v \in V$, the first part of the label $\mathcal{L}(v)$ is a single bit indicating whether v is thin or fat followed by a string of $\log n$ bits representing its identifier. If v is thin, the last part of $\mathcal{L}(v)$ is the concatenation of the identifiers of the neighbors of v. If v is fat, the last part of $\mathcal{L}(v)$ is a *fat bit string* of length k where the ith bit is 1 iff v is incident to the (fat) vertex with identifier i.

Decoding a pair $(\mathcal{L}(u), \mathcal{L}(v))$ is straightforward: if one of the vertices, say u, is thin, u and v are adjacent iff the identifier of v is part of the label of u. If both u and v are fat then they are adjacent iff the ith bit of the fat bit string of $\mathcal{L}(u)$ is 1 where i is the identifier of v. Both decoding processes can be computed in $O(\log n)$ time using standard assumptions.

Since $|E| \leq cn$, we have $k \leq 2cn/\tau(n)$. A fat vertex thus has label size $1 + \log n + k \leq 1 + \log n + 2cn/\tau(n)$ and a thin vertex has label size at most $1 + \log n + \tau(n)\log n$. To minimize the maximum possible label size, we solve $2cn/x = x\log n$. Solving this gives $x = \sqrt{2cn/\log n}$ and setting $\tau(n) = \lceil x \rceil$ gives a label size of at most $1 + \log n + (\sqrt{2cn/\log n} + 1)\log n \leq 1 + 2\log n + \sqrt{2cn\log n}$. ◀

By Proposition 2, graphs in $\mathcal{P}_l$ are sparse for $\alpha > 2$. This gives a label size of $O(\sqrt{n\log n})$ with the labeling scheme in Theorem 3. We now show that this label can be significantly improved, by constructing a labeling scheme for $\mathcal{P}_h$ which contains $\mathcal{P}_l$.

▶ **Theorem 4.** *There is a* $\sqrt[\alpha]{C'n}(\log n)^{1-1/\alpha} + 2\log n + 1$ *labeling scheme for* $\mathcal{P}_h$.

Proof. The proof is very similar to that of Theorem 3. We let $\tau(n)$ denote the degree threshold. If we pick $\tau(n) \geq \sqrt[\alpha]{n/\log n}$ then by Definition 1 there are at most $C'n/\tau(n)^{\alpha-1}$ fat vertices. Defining labels in the same way as in Theorem 3 gives a label size for thin vertices of at most $1 + \log n + \tau(n)\log n$ and a label size for fat vertices of at most $1 + \log n + C'n/\tau(n)^{\alpha-1}$. We minimize by solving $x\log n = C'n/x^{\alpha-1}$, giving $x = \sqrt[\alpha]{C'n/\log n}$. Setting $\tau(n) = \lceil x \rceil$ gives a label size of at most $\sqrt[\alpha]{C'n}(\log n)^{1-1/\alpha} + 2\log n + 1$. ◀

4.1 A labeling scheme for random graphs

There are schemes using randomness to "grow" graphs that, with high probability, have an approximate power-law degree distribution for a range of degrees (see e.g. [23]). For graphs obtained from such models, their degree sequences are instead probability distributions. We now show that applying our labeling scheme for $\mathcal{P}_h$ to random graphs with the power-law distribution results in a small expected worst-case label size.

Using the definition of Mitzenmacher [47], a random variable X is said to have the *power-law* distribution (w.r.t. $\alpha > 1$) if

$$Pr[X \geq x] \sim cx^{-\alpha+1},$$

for a constant $c > 0$, i.e., $\lim_{x\to\infty} Pr[X \geq x]/cx^{-\alpha+1} = 1$.

Let $\epsilon > 0$ be fixed. Consider a graph G picked from a family $\mathcal{F}$ of random graphs whose degree sequences have the power-law distribution. Order the vertices of G arbitrarily

as $v_1, \ldots, v_n$. For $i = 1, \ldots, n$, let indicator variable X_i be 1 iff v_i has degree at least $d = \sqrt[\alpha]{n/\log n}$. There is a constant $N_0 \in \mathbb{N}$ (depending on ϵ) such that if $n \geq N_0$ then for all i,

$$E[X_i] = \Pr[X_i = 1] \leq (1+\epsilon)cd^{-\alpha+1}.$$

With the same labeling scheme as for $\mathcal{P}_h$ with degree threshold $\tau(n) = d$, denote by E_n the expected label size of an n-vertex graph from $\mathcal{F}$. Then for all $n \geq N_0$,

$$\begin{aligned} E_n &= \sum_{x=0}^{n} \Pr\left[\sum_{i=1}^{n} X_i = x\right] O((x + d\log n)) = O\left(d\log n + E\left[\sum_{i=1}^{n} X_i\right]\right) \\ &= O\left(d\log n + \sum_{i=1}^{n} E[X_i]\right) = O\left(d\log n + nd^{-\alpha+1}\right) = O\left(\sqrt[\alpha]{n}(\log n)^{1-1/\alpha}\right). \end{aligned}$$

Thus, we have:

▶ **Theorem 5.** *Let $\mathcal{F}$ be a family of graphs with degree sequences having the power-law distribution w.r.t. $\alpha > 1$. Then there is a labeling scheme for $\mathcal{F}$ such that the expected worst-case label size of any graph $G \in \mathcal{F}$ is $O(\sqrt[\alpha]{n}(\log n)^{1-1/\alpha})$ where n is the number of vertices of G.*

5 Lower Bounds

We now derive lower bounds for the label size of any labeling schemes for both $\mathcal{S}_{c,n}$ and $\mathcal{P}_l$. Our proofs rely on Moon's [48] lower bound of $\lfloor n/2 \rfloor$ bits for labeling scheme for general graphs. We first show that the upper bound achieved for sparse graphs is close to the best possible. The following proposition is essentially a more precise version of the lower bound suggested by Spinrad [51].

▶ **Proposition 4.** *Any labeling scheme for $\mathcal{S}_{c,n}$ requires labels of size at least $\left\lfloor \frac{\sqrt{cn}}{2} \right\rfloor$ bits.*

Proof. Assume for contradiction that there exists a labeling scheme assigning labels of size strictly less than $\lfloor \frac{\sqrt{cn}}{2} \rfloor$. Let G be an n-vertex graph. Let G' be the graph resulting by adding $\left\lfloor \frac{n^2}{c} \right\rfloor - n$ isolated vertices to G, and note that now G' is c-sparse. The graph G is an induced subgraph of G'. It now follows that the vertices of G have labels of size strictly less than $\left\lfloor \frac{\sqrt{c\lfloor n^2/c \rfloor}}{2} \right\rfloor \leq n/2$ bits. As G was arbitrary, we obtain a contradiction. ◀

In the remainder of this section we are assuming that $\alpha > 2$ and prove the following:

▶ **Theorem 6.** *For any n, any labeling scheme for n-vertex graphs of $\mathcal{P}_{h,\chi,\alpha}$ requires label size $\Omega(\sqrt[\alpha]{n})$.*

More precisely, we present a lower bound for $\mathcal{P}_l$ which is contained in $\mathcal{P}_h$. Let $n \in \mathbb{N}$ be given and let $H = (V(H), E(H))$ be an arbitrary graph with i_1 vertices where $i_1 = \Theta(\sqrt[\alpha]{n})$ is defined as in Section 3. We show how to construct a graph $G = (V, E)$ in $\mathcal{P}_l$ with n vertices that contains H as an induced subgraph. Observe that a labeling of G induces a labeling of H. As H was chosen arbitrarily and as any labeling scheme for k-vertex graphs requires $\lfloor i_1/2 \rfloor$ label size in the worst case, Theorem 6 follows if we can show the existence of G.

We construct G incrementally where initially $E = \emptyset$. Partition V into subsets $V_1, \ldots, V_n$ as follows. The set V_1 has size $\lfloor Cn \rfloor - i_1$. For $i = 2, \ldots, i_1 - 1$, V_i has size $\lfloor Cn/i^\alpha \rfloor$. Letting

$n' = \sum_{i=1}^{i_1-1} |V_i|$, we set the size of V_i to 1 for $i = i_1, \ldots, i_1 + n - n' - 1$ and the size of V_i to 0 for $i = i_1 + n - n', \ldots, n$, thereby ensuring that the sum of sizes of all sets is n. Observe that $\sum_{i=1}^{i_1} \lfloor Cn/i^\alpha \rfloor \leq n$ so that $n' \leq n - i_1$, implying that $n - n' \geq i_1$. Hence we have at least i_1 size 1 subsets $V_{i_1}, \ldots, V_{i_1+n-n'-1}$ in each of which the vertex degree allowed by Definition 2 is at least i_1.

Let $v_1, \ldots, v_{i_1}$ be an ordering of $V(H)$, form a set $V_H \subseteq V$ of i_1 arbitrary vertices from the sets $V_{i_1}, \ldots, V_{i_1+n-n'-1}$, and choose an ordering $v'_1, \ldots, v'_{i_1}$ of V_H. For all $i, j \in \{1, \ldots, i_1\}$, add edge (v'_i, v'_j) to E iff $(v_i, v_j) \in E(H)$. Now, H is an induced subgraph of G and since the maximum degree of H is $i_1 - 1$, no vertex of V_i exceeds the degree bound allowed by Definition 2 for $i = 1, \ldots, n$.

We next add additional edges to G in three phases to ensure that it is an element of $\mathcal{P}_l$ while maintaining the property that H is an induced subgraph of G. For $i = 1, \ldots, n$, during the construction of G we say that a vertex $v \in V_i$ is *unprocessed* if its degree in the current graph G is strictly less than i. If the degree of v is exactly i, v is *processed*.

Phase 1

Let $V' = V \setminus (V_1 \cup V_H)$. Phase 1 is as follows: while there exists a pair of unprocessed vertices $(u, v) \in V' \times V_H$, add (u, v) to E.

When Phase 1 terminates, H is clearly still an induced subgraph of G. Furthermore, all vertices of V_H are processed. To see this, note that the sum of degrees of vertices of V_H when they are all processed is $O(i_1^2) = O(n^{2/\alpha})$ which is $o(n)$ since $\alpha > 2$. Furthermore, prior to Phase 1, each of the $\Theta(n)$ vertices of V' have degree 0 and can thus have their degrees increased by at least 1 before being processed.

Phase 2

While there exists a pair of unprocessed vertices $(u, v) \in V' \times V'$, add (u, v) to E. At termination, at most one vertex of V' remains unprocessed. If such a vertex exists we process it by connecting it to $O(\sqrt[c]{n})$ vertices of V_1; as $|V_1| = \Theta(n)$ there are enough vertices of V_1 to accomodate this. Furthermore, prior to adding these edges, all vertices of V_1 have degree 0, and hence the bound allowed for vertices of this set is not exceeded.

Phase 3

We add edges between pairs of unprocessed vertices of V_1 until no such pair exists. If no unprocessed vertices remain we have the desired graph G. Otherwise, let $w \in V_1$ be the unprocessed vertex of degree 0. We add a single edge from w to another vertex w' of V_1, thereby processing w and moving w' from V_1 to V_2. Note that the sizes of V_1 and V_2 are kept in their allowed ranges due to the first two conditions in Definition 2. This proves Theorem 6.

6 $O(\log n)$ adjacency labeling schemes for some power-law graphs

The lower bound presented can be avoided in two interesting cases. The first, for random graphs generated by a popular model, and the second using an extension of the concept of labeling schemes from the literature.

BA model

As discussed in Sec. 4.1, generative models play an important role in the study of power-law graphs. Perhaps the most well-known generative model is the Barabási-Albert (BA) which, roughly, grows a graph in a sequence of time steps by inserting a single vertex at each step and attaching it to m existing vertices with probability weighted by the degree of each existing vertex [12]. The BA model generates graphs that asymptotically have a power-law degree distribution ($\alpha = 3$) for low-degree nodes [15]. However, graphs created by the BA model have low arboricity [2] [34]. We use this fact to devise the following highly efficient labeling scheme for such graphs.

▶ **Proposition 5.** *The family of graphs generated by the BA model has an $O(m \log n)$ adjacency labeling scheme.*

Proof. Let $G = (V, E)$ be an n-vertex graph resulting by the construction by the BA model with some parameter m (starting from some graph $G_0 = (V_0, E_0)$ with $|V_0| \ll n$). While it is not known how to compute the arboricity of a graph efficiently, it is possible in near-linear time to compute a partition of G with at most twice[3] the number of forests in comparison to the optimal [10]. We can thus decompose the graph to $2m$ forests in near linear time and label each forest using the recent $\log n + O(1)$ labeling scheme for trees [6], and achieve a $2m(\log n + O(1))$ labeling scheme for G. ◀

If the encoder operates at the same time as the creation of the graph, Proposition 5 can be tightened to yield a $m \log n$ labeling scheme, by storing the identifiers of the vertices to the node introduced. Theorem 6 and Proposition 5 strongly suggest that local properties of power-law graphs are very different from those of a randomly generated graph using the BA model. In contrast, other generative models such as Waxman's [53], N-level Hierarchical [19]. and Chung and Liu's [23] (Chapter 3) do not seem to have an obvious smaller label size than the one in Proposition 4.

Labeling schemes with a query

The concept of labeling scheme limits the number of nodes participating in a query severely. A relaxed variant thereof, called 1-query labeling scheme [39], assumes that the decoder receives both labels queried, and may access the label of a third node in order to answer the query. If this is allowed, we can construct an $O(\log n)$ 1-query adjacency labeling scheme for sparse (and power-law) graphs as follows: We assign each node v with an identifier $ID(v)$, then produce a classic [26] chaining perfect hash-function[4] from $\{1 \dots cn\}$ to $\{1 \dots n\}$, with the guarantee that the worst case number of collisions is constant. We then compute the hash function for all edges (u, v) and store the tuple $\langle ID(v), ID(u) \rangle$ in the label of the corresponding vertex. The decoder first computes the hash value resulting from $ID(v)$ and $ID(u)$ and proceed to examine if on the label corresponding to the result of the function the tuple appears. The decoder needs only to know the primary and secondary hash functions used, description thereof amount to logarithmic number of bits, which can be concatenated to each label.

---

[2] the arboricity of a graph is the minimum number of spanning forests needed to cover its edges.

[3] More precisely, for any $\epsilon \in (0, 1)$ there exist an $O(|E(G)|/\epsilon)$ algorithm [42] that computes such partition using at most $(1 + \epsilon)$ times more forests than the optimal one.

[4] To this end, we may for example first partition the domain into c parts.

7 A distance labeling scheme

In this section we extend the usefulness of our strategy by showing a labeling scheme for small distances in power-law graphs.

For sparse graphs, Alstrup et al. [7] obtain a distance labeling scheme with maximum label size $O(\frac{n}{D}\log^2 D)$ where $D = (\log n)/(\log \frac{m+n}{n})$ and m is the number of edges in the graph. Gawrychowski et al. obtain an upper bond of [33] $O(\frac{n}{D}\log D)$ with sub-linear decoding time. Few general results on lower bounds exist. The lower bound of $\Omega(\sqrt{n})$ for adjacency given in the present paper is trivially also a lower bound for distance; for total label size, the best known lower bound remains $\Omega(n^{3/2})$ as proved by Gavoille et al. [31].

▶ **Lemma 7.** *For any computable $f : \mathbb{N} \longrightarrow \mathbb{N}$ such that $f(n) \leq n-1$ for all n, and for any $\chi(n) \geq n^{1/(alpha-1+f(n))}$ there is an $f(n)$-distance labeling scheme for $\mathcal{P}_{h,\chi,\alpha}$ that assigns labels of length at most $O(n^{f(n)/(f(n)+1)}\log f(n))$.*

Proof. Let G be a graph in $\mathcal{P}_{h,\chi,\alpha}$. A node of G is *fat* if it has degree at least $n^{1/(\alpha-1+f(n))}$ and *thin* otherwise. The label of each node v contains (i) a table of distances to all fat nodes (if the distance is more than $f(n)$, it is simply ignored), (ii) a table of distances to all thin nodes w that are at most distance $f(n)$ away from v where the shortest path between v and w does not pass through any fat node, and (iii) a single bit signifiying whether the node is fat or thin. Clearly, as $f(n)$ is computable and distances in G are computable, there is a computable encoder assigning labels. A decoder can now compute the distance between any two nodes u, v as follows: If both u or v are fat, the distance can be directly read off part (i) of the label of any node. If at least one of u and v is fat, the distance can be read off part (i) of the label of the thin node. If both nodes are thin, the decoder can check if the distance is in part (ii) of the label of either node; if the distance is not present, either the distance is strictly greater than $f(n)$, or the shortest path between u and v passes through a fat node; in this case, the decoder may brute-force check the distances from u and v to each fat node, and output the smallest sum of these two distances.

Furthermore, as all nodes of G are either thin or fat, it is clearly possible for an encoder to compute all distances less than or equal to $f(n)$ between any pair of nodes. Note that as all distances we care for are bounded above by $f(n)$, each such distance can be stored using at most $\log f(n)$ bits.

As $G = G(V, E)$ is in $\mathcal{P}_{h,\chi,\alpha}$, we have

$$\sum_{i=\chi(n)}^{n-1} |V_i| \leq \sum_{i=n^{\frac{1}{\alpha-1+f(n)}}}^{n-1} |V_i| \leq C'\left(\frac{n}{\left(n^{\frac{1}{\alpha-1+f(n)}}\right)^{\alpha-1}}\right)$$
$$\leq C' n^{1-(\alpha-1)/\alpha-1+f(n)} = C' n^{f(n)/(\alpha-1+f(n))}$$

Thus, a table of distances to all fat nodes takes up at most $O\left(n^{\frac{f(n)}{\alpha-1+f(n)}}\log f(n)\right)$ bits.

Similarly, for each node v there are at most $\left(n^{1/(\alpha-1+f(n))}\right)^{f(n)}$ $= n^{f(n)/(\alpha-1+f(n))}$ nodes at distance at most $f(n)$ away from v where the shortest path consists only of thin nodes. Hence, the associated table of distances takes up at most $O(n^{f(n)/(\alpha-1+f(n))}\log n)$ bits.

In total, each label thus has size at most $O(n^{f(n)/(f(n)+1)}\log n)$ bits. ◀

For $f(n) = \log n$, Lemma 7 yields labels of size $O\left(n^{(\log n)/(\alpha-1+\log n)}\log\log n\right)$. Unsurprisingly, as we are only considering distances up to $f(n)$, this label size is asymptotically

smaller than for the labeling schemes working for all distances in *sparse* graphs, e.g. the largest label sizes of [33] for sparse graphs is $O(n\frac{\log\log n}{\log n})$. For power-law random graphs, Chung and Lu show in [22] that, subject to mild conditions, the diameter of power-law graphs with $\alpha > 2$ is almost surely $\Theta(\log n)$. We thus expect our labeling scheme to have superior performance for such graphs.

8 Conclusion and Future Work

We have devised adjacency and distance labeling schemes for sparse graphs and graphs whose degree distribution approximately follows a power-law distribution. We have proven lower bounds for the class of power-law graphs showing that our strategy for adjacency labeling scheme is almost optimal, and showed two relaxations that allow for logarithmic size labels. In the full version of the paper we also validate experimentally that the labeling scheme for power-law graphs obtains results in practice requiring little space, and that the theoretical threshold we use in our strategy is reasonably close to the optimum threshold.

8.1 Future work

We propose the following directions:

- Our labeling schemes are designed for static networks, and while it seems not difficult to extend our idea to dynamic networks, an analysis is required to account for the communication and number of re-labels incurred by such an extension.
- Labeling schemes for power-law graphs can likely be devised for the realistic case where the scheme only has incomplete knowledge of the graph, for example when the expected frequency of vertices of each degree is known, but not the exact frequency of each vertex.
- Closing the gap of the multiplicative logarithmic factor may be of interest to the theory community. A more interesting gap exists for distance labeling schemes. As we have seen, there is a large gap between labeling schemes for short distance and adjacency for power-law (and sparse) graphs. This gap effectively deemed the distance labels uninteresting for practical applications.
- Finally, while power-law distributions may model the degree distribution of real-world networks, other distributions may fit better (see, e.g., [24]); it is interesting to see whether refinements of our labeling scheme that utilize knowledge about such distributions would result in superior labeling schemes for real-world data.

References

1 Ittai Abraham, Daniel Delling, Andrew V Goldberg, and Renato F Werneck. A hub-based labeling algorithm for shortest paths in road networks. In *Experimental Algorithms*, pages 230–241. Springer, 2011.

2 Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cristopher Moore. On the bias of traceroute sampling: Or, power-law degree distributions in regular graphs. *J. ACM*, 56(4), 2009. doi:10.1145/1538902.1538905.

3 David Adjiashvili and Noy Rotbart. Labeling schemes for bounded degree graphs. In *Automata, Languages, and Programming*, pages 375–386. Springer, 2014.

4 Aditya Akella, Shuchi Chawla, Arvind Kannan, and Srinivasan Seshan. Scaling properties of the internet graph. In *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003*, pages 337–346, 2003. doi:10.1145/872035.872087.

5 Noga Alon and Vera Asodi. Sparse universal graphs. *J. Comput. Appl. Math.*, 142(1):1–11, May 2002. doi:10.1016/S0377-0427(01)00455-1.

6 Stephen Alstrup, Søren Dahlgaard, and Mathias Bæk Tejs Knudsen. Optimal induced universal graphs and adjacency labeling for trees. In *Proceedings of the 58th Symposium on Foundations of Computer Science*, FOCS'15, Washington, DC, USA, 2015. IEEE Computer Society.

7 Stephen Alstrup, Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Ely Porat. Sublinear distance labeling for sparse graphs. *CoRR*, abs/1507.02618, 2015. URL: `http://arxiv.org/abs/1507.02618`.

8 Stephen Alstrup, Haim Kaplan, Mikkel Thorup, and Uri Zwick. Adjacency labeling schemes and induced-universal graphs. *To appear in the 47th symposium on Theory of computing (STOC)*, 2015.

9 Stephen Alstrup and Theis Rauhe. Small induced-universal graphs and compact implicit graph representations. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, FOCS'02, pages 53–62, Washington, DC, USA, 2002. IEEE Computer Society. URL: `http://dl.acm.org/citation.cfm?id=645413.652151`.

10 Srinivasa R Arikati, Anil Maheshwari, and Christos D Zaroliagis. Efficient computation of implicit representations of sparse graphs. *Discrete Applied Mathematics*, 78(1):1–16, 1997.

11 Laszlo Babai, Fan RK Chung, Pál Erdös, Ronald L Graham, and J Spencer. On graphs which contain all sparse graphs. *Ann. Discrete Math*, 12:21–26, 1982.

12 Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

13 Paolo Boldi, Marco Rosa, Massimo Santini, and Sebastiano Vigna. Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks. In *Proceedings of the 20th international conference on World Wide Web*, pages 587–596. ACM, 2011.

14 Paolo Boldi and Sebastiano Vigna. The webgraph framework I: compression techniques. In *Proceedings of the 13th international conference on World Wide Web*, pages 595–602. ACM, 2004.

15 Béla Bollobás, Oliver Riordan, Joel Spencer, and Gábor E. Tusnády. The degree sequence of a scale-free random graph process. *Random Struct. Algorithms*, 18(3):279–290, 2001. `doi:10.1002/rsa.1009`.

16 Pawek Brach, Marek Cygan, Jakub Lacki, and Piotr Sankowski. Algorithmic complexity of power law networks. In *To appear in Proceedings of the thirty third annual ACM-SIAM symposium on Discrete algorithms*, SODA'16, 2014. URL: `http://arxiv.org/abs/1507.02426`.

17 Arthur Brady and Lenore J Cowen. Compact routing on power law graphs with additive stretch. In *ALENEX*, volume 6, pages 119–128. SIAM, 2006.

18 Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 46–52. ACM, 2009.

19 Kenneth L Calvert, Matthew B Doar, and Ellen W Zegura. Modeling internet topology. *Communications Magazine, IEEE*, 35(6):160–163, 1997.

20 Saverio Caminiti, Irene Finocchi, and Rossella Petreschi. Engineering tree labeling schemes: A case study on least common ancestors. In *Algorithms-ESA 2008*, pages 234–245. Springer, 2008.

21 Wei Chen, Christian Sommer, Shang-Hua Teng, and Yajun Wang. A compact routing scheme and approximate distance oracle for power-law graphs. *ACM Transactions on Algorithms*, 9(1):4, 2012.

22 Fan Chung and Linyuan Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–113, 2004.

23 Fan RK Chung and Linyuan Lu. *Complex Graphs and Networks*, volume 107. American mathematical society Providence, 2006.

24 Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

25 Edith Cohen, Haim Kaplan, and Tova Milo. Labeling dynamic xml trees. *SIAM Journal on Computing*, 39(5):2048–2074, 2010.

26 Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

27 Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Noy Rotbart. Dynamic and multi-functional labeling schemes. In *Algorithms and Computation*, pages 141–153. Springer, 2014.

28 YH Eom, S Fortunato, and Matjaz Perc. Characterizing and modeling citation dynamics. *PLoS ONE*, 6(9):e24926, 2011.

29 Johannes Fischer. Short labels for lowest common ancestors in trees. In *Algorithms-ESA 2009*, pages 752–763. Springer, 2009.

30 Cyril Gavoille and Arnaud Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In *Algorithms–ESA 2007*, pages 582–593. Springer, 2007.

31 Cyril Gavoille, David Peleg, Stéphane Pérennes, and Ran Raz. Distance labeling in graphs. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA'01, pages 210–219, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics. URL: `http://dl.acm.org/citation.cfm?id=365411.365447`.

32 Cyril Gavoille, David Peleg, Stéphane Pérennesc, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53:85–112, 2004.

33 Pawel Gawrychowski, Adrian Kosowski, and Przemyslaw Uznanski. Even simpler distance labeling for (sparse) graphs. *CoRR*, abs/1507.06240, 2015. URL: `http://arxiv.org/abs/1507.06240`.

34 Gaurav Goel and Jens Gustedt. Bounded arboricity to determine the local structure of sparse graphs. In *Graph-Theoretic Concepts in Computer Science*, pages 159–167. Springer, 2006.

35 Joseph E Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. Power-graph: Distributed graph-parallel computation on natural graphs. *OSDI*, 12(1):2, 2012.

36 Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. In *SIAM Journal On Discrete Mathematics*, pages 334–343, 1992.

37 Michal Katz, Nir A Katz, Amos Korman, and David Peleg. Labeling schemes for flow and connectivity. *SIAM Journal on Computing*, 34(1):23–40, 2004.

38 Amos Korman. General compact labeling schemes for dynamic trees. *Distributed Computing*, 20(3):179–193, 2007.

39 Amos Korman and Shay Kutten. Labeling schemes with queries. In *Structural Information and Communication Complexity*, pages 109–123. Springer, 2007.

40 Amos Korman and David Peleg. Compact separator decompositions in dynamic trees and applications to labeling schemes. In *Distributed Computing*, pages 313–327. Springer, 2007.

41 Amos Korman and David Peleg. Labeling schemes for weighted dynamic trees. *Inf. Comput.*, 205(12):1721–1740, December 2007. `doi:10.1016/j.ic.2007.08.004`.

42 Łukasz Kowalik. Approximation scheme for lowest outdegree orientation and graph density measures. In *Algorithms and computation*, pages 557–566. Springer, 2006.

43 Dmitri Krioukov, Kevin Fall, and Xiaowei Yang. Compact routing on internet-like graphs. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2004.

44 Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

45 Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

46 Robert Ryan McCune, Tim Weninger, and Greg Madey. Thinking like a vertex: a survey of vertex-centric frameworks for large-scale distributed graph processing. *CoRR*, abs/1507.04405, 2015. URL: `http://arxiv.org/abs/1507.04405`.

47 Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2):226–251, 2004.

48 JW Moon. On minimal n-universal graphs. *Proceedings of the Glasgow Mathematical Association*, 7(01):32–33, 1965.

49 Noy Rotbart, Marcos Vaz Salles, and Iasonas Zotos. An evaluation of dynamic labeling schemes for tree networks. In *Experimental Algorithms*, pages 199–210. Springer, 2014.

50 Georgos Siganos, Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. Power laws and the as-level internet topology. *IEEE/ACM Trans. Netw.*, 11(4):514–524, 2003. `doi:10.1109/TNET.2003.815300`.

51 Jeremy P Spinrad. *Efficient graph representations*. American mathematical society, 2003.

52 Isabelle Stanton and Gabriel Kliot. Streaming graph partitioning for large distributed graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2012.

53 Bernard M Waxman. Routing of multipoint connections. *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622, 1988.

54 Cong Xie, Ling Yan, Wu-Jun Li, and Zhihua Zhang. Distributed power-law graph computing: Theoretical and empirical analysis. In *Advances in Neural Information Processing Systems*, pages 1673–1681, 2014.

On the Resiliency of Randomized Routing Against Multiple Edge Failures*†

Marco Chiesa[1], Andrei Gurtov[2], Aleksander Mądry[3], Slobodan Mitrović[4], Ilya Nikolaevskiy[5], Michael Schapira[6], and Scott Shenker[7]

1 **Université catholique de Louvain, Louvain-la-Neuve, Belgium**
2 **Helsinki Institute for Information Technology, Helsinki, Finland; and Aalto University, Department of Computer Science, Aalto,Finland**
3 **Massachusetts Institute of Technology, Cambridge, USA**
4 **École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland**
5 **Aalto University, Department of Computer Science, Aalto, Finland**
6 **Hebrew University of Jerusalem, Jerusalem, Israel**
7 **University of California, Berkeley, USA; and International Computer Science Institute, Berkeley, USA**

Abstract

We study the Static-Routing-Resiliency problem, motivated by routing on the Internet: Given a graph G = (V, E), a unique destination vertex d, and an integer constant c > 0, does there exist a static and destination-based routing scheme such that the correct delivery of packets from any source s to the destination d is guaranteed so long as (1) no more than c edges fail and (2) there exists a physical path from s to d? We embark upon a study of this problem by relating the edge-connectivity of a graph, i.e., the minimum number of edges whose deletion partitions G, to its resiliency. Following the success of randomized routing algorithms in dealing with a variety of problems (e.g., Valiant load balancing in the network design problem), we embark upon a study of randomized routing algorithms for the Static-Routing-Resiliency problem. For any k-connected graph, we show a surprisingly simple randomized algorithm that has expected number of hops O(|V|k) if at most k-1 edges fail, which reduces to O(|V|) if only a fraction t of the links fail (where t < 1 is a constant). Furthermore, our algorithm is deterministic if the routing does not encounter any failed link.

1998 ACM Subject Classification C.2.2 Network Protocols

Keywords and phrases Randomized, Routing, Resilience, Connectivity, Arborescenses

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.134

1 Introduction

Routing on the Internet (both within an organizational network and between such networks) typically involves computing a set of *destination-based* routing tables (i.e., tables that map the destination IP address of a packet to an outgoing link). Whenever a link or node fails, routing tables are recomputed by invoking the routing protocol to run again (or having it

* Additional details of the results in this note appear in [9].

† This research was supported in part by European Union's Horizon 2020 research and innovation programme under the ENDEAVOUR project (grant agreement 644960), by Swiss National Science Foundation (grant number P1ELP2_161820), the NSF award 1553428, and a Sloan Research Fellowship.

run periodically, independent of failures). This produces well-formed routing tables, but results in relatively long outages after failures as the protocol is recomputing routes.

As critical applications began to rely on the Internet, such outages became unacceptable. As a result, "fast failover" techniques have been employed to facilitate immediate recovery from failures. The most well-known of these is Fast Reroute in MPLS where, upon a link failure, packets are sent along a precomputed alternate path without waiting for the global recomputation of routes [23]. This, and other similar forms of fast failover thus enable rapid response to failures but are limited to the set of precomputed alternate paths. Most of existing approaches protect only from a single failure, however in many scenarios (e.g. overlay networks [11], highly-connected large datacenter networks [15]) multiple failures at the same time may be a common occurrence.

The goal of this paper is to perform a theoretical study of failover routing. The fundamental question is, how resilient can failover routing be? That is, how many link failures can failover routing schemes tolerate before connectivity is interrupted (i.e., packets are trapped in a forwarding loop, or hit a dead end)? The answer to this question depends on both the structural properties of the graph, and the limitations imposed on the routing scheme.

Clearly, if it is possible to store arbitrary amount of information in the packet header, perfect resiliency can be achieved by collecting information about every failed link that is hit by a packet [19, 26]. Such approaches are not feasibly deployable in modern-day networks as the header of a packet may be too large for today's routing tables. Our focus is thus on failover routing schemes that do not involve any change in the packet headers. Another traditional approach to achieving high resiliency is implementing stateful routing, i.e., storing information at a node every time a packet is seen being received from a different incoming link (see, e.g., link reversal [14] and other approaches [20, 21]). As current routing protocols do not allow network operators to implement such stateful failover routing, our goal is to design protocols that correspond to a stateless, or *static*, failover routing.

Specifically, we consider a particularly simple and practical form of static failover routing: for each incoming link, a router maintains a destination-based routing table that maps the destination address of a packet and the set of non-failed ("active") links, to an output link. The router can locally detect which outgoing links are down and forwards packets accordingly. One should note that maintaining such per-incoming-link destination-based routing tables is necessary; not only is destination-based routing unable to achieve robustness against even a single link failure [18], but it is even computationally hard to devise failover routing schemes that maximize the number of nodes that are protected [2, 5, 18, 24]. We only consider link failures, not router failures (which are not always detectable by neighboring routers, and so such fast failover techniques may not apply).

A failover routing algorithm is responsible for computing, for each node (vertex) of a network (graph), a *routing function* that *matches* an incoming packet to an outgoing edge. A set of such routing functions for each vertex guarantees *reachability* between a pair of vertices, u and v, for which there exists a connecting path in the graph, if any packet directed to node v originated at node u is correctly routed from u to v.

We are interested in routing functions that rely solely on information that is locally available at a node (e.g., the set of non-failed edges, the incoming link along which the packet arrived, and any information stored in the header of the packet).

While it is known that every k-connected network cannot be partitioned by deleting at most $k-1$ links, it is not known whether any static "deterministic" routing (i.e., the outgoing port of each packet is always uniquely determined at a vertex v by its incoming link and the failed edges incident at v) achieves such resiliency.

On the other hand, routing based on random walks, i.e., choosing the outgoing link at random, achieves the best possible resilience as they will eventually deliver a packet to the destination as long as the network is connected. But, it comes with a huge cost. Namely, a random walk might traverse the whole network even when there is no single failed link. In fact, the expected delivery time of a packet would be as large as $\Theta(|V|^3)$ in some network topologies [6]. Furthermore, a random walk almost never reaches the destination following the shortest path. So, although extremely robust, when it comes to the time needed to deliver a packet to the destination, the behaviour of random walks is undesirable.

1.1 Our Results

In this paper, we show how randomness can be used to achieve $k-1$ resilient routing in k-connected networks while significantly outperforming random walks in terms of number of traversed nodes. Namely, we introduce **Randomized failover routing (`RND`)** in which outgoing edge is chosen for packets in a probabilistic manner based on the destination label, the incoming edge, and the set of non-failed edges. The randomized protocol that we present provides bound on the expected delivery time that gracefully grows with the number of actual link failures.

Our randomized routing functions provide delivery in case of any $k-1$ link failures for any k-connected graph. We achieve that by leveraging the standard decomposition of k-connected graphs into k arc-disjoint spanning arborescences $\mathcal{T}$ [10]. We also provide a bound on the expected number of hops that our algorithm performs, which is $O(Hk)$ for any $k-1$ failures and $O(H)$ for αk failures, where H is the length of the longest branch of any arborescence of $\mathcal{T}$ and $\alpha < 1$ is a constant. Furthermore, our routing functions are deterministic as long as the routing does not encounter any failure. Hence, packets belonging to the same logical connection are routed along the same path, minimizing reordering complexity at the receiver side.

Motivated by the fact that one can protect against $k-1$ failures in k-connected graphs using randomness, we make the following general conjecture, whose proof eludes us despite much effort.

- **Conjecture:** For any k-connected graph, one can find deterministic failover routing functions that are robust to any $k-1$ failures.

1.2 Organization

The rest of the paper is organized as follows. Section 2 provides background on existing works. In Section 3, we introduce our routing model and formally state the Static-Routing-Resiliency problem. The summary of our routing techniques that are leveraged throughout the whole paper are presented in Section 4. In Section 5, we focus on studying the relation between arborescences our input graph decomposes into and failed links. Section 6 builds on Section 5 and is devoted to designing an algorithm that, for any k-connected graph, computes randomized routing functions that are robust to $k-1$ edge failures and have bounded expected delivery time.

2 Related Work

Past work [1, 29] (1) designed such routing functions with guaranteed robustness against *only* a single link/node failure [12, 13, 22, 28, 30, 32], (2) achieved robustness against $\lfloor \frac{k}{2} - 1 \rfloor$

edge failures for k-connected graphs [11], and (3) proved that it is impossible to be robust against any set of edge failures that does not partition the network [13].

Thanks to its flexibility and oblivious behavior, another line of study was motivated by randomization. Namely, some of the previous work developed randomized routing schemes, usually to directly or indirectly achieve low congestion and/or balance the network load. In particular, Busch et al. [7] use randomization to adjust packet priorities, which in turn allows them to control deflection of packets.

Valiant [27] proposed a randomized routing algorithm with the goal to balance the load of the underlying network. Since then, that scheme is called *Valiant Load-Balancing* (VLB), whose one of the main ingredients is randomization. VLB was extensively used in designing networks. Zhang-Shen et al. [31] employed VLB to design fault-tolerant networks with guaranteed no congestion under few router or link failures. Greenberg et al. [16] adopt VLB to reduce volatility of traffic and failure pattern of their data centers. In [25], Shepherd et al. extend VLB in order to build cost-effective networks robust to changes in demand patterns.

Beraldi [3] presents a search protocol for mobile networks that is based on modified random walks, i.e. based on biased random walks with look-ahead. Motivated by the success of ant-colonies in their search for food, Günes et al.[17] studied ant algorithms, which in their heart rely on randomization, as an approach to designing on-demand ad-hoc routing algorithms.

Chiesa et al. [8] studied resilience under link failures in k-connected networks. They devise static routing schemes that are resilient under $k-1$ failures in the following regimes: (1) if the routers are allowed to use three bits in the packet header for read/write operation, or (2) if the network supports broadcasting. A building block of those schemes is the result that every k-connected graph contains k arc-disjoint arborescences rooted at the same vertex [10].

3 Model

We represent our network as an undirected multigraph $G = (V(G), E(G))$, where each router in the network is modeled by a vertex in $V(G)$ and each link between two routers is modeled by an undirected edge in the multiset $E(G)$. When it is clear from the context, we simply write V and E instead of $V(G)$ and $E(G)$. We denote an (undirected) edge between x and y by $\{x, y\}$. A graph is *k-edge-connected* if there exist k edge-disjoint paths between any pair of vertices of G.

Each vertex v routes packets according to a *routing function* that matches an incoming packet to a sequence of forwarding actions. Packet *matching* is performed according to the set of active (non-failed) edges incident at v, the incoming edge, and any information stored in the packet header (e.g., destination label, extra bits), which all are *locally* available at a vertex.

Since our focus is on per-destination routing functions, we assume that there exists a unique destination $d \in V$ to which every other vertex wishes to send packets and, therefore, that the destination label is not included in the header of a packet. Forwarding *actions* consist of routing packets through an outgoing edge, rewriting some bits in the packet header, and creating duplicates of a packet.

In this paper we consider *randomized* routing functions, in which a vertex forwards a packet through an outgoing edge with a probability based only on the incoming port and the set of active outgoing edges. We present the formal definitions of the randomized routing model in Section 6.

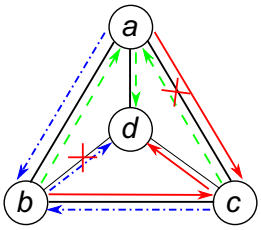


Figure 1 A 3-connected graph with 3 arc-disjoint arborescences colored red, blue, and green.

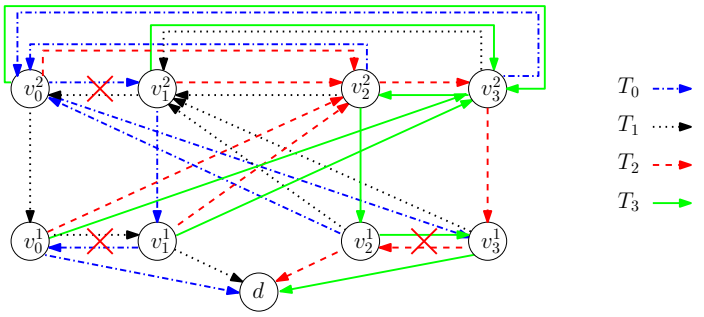


Figure 2 Graph used in the proof of Theorem 7 for $k = 2$.

The Static-Routing-Resiliency (Srr) problem. Given a graph G, a routing function f is *k-resilient* if, for each vertex $v \in V$, a packet originated at v and routed according to f reaches its destination d as long as at most k edges fail and there still exists a path between v and d. The input of the Srr problem is a graph G, a destination $d \in V(G)$, and an integer $k > 0$, and the goal is to compute a set of resilient routing functions that is k-resilient.

4 General Routing Techniques and Randomized Algorithm

Definition and notation. We denote a directed arc from x to y by (x, y) and by $\vec{G}$ the directed copy of G, i.e. a directed graph such that $V(\vec{G}) = V$ and $\{x, y\} \in E$ if and only if $(x, y), (y, x) \in E(\vec{G})$.

A subgraph T of $\vec{G}$ is an *r-rooted arborescence* of $\vec{G}$ if (i) $r \in V$, (ii) $V(T) \subseteq V$, (iii) r is the only vertex without outgoing arcs and (iv), for each $v \in V(T) \setminus \{r\}$, there exists a single directed path from v to r that only traverses vertices in $V(T)$. If $V(T) = V$, we say that T is a r-rooted *spanning* arborescence of $\vec{G}$. When it is clear from the context, we use the word "arborescence" to refer to a d-rooted spanning arborescence, where d is the destination vertex. We say that two arborescences T_1 and T_2 are *arc-disjoint* if $(x, y) \in E(T_1) \implies (x, y) \notin E(T_2)$. A set of l arborescences $\{T_1, \ldots, T_l\}$ is *arc-disjoint* if the arborescences are pairwise arc-disjoint. We say that two arc-disjoint arborescences T_1 and T_2 do not *share* an edge $\{x, y\} \in E$ if $(x, y) \in E(T_1) \implies (y, x) \notin E(T_2)$.

For example, consider Fig. 1, in which each pair of nodes is connected by an edge (ignore the red crosses) and three arc-disjoint (d-rooted spanning) arborescences `Red`, `Green`, and `Blue` are depicted by colored arrows.

Arborescence-based routing. Throughout the paper, unless specified otherwise, we let $\mathcal{T} = \{T_1, \ldots, T_k\}$ denote a set of k d-rooted arc-disjoint spanning arborescences of $\vec{G}$. All our routing techniques are based on a decomposition of $\vec{G}$ into $\mathcal{T}$. The existence of k arc-disjoint arborescences in any k-connected graph was proven in [10], while fast algorithms to compute such arborescences can be found in [4]. We say that a packet is routed in *canonical* mode along an arborescence T if a packet is routed through the unique directed path of T towards the destination. If the packet hits a failed edge at vertex v along T, it is processed by v (e.g., duplication, header-rewriting) according to the capabilities of a specific routing function and it is rerouted along a different arborescence. We call such routing technique *arborescence-based* routing. One crucial decision that must be taken is the next arborescence

to be used after a packet hits a failed edge. In this paper, we propose two natural choices that represent the building blocks of all our routing functions. When a packet is routed along T_i and it hits a failed arc (v, u), we consider the following two possible actions:

- **Reroute along some available arborescence**, e.g., reroute along T', where T' is chosen randomly from distribution that we define in the sequel. Observe that, if the outgoing arc belonging to T' failed, we randomly pick another arborescence T'', and so on.
- **Bounce on the reversed arborescence**, i.e., we reroute along the arborescence T_{next} that contains arc (u, v).

To grasp how bouncing enters in our picture for obtaining $k-1$ resiliency, consider the following case. Assume that in the network there are $k/2$ failed links, such that every single out of k arborescences contains one of the links. (As a reminder, arborescences that we construct might share links, but not arcs.) So, this example might suggest that there are scenarios in which already $k/2$ failed links make all the arborescences not very useful, and that no algorithm can cope with that. But, there is a twist. Let $k = 2$, and T_i and T_j be the two arborescences and let them share the same failed edge a. Furthermore, let a be the only failed edge T_i and T_j contain. If a packet hits a while routed along T_i or T_j, then after bouncing on a the packet will reach d without any further interruption! So, we have just found a way to resolve a case in which every arborescence contains one failed link, and that is not an isolated scenario, as we discuss in the sequel.

From a different point of view, bouncing is a way of recycling arborescences that contain one failed link. This observation is crucial to obtain an efficient and a simple randomized $(k-1)$-resilient routing scheme, which we are now ready to present. The algorithm is parametrized by q that we define later.

Algorithm 1 Definition of Rand-Bouncing-Algo.

Rand-Bouncing-Algo: Given $\mathcal{T} = \{T_1, \ldots, T_k\}$

1. $T :=$ an arborescence from $\mathcal{T}$ sampled uniformly at random (u.a.r.)
2. While d is not reached
   a. Route along T (canonical mode)
   b. If a failed edge is hit then
      i. With probability q, replace T by an arborescence from $\mathcal{T}$ sampled u.a.r.
      ii. Otherwise, bounce the failed edge and update T correspondingly

In the following sections, we first study the connection between arborescences of $\mathcal{T}$ and failed links, and show how a part of their intricate interaction can be represented in a simple and an elegant way via, so-called, *meta-graph*. Afterwards, we show the Rand-Bouncing-Algo is $(k-1)$-resilient and we analyze its efficiency.

5 Meta-graph, Good Arcs, and Good Arborescences

The goal of this section is to provide an understanding of the structural relation between the arborescences of $\mathcal{T}$ when the underlying k-connected network has at most $k-1$ failed edges. The perspective that we are building here drives the construction of our randomized algorithm.

We start by introducing the notion of a *meta-graph*. To that end, we fix an arbitrary set of failed edges F. Throughout the section, we assume $|F| < k$, and define $f := |F|$. Then, we define a meta-graph $H_F = (V_F, E_F)$ as follows:

- $V_F = \{1, \ldots, k\}$, where vertex i is a representative of arborescence T_i.
- For each failed edge $e \in E$ belonging to at least one arborescences of $\mathcal{T}$ we define the corresponding edge e_F in H_F in the following way:
  - $e_F := \{i, j\}$, if e belongs to two different arborescences T_i and T_j;
  - $e_F := \{i, i\}$, i.e. e_F is a self-loop, if e belongs to a single arborescence T_i only.

Note that in our construction H_F might contain parallel edges. Intuitively, the meta-graph represents a relation between arborescences of $\mathcal{T}$ for a fixed set of failed edges. We provide the following lemma as the first step towards understanding the structure of H_F.

▶ **Lemma 1.** *The set of connected components of H_F contains at least $k - f$ trees.*

Proof. We give a proof by contradiction. To that end, assume that the set of connected components of H_F, denoted by $\mathcal{C}$, contains at most $k - f - 1$ trees. Now, if $C \in \mathcal{C}$ is a tree, we have $|E(C)| = |V(C)| - 1$, and $|E(C)| \geq |V(C)|$ otherwise. We also have

$$\begin{aligned} \sum_{C \in \mathcal{C}} |E(C)| &= \sum_{C \in \mathcal{C} \text{ is not a tree}} |E(C)| + \sum_{C \in \mathcal{C} \text{ is a tree}} |E(C)| \\ &\geq \sum_{C \in \mathcal{C} \text{ is not a tree}} |V(C)| + \sum_{C \in \mathcal{C} \text{ is a tree}} (|V(C)| - 1). \end{aligned} \tag{1}$$

Next, following our assumption that $\mathcal{C}$ contains at most $k - f - 1$ trees, from (1) we obtain

$$\sum_{C \in \mathcal{C}} |E(C)| \geq \sum_{C \in \mathcal{C}} |V(C)| - (k - f - 1). \tag{2}$$

Furthermore, as by the construction we have $\sum_{C \in \mathcal{C}} |V(C)| = |V_F| = k$, (2) implies

$$\sum_{C \in \mathcal{C}} |E(C)| \geq |V_F| - (k - f - 1) = f + 1. \tag{3}$$

On the other hand, from the construction of H_F we have

$$\sum_{C \in \mathcal{C}} |E(C)| = f,$$

which leads to a contradiction with (3). ◀

Lemma 1 implies that the fewer failed edges there are, the larger fraction of connected components of the meta-graph H_F are trees. Note that an isolated vertex is a tree as well.

In the sequel, we show that each tree-component of H_F contains at least one vertex corresponding to an arborescence from which any bounce on a failed edge leads to the destination d without hitting any new failed edge. To that end, we introduce the notion of good arcs and good arborescences. We say that an arc (u, v) is a *good arc* of an arborescence T if on the (unique) v-d path in T there is no failed edge. Let $a = (i, j)$, for $i \neq j$, be an arc of $\vec{H_F}$, $\{u, v\}$ be the edge that corresponds to a, and w.l.o.g. assume (u, v) is an arc of T_j. Then, we say a is a *well-bouncing* arc if (u, v) is a good arc of T_j. Intuitively, a well-bouncing arc (i, j) of $\vec{H_F}$ means that by bouncing from T_i to T_j on the failed edge $\{v, u\}$ the packet will reach d via routing along T_j without any further interruption. Finally, we say that an arborescence T_i is a *good arborescence* if every outgoing arc of vertex $i \in V_F$ is well-bouncing.

▶ **Lemma 2.** *Let T be a tree-component of H_F s.t. $|V(T)| > 1$. Then, $\vec{T}$ contains at least $|V(T)|$ well-bouncing arcs.*

Proof. Let T_i be an arborescence of $\mathcal{T}$ such that $i \in V(T)$. Then, by the construction of H_F we have that T_i contains a failed link. Next, a failed link closest to the root of T_i is a good arc of T_i. Therefore, for every $i \in V(T)$, we have that T_i contains an arc which is both good and failed. Furthermore, by the construction of H_F and the definition of well-bouncing arcs, we have that for every good, failed link of T_i there is the corresponding well-bouncing arc of $\vec{T}$. Also, observe that the construction of H_F implies that a well-bouncing arc corresponds to exactly one good-arc.

Now, putting all the observations together, we have that each T_i, for every $i \in V(T)$, has a good failed link which further corresponds to a well-bouncing arc of $\vec{T}$. As all the arborescences are arc-disjoint, and there are $|V(T)|$ many of them represented by the vertices of T, we have that $\vec{T}$ contains at least $|V(T)|$ well-bouncing arcs. ◀

Now, building on Lemma 2, we prove the following.

▶ **Lemma 3.** *Let T be a tree-component of H_F. Then, there is an arborescence T_i such that $i \in V(T)$ and T_i is good.*

Proof. Consider two cases: $|V(T)| = 1$, and $|V(T)| > 1$. In the case $|V(T)| = 1$, T is an isolated vertex which implies that it has no outgoing arcs. Therefore, T represents a good arborescence.

If $|V(T)| > 1$, then from Lemma 2 we have that $\vec{T}$ contains at most $2(|V(T)|-1)-|V(T)| < |V(T)|$ arcs which are not well-bouncing. This implies that there is at least one vertex in T from which every outgoing arc is well-bouncing. ◀

Let us understand what this implies. Consider an arborescence T_i, and a routing of a packet along it. In addition, assume that the routing hits a failed edge e, such that e is shared with some other arborescence T_j. Now, if e corresponds to a well-bouncing arc of $\vec{H_F}$, then by bouncing on e and routing solely along T_j, the packet will reach d without any further interruption. Lemma 3 claims that for each tree-component T of H_F there always exists an arborescence T_i, with $i \in V(T)$, which is good, i.e. every failed edge of T_i corresponds to a well-bouncing arc of $\vec{H_F}$.

We can now state the main lemma of this section.

▶ **Lemma 4.** *If G contains at most $k-1$ failed edges, then $\mathcal{T}$ contains at least one good arborescence.*

Proof. We prove that there exists an arborescence T_i such that if a packet bounces on any failed edge of T_i it will reach d without any further interruption. Let F be the set of failed edges, at most $k-1$ of them. Then, by Lemma 1 we have that H_F contains at least $k-f \geq 1$ tree-components. Let T be one such component.

By Lemma 3, we have that there exists at least an arborescence T_i such that every outgoing arc from i is well-bouncing. Therefore, bouncing on any failed arc of T_i the packet will reach d without any further interruption. ◀

6 Randomized Routing via Good Arborescences

In this section, we show that a set of routing functions for G obtained by Rand-Bouncing-Algo is $(k-1)$-resilient. Note that our routing function (`RND`) maps an incoming edge and the set of active edges incident at v to a set of pairs (e, q), where e is an outgoing edge and q is the probability of forwarding a packet through e. A packet is forwarded through a unique outgoing edge.

The section is structured as follows. As a prelude, we show a simple, yet inefficient, randomized routing algorithm, called Rand-Algo, that although is $(k-1)$-resilient, fails to achieve low expected number of hops in case of $k-1$ failed edges. We then apply our results from Section 5 to show that Rand-Bouncing-Algo is both $(k-1)$-resilient and requires up to an order fewer number of hops, compared to Rand-Algo, to reach the destination.

6.1 A Simple (Inefficient) Randomized Routing

Consider the following naive randomized algorithm Rand-Algo for routing along arborescences. A packet is routed along the same arborescence until it either reaches its destination or hits a failed edge. In the latter case, it is rerouted along another arborescences chosen uniformly at random. We show that there exists a k-connected graph and a set of failed edges such that the expected number of tree switches that Rand-Algo makes is $\Omega(k^2)$. This further implies that the expected number of hops is $\Omega(Hk^2)$ in the worst case, where H is the length of a longest path in any arborescence and assuming that longest path in all the arborescences are up to a constant factor the same.

To prove the promised bound, we start by defining a $2k$ edge connected graph $G=(V,E)$ and its set of $2k$ arc disjoint spanning trees $T_0,\ldots,T_{2k-1}$ as follows.

- Set V consists of a destination vertex d and $4k$ additional vertices arranged into two equal-sized layers $L_1=\{v_0^1,\ldots,v_{2k-1}^1\}$ and $L_2=\{v_0^2,\ldots,v_{2k-1}^2\}$.
- Set E is defined by the following four subgraphs: (1) L_2 is a clique of size $2k$; (2) (L_1,L_2) is a complete bipartite graph; (3) for each $k=0,\ldots,k-1$, there is an edge (v_{2i}^1,v_{2i+1}^1) and (4) vertex d is connected to each vertex of L_1. There is no other edge included in G.

Next, we construct $2k$ arc-disjoint spanning trees $T_0,\ldots,T_{2k-1}$ (see Fig. 2 for an example with $k=2$). We use $[t]_0$ to denote set $\{0,1,2,\ldots,t-1\}$.

- For each $i\in[k]_0$, add the following arcs:
  - (v_{2i+1}^2,v_{2i}^2), (v_{2i}^2,v_{2i}^1), (v_{2i}^1,v_{2i+1}^1), and (v_{2i+1}^1,d) into T_{2i+1};
  - arcs (v_{2i}^2,v_{2i+1}^2), (v_{2i+1}^2,v_{2i+1}^1), (v_{2i+1}^1,v_{2i}^1), and (v_{2i}^1,d) into T_{2i}.
- For each $i\in[k]_0$, and for each $j\in[2k]_0\setminus\{2i,2i+1\}$, add the following arcs:
  - (v_j^2,v_{2i}^2), and (v_j^1,v_{2i}^2) into T_{2i};
  - (v_j^2,v_{2i+1}^2), and (v_j^1,v_{2i+1}^2) into T_{2i+1}.

Finally, consider a scenario in which edges (v_0^2,v_1^2), (v_2^2,v_3^2), $\ldots$, (v_{2k-4}^2,v_{2k-3}^2) and (v_0^1,v_1^1), (v_2^1,v_3^1), $\ldots$, (v_{2k-4}^1,v_{2k-3}^1), (v_{2k-2}^1,v_{2k-1}^1) failed.

We say that a packet is routed *downwards* (*upwards*) if it is routed from a vertex in L_2 (L_1) to a vertex in L_1 (L_2). Let E_d be the expected number, minimized over all the vertices, of tree switches of a packet that is routed downwards, E_u be the expected number of tree switches of a packet that is routed along T_i and is currently located at v_i^2, for some $i\in[2k-2]_0$, and E_2 be the expected number of tree switches of a packet that is originated by a vertex in L_2. Then, we can show.

▶ **Lemma 5.** *It holds* $E_u \geq \frac{3}{2k-1}E_d + \frac{2k-4}{2k-1}E_u + 1$.

Proof. Let p be routed along T_h and located at v_h^2, for some $h\in[2k-2]_0$. W.l.o.g, let $h=0$. By the construction of T_0, from v_0^1 packet p should be forwarded to v_1^2 but (v_0^2,v_1^2) has failed. So, from v_0^2, p is forwarded downwards along T_1, T_{2k-2} or T_{2k-1} with probability $\frac{3}{2k-1}$ and routed along any other tree T_j to a vertex v_j^2 in L_2 with probability at least $\frac{2k-4}{2k-1}$. Hence, the lemma follows. ◀

▶ **Lemma 6.** *We have* $E_d\in\Omega(k^2)$.

Proof. By the construction, a packet routed downwards traverses arc (v_i^2, v_i^1) of T_i. W.l.o.g, let p be routed along (v_{2i}^2, v_{2i}^1) of T_{2i}. As (v_{2i}^1, v_{2i+1}^1), which belongs to T_{2i}, has failed, p is rerouted along T_j for some $j \in [2k]_0 \setminus \{2i\}$. Among them, only T_{2i+1} has a path from v_{2i}^1 to d that does not contain any failed link. T_{2i+1} is chosen with probability $\frac{1}{2k-1}$.

If any other tree T_j is chosen except T_{2k-2} and T_{2k-1}, which happens with probability $\frac{2k-4}{2k-1}$, then p is rerouted through T_j from v_{2i}^1 to a vertex v_j^2 in L_2, and hence

$$E_d \geq \frac{2k-4}{2k-1} E_u + 1. \tag{4}$$

Putting together with (4) and Lemma 5 we obtain $E_d \in \Omega(k^2)$. ◀

We finally observe that any packet originated at a vertex of L_2 is routed downwards at least once before reaching the destination vertex, i.e., $E_2 \geq E_d = \Omega(k^2)$, which proves the following theorem.

▶ **Theorem 7.** *For any $k > 0$, there exists a $2k$ edge-connected graph, a set of $2k$ arc-disjoint spanning trees, and a set of $2k-1$ failed edges, such that the expected number of tree switches with* Rand-Algo *is $\Omega(k^2)$.*[1]

6.2 Correctness of Randomized-Bouncing Routing

In this section we prove that Rand-Bouncing-Algo eventually delivers a packet to d, i.e. it avoids loops, and in the next section we analyze its efficiency.

Assume that we, magically, know whether the arborescence we are routing along is a good one or not. Then, on a failed edge we could bounce if the arborescence is good, or switch to the next arborescence otherwise. And, we would not even need any randomness. However, we do not really know whether an arborescence is good or not since we do not know which edges will fail. To alleviate this lack of information we use a random guess. So, each time we hit a failed edge we take a guess that the arborescence is good, where the parameter q estimates this likelihood. Notice that Rand-Bouncing-Algo implements exactly this approach. As an example, consider Fig. 1. If a packet originated at a is first routed through `Red` and the corresponding outgoing edge $\{a, c\}$ is failed, then the packet is forwarded with probability q to `Blue` or `Green` chosen u.a.r., and with probability $1 - q$ it is bounced to `Green`, which shares the outgoing failed edge $\{a, c\}$ with `Red`. By the following lemma we show that this approach leads to $(k-1)$-resilient routing.

▶ **Lemma 8.** Rand-Bouncing-Algo *produces a set of $(k-1)$-resilient routing functions.*

Proof. By Lemma 4 we have that there exists at least one arborescence T_i of $\mathcal{T}$ such that bouncing on any failed edge of T_i the packet will reach d without any further interruption. Now, as on a failed edge algorithm Rand-Bouncing-Algo will switch to T_i with positive probability, and on a failed edge of T_i the algorithm will bounce with positive probability, we have that the algorithm will eventually reach d. ◀

[1] In the extended version of this paper [9], we show a more involved example for which Rand-Algo makes $\Omega(|V|k^2)$ hops, in expectation, to deliver a packet to d.

6.3 Number of Switches of Rand-Bouncing-Algo

In this subsection we analyze the expected number of times I the packet is rerouted from one arborescence to another one in Rand-Bouncing-Algo. As we are interested in providing an upper bound on I, we make the following assumptions. First, we assume that bouncing from an arborescence which is not good the routing always bounces to an arborescence which is not good as well. Second, we assume that only by bouncing from a good arborescence the routing will reach d without switching to any other arborescence. Third, we assume that there are exactly $k - f$ good arborescences, which is the lower bound provided by Lemma 1 and Lemma 3. Clearly, these assumptions can only lead to an increased number of iterations compared to the real case. Finally, for the sake of brevity we define $t := \frac{f}{k}$.

Now, we are ready to start with the analysis. As the first step we define a random variable, where in the definitions T is the arborescence variable from algorithm Rand-Bouncing-Algo,

$$X := \text{ number of times a failed edge is hit before reaching } d \text{ if routing on } T\,.$$

Let T_{init} be the first arborescence that we consider in Rand-Bouncing-Algo. Then, $\mathbb{E}[I]$ is upper-bounded by

$$\begin{aligned}\mathbb{E}[I] \leq &\Pr[T_{init} \text{ is not good}]\,\mathbb{E}[X|T_{init} \text{ is not good}] + \\ &\Pr[T_{init} \text{ is good}]\,\mathbb{E}[X|T_{init} \text{ is good}]\,,\end{aligned} \tag{5}$$

where from our assumptions we have

$$\Pr[T_{init} \text{ is not good}] = t, \text{ and } \Pr[T_{init} \text{ is good}] = 1 - t.$$

To simplify calculations, let X_P and Y_P be *pessimistic* upper bound on conditional expected values. That is, let X_P be the same as $\mathbb{E}[X|T_{init} \text{ is not good}]$ and Y_P as $\mathbb{E}[X|T_{init} \text{ is good}]$ under assumption that: the packet always hits a failed edge unless it bounces on a good arborescence; and, whenever packet bounces on a non-good arborescence it switches to a non-good one.

Now, let us express X_P and Y_P as functions in X_P, Y_P, q, and t, while following our assumptions. If T is not a good arborescence, then a routing along T will hit a failed edge. If it hits a failed edge, with probability $1 - q$ the routing will bounce and switch to a non good arborescence. With probability qt the routing scheme will set T to be a non good arborescence, and with probability $q(1 - t)$ it will set T to be a good arborescence. Formally, we have

$$X_P = 1 + qtX_P + q(1 - t)Y_P + (1 - q)X_P. \tag{6}$$

Applying an analogous reasoning about Y_P, we obtain

$$Y_P = 1 + qtX_P + q(1 - t)Y_P. \tag{7}$$

Observe that the equations describing X_P and Y_P differ only in the term $(1 - q)X_P$. This comes from the fact that bouncing on a good arborescences the packet will reach d without hitting any other failed edge.

By some simple calculations (see [9]), we obtain

$$\mathbb{E}[I] \leq U(q) := \frac{t}{(1 - q)q(1 - t)} + \frac{1}{1 - q}. \tag{8}$$

Now we can prove the following lemma.

▶ **Lemma 9.** *We have that*

$$\mathbb{E}[I] \leq 2 + 4\frac{t}{1-t} = 2 + 4\frac{f}{k-f}.$$

Proof. From (8) we have $\mathbb{E}[I] \leq U(q)$. Setting $q = 1/2$ we obtain

$$U(1/2) \leq 2 + 4\frac{t}{1-t},$$

and by plugging $t = f/k$ the lemma follows. ◀

Note that if we know f in advance, or have some guarantee in terms of an upper bound on f, we can derive parameter q that improves the running time of Rand-Bouncing-Algo, as provided by the following lemma.

▶ **Lemma 10.** *$U(q)$ is minimized for $q = q^* := 1 - (1 + \sqrt{t})^{-1}$, and equal to*

$$U(q^*) = \frac{1+\sqrt{t}}{1-\sqrt{t}}. \tag{9}$$

Proof. Consider $U(q)'$, which is

$$U(q)' = \frac{t(1-q)^2 - q^2}{(1-q)^2 q^2 (t-1)}.$$

In order to find the value of q that minimizes $U(q)$, denote it by q^*, we find the roots of $U(q)' = 0$ with respect to q. There is only one positive solution of equation $U(q)' = 0$, which is also the minimizer q^*, and is equal to $q^* = 1 - \frac{1}{1+\sqrt{t}}$, as desired.

Finally, substituting q^* into (8) and simplifying the expression we obtain (9). ◀

Observe that

$$U(q^*) \leq \frac{4}{1-\frac{f}{k}}.$$

Therefore, if $f = \alpha k$, i.e., only a fraction of the edges fail, we obtain $U(q^*) \leq \frac{4}{1-\alpha}$. This means that the expected number of arborescence switches does not depend on the number of failed edges but on the ratio between this number and the connectivity of the graph. Otherwise, if $f = k - 1$, we have that the expected number of arborescence switches is bounded by $4k$, which is linear w.r.t. to the connectivity of the graph. Combining these conclusions with Lemma 8, we obtain the following.

▶ **Theorem 11.** *Given a k-connected graph G, destination d and a decomposition of G into k arc-disjoint arborescences $\mathcal{T}$ rooted at d, there exists a $(k-1)$-resilient algorithm that delivers a packet to d after $O\left(\frac{k}{k-f}H\right)$ hops in expectation, where H is the length of a longest path of any arborescence of $\mathcal{T}$ and f the number of failed edges. The algorithm uses randomization only when encounters a failed edge. In particular, if $f = 0$, the algorithm is deterministic.*

6.4 An Extension: Rerouting in a Non-uniform Manner

In this section we briefly study non-uniform choice of arborescence used for rerouting in algorithm Rand-Bouncing-Algo. To motivate that discussion, consider a scenario in which a packet hits a failed edge u, v while routed along arborescence T. Wlog, assume $T = T_k$. Furthermore, assume that path v-d along every other arborescence does not contain any

failed link. Therefore, switching from T_k to any other arborescence the packet will reach d without any further interruption. If the packet is rerouted at step 2.2.(a) of algorithm Rand-Bouncing-Algo but not bounced, then the rerouting tree is chosen uniformly at random. It further means that the expected number of edges the packet will traverse before reaching d from v is

$$E_U = \sum_{i=1}^{k-1} \frac{dist_{T_i}(v)}{k-1} = \frac{\sum_{i=1}^{k-1} dist_{T_i}(v)}{k-1},$$

where $dist_{T_i}(a)$ is the number of the edges on the unique path from a to d along arborescence T_i.[2] However, the distances from v to d along different arborescences might significantly differ. This naturally suggests us to consider a non-uniform distribution of arborescences chosen at step 2.2.(a) of Rand-Bouncing-Algo, as we do in the rest of this section.

For each vertex $v \neq d$ and each arborescence T_i define probability p_v^i as

$$p_v^i := \frac{\frac{1}{dist_{T_i}(v)}}{\sum_{j=1}^{k-1} \frac{1}{dist_{T_j}(v)}}.$$

The expected number of the edges the packet will traverse if each arborescence is chosen with respect to the distribution given by p_v is

$$E_{NU} = \sum_{i=1}^{k-1} p_v^i dist_{T_i}(v) = \sum_{i=1}^{k-1} \frac{1}{\sum_{j=1}^{k-1} \frac{1}{dist_{T_j}(v)}} = \frac{k-1}{\sum_{i=1}^{k-1} \frac{1}{dist_{T_i}(v)}}.$$

Now we would like to show that indeed $E_U \stackrel{?}{\geq} E_{NU}$. But, it is the same as showing that

$$(k-1)^2 \stackrel{?}{\leq} \sum_{i=1}^{k-1} dist_{T_i}(v) \sum_{i=1}^{k-1} \frac{1}{dist_{T_i}(v)}.$$

However, the latter follows from Cauchy–Schwarz inequality as

$$(k-1)^2 = \left(\sum_{i=1}^{k-1} \sqrt{dist_{T_i}(v)} \sqrt{\frac{1}{dist_{T_i}(v)}} \right)^2 \leq \sum_{i=1}^{k-1} dist_{T_i}(v) \sum_{i=1}^{k-1} \frac{1}{dist_{T_i}(v)}.$$

Hence, $E_U \geq E_{NU}$, as advertised.

We note that this example is a potential scenario that might occur. However, and unfortunately, in case of failures we are unable to detect whether the described situation has occurred or not. Nevertheless, we believe that in practical applications the non-uniform choice of arborescences used for rerouting, as described above, would result in a more efficient routing than its uniform counterpart.

References

1 A. Atlas and A. Zinin. U-turn Alternates for IP/LDP Fast-Reroute. *IETF Internet draft version 03*, February 2006.

[2] As a remark, the best option in this scenario would be to reroute the packet along the arborescence T_i such that $i = \arg\min_i dist_{T_i}(v)$. Unfortunately, our model does not provide the information whether there is any failed edge on path v-d along T_i or not.

2 A. Atlas and A. Zinin. Basic Specification for IP Fast Reroute: Loop-Free Alternates. *IETF, RFC 5286*, 2008.

3 Roberto Beraldi. Biased random walks in uniform wireless networks. *Mobile Computing, IEEE Transactions on*, 8(4):500–513, 2009.

4 Anand Bhalgat, Ramesh Hariharan, Telikepalli Kavitha, and Debmalya Panigrahi. Fast Edge Splitting and Edmonds' Arborescence Construction for Unweighted Graphs. In *Proc. SODA*, pages 455–464, 2008. URL: `http://dl.acm.org/citation.cfm?id=1347082.1347132`.

5 Michael Borokhovich and Stefan Schmid. How (Not) to Shoot in Your Foot with SDN Local Fast Failover – A Load-Connectivity Tradeoff. In *OPODIS*, pages 68–82, 2013.

6 Graham Brightwell and Peter Winkler. Maximum hitting time for random walks on graphs. *Random Struct. Algorithms*, 1(3):263–276, October 1990. `doi:10.1002/rsa.3240010303`.

7 Costas Busch, Maurice Herlihy, and Roger Wattenhofer. Randomized greedy hot-potato routing. In *SODA*, pages 458–466, 2000.

8 Marco Chiesa, Ilya Nikolaevskiy, Slobodan Mitrović, Aurojit Panda, Andrei Gurtov, Aleksander Mądry, Michael Schapira, and Scott Shenker. The quest for resilient (static) forwarding tables. In *International Conference on Computer Communications (INFOCOM), 2016 IEEE*. IEEE, 2016.

9 Marco Chiesa, Ilya Nikolaevskiy, Aurojit Panda, Andrei Gurtov, Michael Schapira, and Scott Shenker. Exploring the limits of static failover routing. *CoRR*, abs/1409.0034, 2014. URL: `http://arxiv.org/abs/1409.0034`.

10 Jack Edmonds. Edge-disjoint branchings. *Combinatorial Algorithms*, pages 91–96, 1972.

11 Theodore Elhourani, Abishek Gopalan, and Srinivasan Ramasubramanian. IP Fast Rerouting for Multi-Link Failures. In *Proc. IEEE INFOCOM*, pages 2148–2156, 2014.

12 Gábor Enyedi, Gábor Rétvári, and Tibor Cinkler. A Novel Loop-free IP Fast Reroute Algorithm. In *Proc. EUNICE*, pages 111–119. Springer-Verlag, 2007.

13 Joan Feigenbaum, P. Brighten Godfrey, Aurojit Panda, Michael Schapira, Scott Shenker, and Ankit Singla. On the resilience of routing tables. In *Brief announcement PODC*, July 2012.

14 Eli M. Gafni and Dimitri P. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *IEEE Transactions on Communications*, 1981.

15 Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. *SIGCOMM Comput. Commun. Rev.*, 41(4):350–361, August 2011.

16 Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: a scalable and flexible data center network. In *ACM SIGCOMM computer communication review*, pages 51–62. ACM, 2009.

17 Mesut Günes, Martin Kähmer, and Imed Bouazizi. Ant-routing-algorithm (ara) for mobile multi-hop ad-hoc networks-new features and results. In *The second mediterranean workshop on ad-hoc networks*, 2003.

18 Kin-Wah Kwong, Lixin Gao, Roch Guérin, and Zhi-Li Zhang. On the Feasibility and Efficacy of Protection Routing in IP Networks. *IEEE/ACM Trans. Networking*, 19(5):1543–1556, October 2011.

19 K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica. Achieving convergence-free routing using failure-carrying packets. In *SIGCOMM*, 2007.

20 Junda Liu, Aurojit Panda, Ankit Singla, Brighten Godfrey, Michael Schapira, and Scott Shenker. Ensuring Connectivity via Data Plane Mechanisms. In *Proc. of NSDI*, pages 113–126, 2013.

21 Junda Liu, Baohua Yan, Scott Shenker, and Michael Schapira. Data-driven Network Connectivity. In *Proc. of HotNets*, pages 8:1–8:6, New York, NY, USA, 2011. ACM. doi:10.1145/2070562.2070570.

22 Srihari Nelakuditi, Sanghwan Lee, Yinzhe Yu, Zhi-Li Zhang, and Chen-Nee Chuah. Fast Local Rerouting for Handling Transient Link Failures. *IEEE/ACM Trans. Networking*, 15(2):359–372, April 2007.

23 Ping Pan, George Swallow, and Alia Atlas. Rfc 4090 fast reroute extensions to rsvp-te for lsp tunnels. *Internet Request for Comments*, page 42, 2005.

24 Gero Schollmeier, Joachim Charzinski, Andreas Kirstadter, Christoph Reichert, Karl J. Schrodi, Yuri Glickman, and Chris Winkler. Improving the Resilience in IP Networks. In *Proc. HPSR*, 2003.

25 FB Shepherd and PJ Winzer. Selective randomized load balancing and mesh networks with changing demands. *Journal of Optical Networking*, 5(5):320–339, 2006.

26 Brent Stephens, Alan L. Cox, and Scott Rixner. Plinko: Building Provably Resilient Forwarding Tables. In *Proc. of HotNets*, pages 26:1–26:7. ACM, 2013. doi:10.1145/2535771.2535774.

27 Leslie G. Valiant. A scheme for fast parallel communication. *SIAM journal on computing*, 11(2):350–361, 1982.

28 Junling Wang and Srihari Nelakuditi. IP Fast Reroute with Failure Inferencing. In *Proc. of SIGCOMM Workshop on Internet Network Management*, INM, pages 268–273, New York, NY, USA, 2007. ACM. doi:10.1145/1321753.1321764.

29 Baohua Yang, Junda Liu, Scott Shenker, Jun Li, and Kai Zheng. Keep Forwarding: Towards K-link Failure Resilient Routing. In *Proc. IEEE INFOCOM*, pages 1617–1625, 2014.

30 Baobao Zhang, Jianping Wu, and Jun Bi. RPFP: IP fast reroute with providing complete protection and without using tunnels. In *IWQoS*, pages 137–146, 2013.

31 Rui Zhang-Shen and Nick McKeown. Designing a fault-tolerant network using valiant load-balancing. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008.

32 Zifei Zhong, Srihari Nelakuditi, Yinzhe Yu, Sanghwan Lee, Junling Wang, and Chen nee Chuah. Failure Inferencing based Fast Rerouting for Handling Transient Link and Node Failures. In *Proc. IEEE INFOCOM*, 2005.

Partition Bound Is Quadratically Tight for Product Distributions*

Prahladh Harsha[†1], Rahul Jain[‡2], and Jaikumar Radhakrishnan[3]

1 **Tata Institute of Fundamental Research, Mumbai, India**
prahladh@tifr.res.in
2 **Center for Quantum Technologies, MajuLab and NUS, Singapore**
rahul@comp.nus.edu.sg
3 **Tata Institute of Fundamental Research, Mumbai, India**
jaikumar@tifr.res.in

Abstract

Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a 2-party function. For every product distribution μ on $\{0,1\}^n \times \{0,1\}^n$, we show that

$$\mathsf{CC}^{\mu}_{0.49}(f) = O\left(\left(\log \mathsf{prt}_{1/8}(f) \cdot \log\log \mathsf{prt}_{1/8}(f)\right)^2\right),$$

where $\mathsf{CC}^{\mu}_{\varepsilon}(f)$ is the distributional communication complexity of f with error at most ε under the distribution μ and $\mathsf{prt}_{1/8}(f)$ is the *partition bound* of f, as defined by Jain and Klauck [*Proc. 25th CCC*, 2010]. We also prove a similar bound in terms of $\mathsf{IC}_{1/8}(f)$, the *information complexity* of f, namely,

$$\mathsf{CC}^{\mu}_{0.49}(f) = O\left(\left(\mathsf{IC}_{1/8}(f) \cdot \log \mathsf{IC}_{1/8}(f)\right)^2\right).$$

The latter bound was recently and independently established by Kol [*Proc. 48th STOC*, 2016] using a different technique.

We show a similar result for query complexity under product distributions. Let $g : \{0,1\}^n \to \{0,1\}$ be a function. For every bit-wise product distribution μ on $\{0,1\}^n$, we show that

$$\mathsf{QC}^{\mu}_{0.49}(g) = O\left(\left(\log \mathsf{qprt}_{1/8}(g) \cdot \log\log \mathsf{qprt}_{1/8}(g)\right)^2\right),$$

where $\mathsf{QC}^{\mu}_{\varepsilon}(g)$ is the distributional query complexity of f with error at most ε under the distribution μ and $\mathsf{qprt}_{1/8}(g))$ is the *query partition bound* of the function g.

Partition bounds were introduced (in both communication complexity and query complexity models) to provide LP-based lower bounds for randomized communication complexity and randomized query complexity. Our results demonstrate that these lower bounds are polynomially tight for *product* distributions.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes, G.2.1 Combinatorics

Keywords and phrases partition bound, product distribution, communication complexity, query complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.135

* A full version of the paper is available at http://arxiv.org/abs/1512.01968 [8]. This work was done when the three authors were visiting the Simons Institute for the Theory of Computing, Berkeley, USA.

† Prahladh Harsha's research supported in part by ISF-UGC grant 1399/4 and Google India Fellowship.
‡ Rahul Jain's research partly supported by the Singapore Ministry of Education via Academic Research Fund Tier 3 MOE2012-T3-1-009 and Young Researcher Award, National University of Singapore.

1 Introduction

Over the last decade, several lower bound techniques using linear programming formulations and information complexity methods have been developed for problems in communication complexity and query complexity. One of the central questions in communication complexity is to understand the tightness of these lower bound techniques. For instance, over the last few years, considerable effort has gone into understanding the *information complexity* measure. Informally speaking, (internal) information complexity is the amount of information the two parties reveal to each other about their respective inputs while computing the joint function. It is known that for product distributions, the internal information complexity not only lower bounds but also upper bounds the distributional communication complexity (up to logarithmic multiplicative factors in the communication complexity) [1]. On the other hand, recent works due to Ganor, Kol and Raz [3, 4, 5] show that there exist non-product distributions which exhibit exponential separation between internal information complexity and distributional communication complexity[1]. However, it is still open if internal information complexity (or a polynomial of it) upper bounds the public-coin randomized communication complexity (up to logarithmic multiplicative factors in the input size) [2].

Jain and Klauck [9], using tools from linear programming, gave a uniform treatment of several of the existing lower bound techniques and proposed the *partition bound*. This leads to following related (but incomparable) conjecture: does a polynomial of the partition bound yield an upper bound on the communication complexity? We are not aware of any counterexample to this conjecture[2].

We consider these questions when the inputs to Alice and Bob are drawn from a product distribution and show the following.

▶ **Theorem 1.** *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and let $\mathsf{IC}_\varepsilon(f)$ and $\mathsf{prt}_\varepsilon(f)$ be the information complexity and partition bound respectively of f with error at most ε. For a product distribution μ on $\{0,1\}^n \times \{0,1\}^n$, the distributional communication complexity of f under distribution μ with error at most 0.49, denoted by $\mathsf{CC}^\mu_{0.49}(f)$, can be bounded above as follows:*

$$\mathsf{CC}^\mu_{0.49}(f) = O\left(\left(\mathsf{IC}_{1/8}(f) \cdot \log \mathsf{IC}_{1/8}(f)\right)^2\right), \tag{1.1}$$

$$\mathsf{CC}^\mu_{0.49}(f) = O\left(\left(\log \mathsf{prt}_{1/8}(f) \cdot \log\log \mathsf{prt}_{1/8}(f)\right)^2\right). \tag{1.2}$$

Our technique yields bounds more general than those stated above (see discussion after Proposition 7 for this generalization). We remark that recently (and independently of this work) Kol [11] obtained the bound (1.1) using very different techniques. Kol's result is stronger in the sense that her bound is in terms of the information complexity $\mathsf{IC}^\mu(f)$ for the product distribution μ, while our result is in terms of the worst case information complexity $\mathsf{IC}(f)$ (note, $\mathsf{IC}_\varepsilon(f) = \max_\mu \mathsf{IC}^\mu_\varepsilon(f)$). In fact, Kol showed that

$$\mathsf{CC}^\mu_{\delta+\varepsilon}(f) = O\left(\mathsf{IC}^\mu_\delta(f)^2 \cdot \operatorname{poly}\log \mathsf{IC}^\mu_\delta(f)/\varepsilon^5\right), \tag{1.3}$$

and concluded that

$$\mathsf{CC}^\mu_{0.49}(f) = O\left(\mathsf{IC}_{1/8}(f)^2 \cdot \operatorname{poly}\log \mathsf{IC}_{1/8}(f)\right). \tag{1.4}$$

[1] The third result of Ganor, Kol and Raz [5] actually demonstrates an exponential separation between external information and communication complexity, albeit not for computing a Boolean function.

[2] The recent work of Göös et al. [6] demonstrates the existence of a total function for which the partition bound is strictly sublinear in the randomized communication complexity. This still does not rule out communication complexity being bound by a polynomial of the partition bound.

Kol's result (1.3) is incomparable to our second result in terms of partition bound (1.2).

We consider a similar question in query complexity and show the following.

▶ **Theorem 2.** *Let $g : \{0,1\}^n \to \{0,1\}$ be a function and μ be a bit-wise product distribution on $\{0,1\}^n$. Let $\mathsf{qprt}_\varepsilon(g)$ be the query partition bound for g with error ε. Then, the distributional query complexity with error at most 0.49 under the distribution μ, denoted by $\mathsf{QC}^\mu_{0.49}(f)$, can be bounded above as follows:*

$$\mathsf{QC}^\mu_{0.49}(g) = O\left(\left(\log \mathsf{qprt}_{1/8}(g) \cdot \log\log \mathsf{qprt}_{1/8}(g)\right)^2\right).$$

A similar quadratic upper bound for query complexity for product distributions in terms of approximate certificate complexity was obtained by Smyth [14]. His proof uses Reimer's inequality while our proof technique is based on Nisan and Wigderson's [13] more elementary approach.

Organization. The communication complexity result is proven in §2 while the query complexity result is deferred to the full version [8] for lack of space.

2 Communication Complexity

2.1 Preliminaries

We work in Yao's two-party communication model [15] (see Kushilevitz and Nisan [12] for an excellent introduction to the area). Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$ be finite non-empty sets, and let $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a function. A two-party protocol for computing f consists of two parties, Alice and Bob, who get inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively, and exchange messages in order to compute $f(x,y) \in \mathcal{Z}$ (using shared randomness).

For a distribution μ on $\mathcal{X} \times \mathcal{Y}$, let the ε-error distributional communication complexity of f under μ (denoted by $\mathsf{CC}^\mu_\varepsilon(f)$), be the number of bits communicated (for the worst-case input) by the best deterministic protocol for f with average error at most ε under μ. Let $\mathsf{CC}^{\mathrm{pub}}_\varepsilon(f)$, the public-coin randomized communication complexity of f with worst case error ε, be the number of bits communicated (for the worst-case input) by the best public-coin randomized protocol that for each input (x,y) computes $f(x,y)$ correctly with probability at least $1-\varepsilon$. Randomized and distributional complexity are related by the following special case of von Neumann's minmax principle.

▶ **Theorem 3** (Yao's minmax principle [16])**.** $\mathsf{CC}^{\mathrm{pub}}_\varepsilon(f) = \max_\mu \mathsf{CC}^\mu_\varepsilon(f)$.

We will prove Theorem 1 by first showing an upper bound on communication complexity in terms of the smooth rectangle bound and then observing that the smooth rectangle bound is bounded above by the partition bound.

Smooth rectangle bound

The smooth rectangle bound was introduced by Jain and Klauck [9] as a generalization of the rectangle bound. Just like the rectangle bound, the smooth rectangle bound also provides a lower bound for randomized communication complexity. Informally, the smooth rectangle bound for a function f under a distribution μ, is the maximum over all functions g , which are close to f under the distribution μ, of the rectangle bound of g. However, it will be more convenient for us to work with the following linear programming formulation. (See [9,

Lemma 2] and [10, Lemma 6] for the relations between the LP formulation and the more "natural" formulation in terms of rectangle bound.) It is evident from the LP formulation that the smooth rectangle bound is a further relaxation of the partition bound (defined in the appendix). We will formulate our results in terms of a distributional version of the above smooth rectangle bound. For $\mu : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and any $z \in \mathcal{Z}$ and rectangle R, let $\mu_z(R) := \mu(R \cap f^{-1}(z))$ and $\mu_{\bar{z}}(R) := \mu(R) - \mu_z(R)$. Furthermore, let $\mu_z := \mu_z(\mathcal{X} \times \mathcal{Y})$ and $\mu_{\bar{z}} := \mu_{\bar{z}}(\mathcal{X} \times \mathcal{Y})$. The smooth rectangle and its distributional version are defined below.

▶ **Definition 4** (Smooth rectangle bound)**.**

- For a function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ and $\varepsilon \in (0,1)$, the (ε,δ)-smooth rectangle bound of f denoted $\mathsf{srec}_{\varepsilon,\delta}(f)$ is defined to be $\max\{\mathsf{srec}^z_{\varepsilon,\delta}(f) : z \in \mathcal{Z}\}$, where $\mathsf{srec}^z_{\varepsilon,\delta}(f)$ is the optimal value of the following linear program.
- For a distribution μ on $\mathcal{X} \times \mathcal{Y}$ and function $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, the (ε,δ)-smooth rectangle bound of f with respect to μ denoted $\mathsf{srec}^\mu_{\varepsilon,\delta}(f)$ is defined to be $\max\{\mathsf{srec}^{z,\mu}_{\varepsilon,\delta}(f) : z \in \mathcal{Z}\}$, where $\mathsf{srec}^{z,\mu}_{\varepsilon,\delta}(f)$ is the optimal value of the following linear program.

$\underline{\mathsf{srec}^z_{\varepsilon,\delta}(f)}$

$$\begin{aligned}
\min \quad & \sum_R w_R \\
& \sum_{R \ni (x,y)} w_R \geq 1-\varepsilon, \quad \forall (x,y) \in f^{-1}(z) \\
& \sum_{R \ni (x,y)} w_R \leq \delta, \quad \forall (x,y) \notin f^{-1}(z) \\
& \sum_{R \ni (x,y)} w_R \leq 1, \quad \forall (x,y) \\
& w_R \geq 0, \quad \forall R \ .
\end{aligned}$$

$\underline{\mathsf{srec}^{z,\mu}_{\varepsilon,\delta}(f)}$

$$\begin{aligned}
\min \quad & \sum_R w_R \\
& \sum_{(x,y) \in f^{-1}(z)} \mu_{x,y} \sum_{R \ni (x,y)} w_R \geq (1-\varepsilon) \cdot \mu_z \qquad (2.1) \\
& \sum_{R \ni (x,y)} w_R \leq \delta, \quad \forall (x,y) \notin f^{-1}(z) \qquad (2.2) \\
& \sum_{R \ni (x,y)} w_R \leq 1, \quad \forall (x,y) \qquad (2.3) \\
& w_R \geq 0, \quad \forall R \ .
\end{aligned}$$

We will refer to the constraint in (2.1) as the covering constraint and the ones in (2.2) as the packing constraints. Note that while there is a single covering constraint (averaged over all the inputs (x,y) that satisfy $f(x,y) = z$) there are packing constraints corresponding to each $(x,y) \notin f^{-1}(z)$.

Similar to Yao's minmax principle Theorem 3, we have the following proposition relating the distributional version of the smooth rectangle bound to the smooth rectangle bound.

▶ **Proposition 5.** $\mathsf{srec}_{\varepsilon,\delta}(f) = \max_\mu \mathsf{srec}^\mu_{\varepsilon,\delta}(f)$.

The main result of this section is the following

▶ **Theorem 6.** *For any Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and any* product *distribution μ on $\{0,1\}^n \times \{0,1\}^n$, we have the following.*

1. $\mathsf{CC}^\mu_{0.49}(f) = O\left((\log \mathsf{srec}^\mu_{1/n^2,1/n^2}(f))^2 \cdot \log n\right)$.
2. *Furthermore, if there exists $k \geq 20$ such that*

$$\lceil 100 \log \mathsf{srec}^\mu_{\delta,\delta}(f) \rceil \leq k,$$

for $\delta \leq 1/(30 \cdot 100(k+1)^4)$, then

$$\mathsf{CC}^\mu_{0.49}(f) = O(k^2).$$

The above theorem is useful only when we have a upper bound on the smooth rectangle bound for very small δ. The following proposition shows that such upper bounds for smooth rectangle bound for such small δ can be obtained in terms of either the information complexity or the partition bound.

▶ **Proposition 7.** *For any Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and any $\delta \in (0,1)$, we have the following bounds on $\mathsf{srec}_{\delta,\delta}(f)$:*

$$\log \mathsf{srec}_{\delta,\delta}(f) \leq O\left(\log \frac{1}{\delta}\right) \cdot \mathsf{IC}_{1/8}(f),$$
$$\log \mathsf{srec}_{\delta,\delta}(f) \leq O\left(\log \frac{1}{\delta}\right) \cdot \log \mathsf{prt}_{1/8}(f).$$

(This proposition depends on the error-reduction properties of information complexity and partition bound; a proof appears in the full version [8].) Using this proposition, we can reduce the error (i.e., δ) to $1/n^2$ and show that $\mathsf{CC}^{\mu}_{0.49}(f) = O\left(\left(\log \mathsf{prt}_{1/8}(f)\right)^2 \cdot (\log n)^3\right)$. However, we can also reduce the error to $1/\mathrm{poly}(\log \mathsf{prt}_{1/8}(f))$ and show that there exists a $k = O\left(\log \mathsf{prt}_{1/8}(f) \cdot \log\log \mathsf{prt}_{1/8}(f)\right)$ that satisfies the hypothesis for the second part of Theorem 6. The bound (1.2) in Theorem 1 now follows by combining Propositions 7 and 5 and Theorem 6. A similar argument yields the bound (1.1).

In particular, the above discussion shows that our techniques apply to any complexity measure (not necessarily partition bound and information complexity) which can be used to bound the smooth rectangle bound for very small δ. An interesting question that arises in this context is if we could bound smooth rectangle bound for small δ in terms of smooth rectangle bound for large δ, say $\delta = 1/3$ (i.e., is error-reduction for srec feasible?). This question was answered in the negative for partial functions by Göös et al. [7] who show that there exists a partial function f that has $\mathsf{srec}_{1/3}(f) = O(\log n)$ and yet $\mathsf{srec}_{1/4}(f) = \Omega(n)$.

2.2 Proof of Theorem 6

In this section, we construct a communication protocol tree with a small number of leaves from the optimal solutions to the LPs corresponding to $\mathsf{srec}^{0,\mu}_{\varepsilon,\delta}$ and $\mathsf{srec}^{1,\mu}_{\varepsilon,\delta}$. The construction of the protocol tree with a small number of leaves is inspired by a construction due to Nisan and Wigderson, in the context of log-rank conjecture [13, Theorem 2] (see also [12, Combinatorial proof of Theorem 2.11]). Unlike the earlier constructions, our protocol works for a distribution and allows for error. As a result, the decomposition into sub-problems needs to be performed more carefully. This step critically uses the product nature of the distribution μ.

The decomposition is accomplished using an inductive argument. We will work with the quantity $\mathsf{srec}^0 + \mathsf{srec}^1$. That is, we will show that if this sum is small, then there is a protocol with few leaves. Suppose $\mathsf{srec}^0 \leq \mathsf{srec}^1$. Since srec^0 is small, we will conclude that there is a large rectangle biased towards 0 (see Lemma 8). Based on this large rectangle, the entire communication matrix is partitioned into three parts: (1) the large biased rectangle itself, (2) a rectangle whose corresponding sub-problem admits an LP solution leading to a smaller srec^1 value (the underlying product nature of the distribution μ is used here) and (3) a rectangle where the total measure with respect to μ drops significantly (see Lemma 9).

We say that a rectangle R is $(1-\alpha)$-biased towards to 0 if $\mu_1(R) \leq \alpha\mu_0(R)$.

▶ **Lemma 8** (large biased rectangle)**.** *Let μ be a product distribution. If $\mathsf{srec}^{0,\mu}_{\varepsilon,\delta}(f) \le D$, then for every $\rho \in (0,1)$ there exists a rectangle S such that S is $(1-\rho)$-biased towards 0 and*

$$\mu(S) \ge \mu_0(S) \ge \frac{1}{D} \cdot \left((1-\varepsilon) \cdot \mu_0 - \left(\frac{\delta}{\rho}\right) \cdot \mu_1 \right).$$

(The proof appears in §2.3.) We will apply the above lemma with $\rho = \sqrt{\delta}$ and conclude that there exists a large rectangle $S = X_0 \times Y_0$ that is $(1-\sqrt{\delta})$-biased towards 0. Let $X_1 = \mathcal{X} \setminus X_0$ and $Y_1 = \mathcal{Y} \setminus Y_0$. For $i,j \in \{0,1\}$, define rectangles $R^{(ij)} := X_i \times Y_j$, $R^{(1*)} := X_1 \times \mathcal{Y}$, and $R^{(*1)} := \mathcal{X} \times Y_1$. (Note, $S = R^{(00)}$.) For $i,j \in \{0,1,*\}$, let $\mu^{(ij)}$ be the restriction of μ to the rectangle $R^{(ij)}$. We show in the lemma below that the function f when restricted to either $R^{(10)}$ or $R^{(01)}$ has the property that the corresponding srec^1 drops by a constant factor. Define

$$\varepsilon(f) := 1 - \frac{\left(\sum_{(x,y)\in f^{-1}(1)} \mu_{x,y} \sum_{R:(x,y)\in R} w_R\right)}{\mu_1},$$

$$\varepsilon^{(ij)}(f) := 1 - \frac{\left(\sum_{(x,y)\in f^{-1}(1)\cap R^{(ij)}} \mu_{x,y} \sum_{R:(x,y)\in R} w_R\right)}{\mu_1(R^{(ij)})}; \qquad \text{for } i,j \in \{0,1\}.$$

It follows from the covering constraint that $\varepsilon(f) \le \varepsilon$. Furthermore, $\varepsilon(f)$ is an average of the $\varepsilon^{(ij)}$'s in the sense that $\varepsilon(f) = \left(\sum_{i,j\in\{0,1\}} \mu_1(R^{(ij)})\varepsilon^{(ij)}\right)/\mu_1$.

▶ **Lemma 9.** *Suppose the product distribution μ and rectangles $R^{(ij)}$ are as above; in particular, $R^{(00)}$ is $(1-\sqrt{\delta})$-biased towards 0. There exists an $(ij) \in \{(01),(10)\}$ such that one of the following holds: (a) $2\mu^{(ij)}(f^{-1}(1)) \le \mu^{(ij)}(f^{-1}(0))$ or (b) $\mathsf{srec}^{1,\mu^{(ij)}}_{\varepsilon^{(ij)}+30\sqrt[4]{\delta},\delta}(f) \le 0.9D$ where $\varepsilon^{(ij)}$ is as defined above.*

We will prove this lemma in §2.3. Let us assume the above lemmas and obtain the low cost communication protocol claimed in Theorem 6.

Suppose $\mu^{(01)}$ satisfies $\mathsf{srec}^{1,\mu^{(01)}}_{\varepsilon^{(01)}+30\sqrt[4]{\delta},\delta}(f) \le 0.9D$ as given by the above lemma. Consider the decomposition of the space $\mathcal{X}\times\mathcal{Y}$ given by $(R^{(00)}, R^{(01)}, R^{(1*)} = R^{(10)} \cup R^{(11)})$. We note that $R^{(00)}$ is a large biased rectangle, $R^{(01)}$ has lower srec^1 value while $R^{(1*)}$ has lower μ value (since $R^{(00)}$ is large) and its srec values are no larger than that of the entire space. In the case when $\mu^{(10)}$ satisfies $\mathsf{srec}^{1,\mu^{(10)}}_{\varepsilon^{(10)}+30\sqrt[4]{\delta},\delta}(f) \le 0.9D$, we similarly have the decomposition $(R^{(00)}, R^{(10)}, R^{(*1)} = R^{(01)} \cup R^{(11)})$.

This suggests a natural inductive protocol Π for f that we formalize in the lemma below.

For our induction it will be convenient to work with μ that are not necessarily normalized. So, we will only assume $\mu : \mathcal{X}\times\mathcal{Y} \to [0,1]$ but not that $|\mu| := \mu(\mathcal{X}\times\mathcal{Y}) = \sum_{(x,y)\in\mathcal{X}\times\mathcal{Y}} \mu(x,y) = 1$. For a protocol Π, let the advantage of Π be defined by

$$\mathsf{adv}_\mu(\Pi) = \sum_{(x,y):f(x,y)=\Pi(x,y)} \mu(x,y) - \sum_{(x,y):f(x,y)\neq\Pi(x,y)} \mu(x,y).$$

Let $L(\Pi)$ be the number of leaves in Π.

We now formulate the induction hypothesis as follows.

▶ **Lemma 10.** *Fix a function $f : \mathcal{X}\times\mathcal{Y} \to \{0,1\}$ and a product distribution (not necessarily normalized) $\mu : \mathcal{X}\times\mathcal{Y} \to [0,1]$ such that $|\mu| \ge 0$. Let $\varepsilon,\delta \in (0,1)$ and $\Delta \in (0,|\mu|)$. Let s,t be*

non-negative integers such that

$$s \geq s(\mu,\varepsilon,\delta) := \left\lceil 100 \cdot \log 2(\mathsf{srec}^{0,\mu}_{\varepsilon,\delta}(f) + \mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f)) \right\rceil;$$
$$t \geq t(\mu,\varepsilon,\delta) := \lceil 100 \cdot 2^s \log(|\mu|/\Delta) \rceil.$$

Then, there is a protocol Π *such that*

$$L(\Pi) \leq 4\binom{s+t}{t} - 1; \tag{2.4}$$
$$\mathsf{adv}_\mu(\Pi) \geq \left(\frac{1}{10} - \varepsilon - 30(s+1)\sqrt[4]{\delta}\right)|\mu| - \Delta \cdot L(\Pi). \tag{2.5}$$

▶ Remark. Since $\varepsilon \leq \frac{1}{2}$, our definitions imply that $\mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f) + \mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f)) \geq \frac{1}{2}$; thus $s \geq 0$. Similarly, since $\Delta \leq |\mu|$, we have $t \geq 0$.

Proof. First, we observe that if $\max\{\mu_0,\mu_1\} \geq 2\min\{\mu_0,\mu_1\}$, then the protocol Π consisting of just one leaf, with the most popular value as label, meets the requirements: for, $\mathsf{adv}_\mu(\Pi) \geq \frac{1}{3}|\mu|$ and $L(\Pi) = 1$, and our claim holds. Also, we may assume that $\varepsilon - 30(s+1)\sqrt[4]{\delta} < \frac{1}{10}$, for otherwise the claim is trivially true.

We now proceed by induction on $s+t$, assuming that μ is balanced: $\max\{\mu_0,\mu_1\} \leq 2\min\{\mu_0,\mu_1\}$.

Base case ($s = 0$)

Since $s = 0$, we have $\log \mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f) \leq \frac{1}{100}$. We will show a protocol Π where Alice sends one bit after which Bob announces the answer. Consider the optimal solution $\langle w_R : R \text{ a rectangle}\rangle$ to the LP corresponding to $\mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f)$; thus, $\mathsf{OPT} := \sum_R w_R = \mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f) \leq 2^{1/100} \leq 2$. Let $R = R_X \times R_Y$ be a random rectangle picked with probability proportional to w_R (using public coins). In the protocol Π, Alice tells Bob if $x \in R_X$, and Bob returns the answer 1 if $(x,y) \in R_Y$ and returns 0 otherwise. Let $p_{xy} := \Pr_R[(x,y) \in R]$. Then, by (2.1) we have $\sum_{(x,y)\in f^{-1}(1)} \mu(x,y)p_{xy} \geq (1-\varepsilon)\mu_1/\mathsf{OPT}$, and by (2.2), we have $\sum_{(x,y)\in f^{-1}(0)} \mu(x,y)p_{xy} \leq \delta\mu_0/\mathsf{OPT}$. Thus,

$$\begin{aligned}
\mathbb{E}_R\left[\sum_{(x,y):\Pi(x,y)\neq f(x,y)} \mu(x,y)\right] &= \sum_{(x,y)\in f^{-1}(1)} \mu(x,y)(1-p_{xy}) + \sum_{(x,y)\in f^{-1}(0)} \mu(x,y)p_{x,y} \\
&\leq \mu_1 - (1-\varepsilon)\mu_1/\mathsf{OPT} + \delta\mu_0/\mathsf{OPT} \\
&\leq \mu_1 - ((1-\varepsilon)\mu_1 - \delta\mu_0)/\mathsf{OPT} \\
&\leq \frac{1}{2}(\mu_1 + \varepsilon\mu_1 + \delta\mu_0) \qquad (\text{since } \mathsf{OPT} \leq 2).
\end{aligned} \tag{2.6}$$

Fix a choice R for which the quantity under the expectation is at most $\frac{1}{2}(\mu_1 + \varepsilon\mu_1 + \delta\mu_0)$. Then,

$$\begin{aligned}
\mathsf{adv}(\Pi) &= |\mu| - 2\sum_{(x,y):\Pi(x,y)\neq f(x,y)} \mu(x,y) \\
&\geq |\mu| - (\mu_1 + \varepsilon\mu_1 + \delta\mu_0) \\
&\geq \left(\frac{1}{3} - \varepsilon - \delta\right)|\mu| \qquad (\text{since } \mu_1 \leq 2\mu_0).
\end{aligned}$$

Base case ($t = 0$)

In this case, $|\mu| = \Delta$, and the protocol Π with a single leaf that gives the most probable answer achieves $\mathsf{adv}(\Pi) \geq 0 \geq |\mu| - \Delta$.

Induction step

We will use Lemma 8 to decompose the communication matrix into a small number of rectangles. After an exchange of a few bits to determine in which rectangle the input lies, Alice and Bob will be left with a problem for which s or t is significantly smaller. Assume $\mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f) \geq \mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f)$; in particular, $\mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f) \leq 2^{s/100}$.

Formally, from Lemma 8 (taking $\rho = \sqrt{\delta}$), we obtain a rectangle $R^{(00)} = X_0 \times Y_0$ such that (a) $R^{(00)}$ is $(1-\sqrt{\delta})$-biased towards 0, and (b) $\mu(R^{(00)}) \geq \frac{1}{2^{s/100}}(1-\varepsilon-2\sqrt{\delta})|\mu_0| \geq \frac{1}{3\cdot 2^{s/100}}(1-\varepsilon-2\sqrt{\delta})|\mu|$. Recall the definitions of the rectangles $R^{(10)}, R^{(01)}, R^{(11)}, R^{(1*)}, R^{(*1)}$ and the corresponding restrictions of μ, namely, $\mu^{(01)}, \mu^{(10)}, \mu^{(11)}, \mu^{(1*)}, \mu^{(*1)}$. Suppose the choice of ij in Lemma 9 for which one of the alternatives holds is $ij = 01$ (the other case $ij = 10$ is symmetric). The protocol Π proceeds as follows. Alice starts by telling Bob if $x \in X_0$.

Alice says $x \in X_0$. Now, Bob tells Alice if $y \in Y_0$.

Bob says $y \in Y_0$. The protocol $\Pi^{(00)}$ in this case has one leaf with answer 0; thus $\mathsf{adv}(\Pi^{(00)}) \geq |\mu^{(00)}| \cdot (1-\sqrt{\delta})$.

Bob says $y \notin Y_0$. Alice and Bob follow the protocol $\Pi^{(01)}$ promised by induction for $R^{(01)}$ under $\mu^{(01)}$. To bound the number of leaves in $\Pi^{(01)}$, we will consider the two alternatives ((a) and (b)) specified in Lemma 9 separately. First (alternative (a)) suppose $2\mu^{(01)}(f^{-1}(1)) \leq \mu^{(01)}(f^{-1}(0))$; then we immediately declare 0 as the response, so that $L(\Pi^{(01)}) = 1$ and $\mathsf{adv}(\Pi^{(01)}) \geq |\mu^{(01)}|/3$. If alternative (b) holds, then we have

$$\mathsf{srec}^{1,\mu^{(01)}}_{\varepsilon^{(01)}+30\sqrt[4]{\delta},\delta}(f) \leq 0.9\mathsf{srec}^{1,\mu}_{\varepsilon,\delta}(f). \tag{2.7}$$

Then, we obtain $\Pi^{(01)}$ by induction. We take $\varepsilon^{(01)} + 30\sqrt[4]{\delta}$ as ε (if this quantity is greater than 1, then we use a trivial protocol with one leaf and zero advantage). With the reduction promised in (2.7), we may use a value of s that is the old s minus 1. Thus, we have

$$L(\Pi^{(01)}) \leq 4\binom{(s-1)+t}{t} - 1;$$
$$\mathsf{adv}(\Pi^{(01)}) \geq |\mu^{(01)}| \cdot \left(\frac{1}{10} - (\varepsilon^{(01)} + 30\sqrt[4]{\delta}) - 30s\sqrt[4]{\delta}\right) - \Delta \cdot L(\Pi^{(01)}).$$

Alice says $x \notin X_0$. Alice and Bob follow the protocol $\Pi^{(1*)}$ obtained by applying the induction hypothesis to the rectangle $R^{(1*)}$ and the associated distribution $\mu^{(1*)}$. Observe that

$$|\mu^{(1*)}| \leq |\mu| - \mu(R^{(00)}) \leq |\mu|\left(1 - \frac{1}{3\cdot 2^{s/100}}(1-\varepsilon-2\sqrt{\delta})\right) \leq |\mu|\left(1-\frac{1}{4\cdot 2^s}\right). \tag{2.8}$$

For the last inequality we used $\varepsilon + 2\sqrt{\delta} \leq \frac{1}{10}$, for otherwise (2.5) holds trivially. Now, (2.8) implies that $\log|\mu^{(1*)}| \leq \log|\mu| - \frac{1}{1002^s}$; so, for our induction we may take $t \leftarrow t-1$. The parameters ε, δ and Δ remain the same. The original LP solutions are still valid for

the subproblem, so we use the same s. The protocol $\Pi^{(1*)}$ obtained by induction satisfies the following inequalities.

$$L(\Pi^{(1*)}) \leq 4\binom{s+(t-1)}{t-1} - 1;$$
$$\mathsf{adv}(\Pi^{(1*)}) \geq |\mu^{(1*)}| \cdot \left(\frac{1}{10} - \varepsilon^{(1*)} - 30(s+1)\sqrt[4]{\delta}\right) - \Delta \cdot L(\Pi^{(1*)}).$$

Putting all the contributions together, we obtain

$$\begin{aligned}
L(\Pi) &= 1 + L(\Pi^{(01)}) + L(\Pi^{(1*)}) \\
&\leq 1 + \left(4\binom{(s-1)+t}{t} - 1\right) + \left(4\binom{s+(t-1)}{t-1} - 1\right) \\
&= 4\binom{s+t}{t} - 1; \\
\mathsf{adv}(\Pi) &\geq |\mu^{(00)}| \cdot (1 - \sqrt{\delta}) \\
&\quad + |\mu^{(01)}| \cdot \left(\frac{1}{10} - (\varepsilon^{(01)} + 30\sqrt[4]{\delta}) - 30s\sqrt[4]{\delta}\right) - \Delta \cdot L(\Pi^{(01)}) \\
&\quad + |\mu^{(1*)}| \cdot \left(\frac{1}{10} - \varepsilon^{(1*)} - 30(s+1)\sqrt[4]{\delta}\right) - \Delta \cdot L(\Pi^{(1*)}) \\
&\geq \left(\frac{1}{10} - \varepsilon - 30(s+1)\sqrt[4]{\delta}\right)|\mu| - \Delta \cdot L(\Pi).
\end{aligned}$$

◀

The above lemma yields a protocol whose protocol tree has a small number of leaves, but not necessarily small depth. We can balance the protocol tree using the following proposition.

▶ **Proposition 11** ([12, Lemma 2.8]). *If f has a deterministic communication protocol tree with ℓ leaves, then f has a protocol tree with depth at most $O(\log \ell)$.*

We are now in a position to complete the proof of the main theorem of this section.

Proof of Theorem 6. To prove the first part of Theorem 6, we invoke Lemma 10 with $\Delta = 1/2^{4n}$ and $\varepsilon = \delta = 1/n^2$ to derive a protocol tree Π with at most

$$L(\Pi) = n^{O\left(\log \mathsf{srec}^{1,\mu}_{1/n^2,1/n^2}(f)\right)^2}$$

leaves and advantage at least $1/20$. The first part now follows from Proposition 11.

To prove the second part of Theorem 6, we invoke Lemma 10 with $s = k$, $\Delta = 1/2^{5k^2}$ and $\varepsilon = \delta = 1/(30 \cdot 100(k+1)^4)$ where k satisfies the hypothesis. With this setting of parameters $t = \lceil 500 \cdot 2^k k^2 \rceil \leq 2^{2k}$ (for $k \geq 20$). Lemma 10 implies a protocol tree Π with at most

$$L(\Pi) \leq (t+s)^s \leq t^{2s} \leq 2^{4k^2}$$

leaves and advantage at most $1/20$. The second claim then follows from Proposition 11. ◀

2.3 Proofs of Lemmas 8–9

Proof of Lemma 8. Fix $z \in \{0,1\}$. In the following we say that a rectangle R is *biased* (towards 0) if $\mu_1(R) \leq \rho \cdot \mu_0(R)$; otherwise, we say it is unbiased. Fix a solution $\langle w_R :$

R is a rectangle$\rangle$ that achieves the optimum $\mathsf{srec}^{0,\mu}_{\varepsilon,\delta}(f) \le D$. It follows

$$\begin{aligned}\sum_{R:\text{unbiased}} w_R \cdot \mu_0(R) &\le \sum_{R:\text{unbiased}} w_R \cdot \frac{\mu_1(R)}{\rho} \\ &\le \frac{1}{\rho} \cdot \sum_R w_R \cdot \mu_1(R) \\ &= \frac{1}{\rho} \sum_{(x,y)\in f^{-1}(1)} \mu(x,y) \sum_{R:(x,y)\in R} w_R \\ &\le \frac{\delta}{\rho} \cdot \mu_1,\end{aligned}$$

where the last inequality follows from the packing constraints (2.2). We now use the covering constraints (2.1) to conclude that

$$\sum_{R:\text{biased}} w_R \cdot \mu_0(R) = \sum_R w_R \cdot \mu_0(R) - \sum_{R:\text{unbiased}} w_r \cdot \mu_0(R) \ge (1-\varepsilon)\cdot\mu_0 - \frac{\delta}{\rho}\cdot\mu_1. \quad (2.9)$$

Define a probability distribution on the rectangles R as follows $p(R) := w_R/\mathsf{srec}^{0,\mu}_{\varepsilon,\delta}(f)$. Then (2.9) can be rewritten as

$$\mathbb{E}_R\left[\mathbb{I}_{\text{biased}}(R)\cdot\mu_0(R)\right] \ge \frac{1}{D}\cdot\left((1-\varepsilon)\cdot\mu_0 - \frac{\delta}{\rho}\cdot\mu_1\right).$$

Hence, there exists a large biased rectangle $S = X_0 \times Y_0$ as claimed. ◀

Proof of Lemma 9. Since $R^{(00)}$ is $(1-\sqrt{\delta})$-biased towards 0, we have from the packing and covering constraints (2.2) and (2.3) that

$$\begin{aligned}&\sum_{(x,y)\in R^{(00)}} \mu_{x,y} \sum_{R\ni(x,y)} w_R \\ &= \sum_{(x,y)\in R^{(00)}\cap f^{-1}(1)} \mu_{x,y} \sum_{R\ni(x,y)} w_R + \sum_{(x,y)\in R^{(00)}\cap f^{-1}(0)} \mu_{x,y} \sum_{R\ni(x,y)} w_R \\ &\le \mu_1(R^{(00)}) + \delta\mu_0(R^{(00)}) \le (\sqrt{\delta}+\delta)\mu_0(R^{(00)}) \le 2\sqrt{\delta}\mu(R^{(00)}).\end{aligned}$$

Hence,

$$\sum_R w_R \cdot \left(\frac{\mu(R^{(00)}\cap R)}{\mu(R^{(00)})}\right) \le 2\sqrt{\delta}. \quad (2.10)$$

Group the rectangles in to subsets as follows:

$$B^{(01)} := \left\{R : \frac{\mu(R^{(01)}\cap R)}{\mu(R^{(01)})} \ge \frac{10\sqrt[4]{\delta}}{D}\right\}, \qquad B^{(10)} := \left\{R : \frac{\mu(R^{(10)}\cap R)}{\mu(R^{(10)})} \ge \frac{10\sqrt[4]{\delta}}{D}\right\},$$
$$B := \left\{R : \frac{\mu(R^{(11)}\cap R)}{\mu(R^{(11)})} \ge \frac{10}{D}\right\}.$$

By (2.3), we have

$$\sum_{(x,y)\in R^{(11)}} \mu_{x,y} \sum_{R\ni(x,y)} w_R \le \sum_{(x,y)\in R^{(11)}} \mu_{x,y} = \mu(R^{(11)}).$$

Or equivalently,

$$\sum_R \frac{w_R}{D} \cdot \frac{\mu(R^{(11)} \cap R)}{\mu(R^{(11)})} \leq \frac{1}{D}.$$

Hence,

$$\sum_{R \in B} w_R \leq 0.1D. \tag{2.11}$$

We will now argue that either $\sum_{R \in B^{(01)}} w_R \leq 0.9D$ or $\sum_{R \in B^{(10)}} w_R \leq 0.9D$. Suppose, for contradiction, that neither is true. Then, by (2.11) we have

$$\sum_{R \in \left(B^{(01)} \cap B^{(10)}\right) \setminus B} w_R \geq 0.7D. \tag{2.12}$$

Since μ is a product distribution we have

$$\frac{\mu(R^{(01)} \cap R)}{\mu(R^{(01)})} \cdot \frac{\mu(R^{(10)} \cap R)}{\mu(R^{(10)})} = \frac{\mu(R^{(00)} \cap R)}{\mu(R^{(00)})} \cdot \frac{\mu(R^{(11)} \cap R)}{\mu(R^{(11)})}.$$

Using the above we have

$$\begin{aligned}
&\sum_R w_R \cdot \left(\frac{\mu(R^{(00)} \cap R)}{\mu(R^{(00)})}\right) \\
&\geq \sum_{R \in \left(B^{(01)} \cap B^{(10)}\right) \setminus B} w_R \cdot \left(\frac{\mu(R^{(00)} \cap R)}{\mu(R^{(00)})}\right) \\
&\geq \sum_{R \in \left(B^{(01)} \cap B^{(10)}\right) \setminus B} w_R \cdot \left(\frac{\mu(R^{(01)} \cap R)}{\mu(R^{(01)})}\right) \cdot \left(\frac{\mu(R^{(10)} \cap R)}{\mu(R^{(10)})}\right) \Bigg/ \left(\frac{\mu(R^{(11)} \cap R)}{\mu(R^{(11)})}\right) \\
&\geq \sum_{R \in \left(B^{(01)} \cap B^{(10)}\right) \setminus B} w_R \cdot \left(\frac{10\sqrt[4]{\delta}}{D}\right) \cdot \left(\frac{10\sqrt[4]{\delta}}{D}\right) \Bigg/ \left(\frac{10}{D}\right) \\
&\geq \frac{10\sqrt{\delta}}{D} \cdot (0.7D) \\
&= 7\sqrt{\delta}.
\end{aligned}$$

This contradicts (2.10). Hence, either $\sum_{R \in B^{(01)}} w_R \leq 0.9D$ or $\sum_{R \in B^{(10)}} w_R \leq 0.9D$. Assume, wlog that $\sum_{R \in B^{(01)}} w_R \leq 0.9D$. If f is 1/2-biased towards 0 with respect to the distribution $\mu^{(01)}$, then the alternative (a) of the lemma holds, and we are done. Otherwise, that is $\mu_0(R^{(01)}) \leq 2\mu_1(R^{(01)})$ or equivalently $\mu(R^{(01)}) \leq 3\mu_1^{(01)}(R^{(01)})$. We will infer from this that $\mathsf{srec}^{1,\mu^{(01)}}_{\varepsilon^{(01)}+30\sqrt[4]{\delta},\delta}(f) \leq 0.9D$. Consider the primal solution given by

$$w'_R = \begin{cases} w_R, & \text{if } R \in B^{(01)} \\ 0, & \text{if } R \notin B^{(01)}. \end{cases}$$

Clearly, w'_R, being a part of the original solution, satisfies (2.2) and (2.3), and has objective value at most $0.9D$. All we need to show is that it satisfies the covering constraint (2.1). For this, we first consider

$$\sum_{R \notin B^{(01)}} w_R \cdot \left(\frac{\mu_1(R^{(01)} \cap R)}{\mu(R^{(01)})}\right) \leq \sum_{R \notin B^{(01)}} w_R \cdot \left(\frac{\mu(R^{(01)} \cap R)}{\mu(R^{(01)})}\right) \leq \frac{10\sqrt[4]{\delta}}{D} \cdot D \leq 10\sqrt[4]{\delta}. \tag{2.13}$$

Now,

$$\begin{aligned}
&\sum_{(x,y)\in f^{-1}(1)\cap R^{(01)}} \mu_{x,y} \sum_{R\in(x,y)} w'_R \\
&= \sum_{(x,y)\in f^{-1}(1)\cap R^{(01)}} \mu_{x,y} \sum_{R\in(x,y),R\in B^{(01)}} w_R \\
&= \sum_{(x,y)\in f^{-1}(1)\cap R^{(01)}} \mu_{x,y} \left(\sum_{R\in(x,y)} w_R - \sum_{R\in(x,y),R\notin B^{(01)}} w_R \right) \\
&= (1-\varepsilon^{(01)})\mu_1(R^{(01)}) - \sum_{(x,y)\in f^{-1}(1)\cap R^{(01)}} \mu_{x,y} \sum_{R\in(x,y),R\notin B^{(01)}} w_R \\
&= (1-\varepsilon^{(01)})\mu_1(R^{(01)}) - \sum_{R\notin B^{(01)}} w_R \mu_1(R^{(01)}\cap R) \\
&\geq (1-\varepsilon^{(01)})\mu_1(R^{(01)}) - 10\sqrt[4]{\delta}\mu(R^{(01)}) && \text{[From (2.13)]} \\
&\geq (1-\varepsilon^{(01)})\mu_1(R^{(01)}) - 30\sqrt[4]{\delta}\mu_1(R^{(01)}) && \text{[Since } \mu(R^{(01)}) \leq 3\mu_1(R^{(01)})\text{]} \\
&= (1-\varepsilon^{(01)} - 30\sqrt[4]{\delta})\mu_1(R^{(01)})
\end{aligned}$$

Thus, (2.1) holds for $R^{(01)}$ with ε replaced by $\varepsilon^{(01)} + 30\sqrt[4]{\delta}$. ◀

Acknowledgements. We thank Mika Göös for pointing out an error in an earlier version of the paper and for referring us to the literature.

References

1. Boaz Barak, Mark Braverman, Xi Chen, and Anup Rao. How to compress interactive communication. *SIAM J. Comput.*, 42(3):1327–1363, 2013. (Preliminary Version in *42nd STOC*, 2010). doi:10.1137/100811969.
2. Lila Fontes, Rahul Jain, Iordanis Kerenidis, Sophie Laplante, Mathieu Laurière, and Jérémie Roland. Relative discrepancy does not separate information and communication complexity. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Proc. 42nd International Colloq. of Automata, Languages and Programming (ICALP), Part I*, volume 9134 of *LNCS*, pages 506–516. Springer, 2015. doi:10.1007/978-3-662-47672-7_41.
3. Anat Ganor, Gillat Kol, and Ran Raz. Exponential separation of information and communication. In *Proc. 55th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 176–185, 2014. doi:10.1109/FOCS.2014.27.
4. Anat Ganor, Gillat Kol, and Ran Raz. Exponential separation of information and communication for boolean functions. In *Proc. 47th ACM Symp. on Theory of Computing (STOC)*, pages 557–566, 2015. doi:10.1145/2746539.2746572.
5. Anat Ganor, Gillat Kol, and Ran Raz. Exponential separation of communication and external information. In *Proc. 48th ACM Symp. on Theory of Computing (STOC)*, 2016. (to appear).
6. Mika Göös, T. S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication vs. partition number. Technical Report TR15-169, Elect. Colloq. on Comput. Complexity (ECCC), 2015. URL: http://eccc.hpi-web.de/report/2015/169.
7. Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. In *Proc. 47th ACM Symp. on Theory of Computing (STOC)*, pages 257–266, 2015. doi:10.1145/2746539.2746596.

8 Prahladh Harsha, Rahul Jain, and Jaikumar Radhakrishnan. Partition bound is quadratically tight for product distributions, 2016. arXiv:1512.01968.

9 Rahul Jain and Hartmut Klauck. The partition bound for classical communication complexity and query complexity. In *Proc. 25th IEEE Conf. on Computational Complexity*, pages 247–258, 2010. arXiv:0910.4266, doi:10.1109/CCC.2010.31.

10 Rahul Jain and Penghui Yao. A strong direct product theorem in terms of the smooth rectangle bound, 2012. arXiv:1209.0263.

11 Gillat Kol. Interactive compression for product distributions. In *Proc. 48th ACM Symp. on Theory of Computing (STOC)*, 2016. (to appear).

12 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997. doi:10.1017/CBO9780511574948.

13 Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995. (Preliminary version in *35th FOCS*, 1994). doi:10.1007/BF01192527.

14 Clifford D. Smyth. Reimer's inequality and Tardos' conjecture. In *Proc. 34th ACM Symp. on Theory of Computing (STOC)*, pages 218–221, 2002. doi:10.1145/509907.509942.

15 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proc. 11th ACM Symp. on Theory of Computing (STOC)*, pages 209–213, 1979. doi:10.1145/800135.804414.

16 Andrew Chi-Chih Yao. Lower bounds by probabilistic arguments (extended abstract). In *Proc. 24th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 420–428, 1983. doi:10.1109/SFCS.1983.30.

Efficient Plurality Consensus, Or: the Benefits of Cleaning up from Time to Time

Petra Berenbrink[1], Tom Friedetzky[2], George Giakkoupis[3], and Peter Kling*[4]

1 **Simon Fraser University, Burnaby, Canada**
2 **Durham University, Durham, U.K.**
3 **INRIA, Rennes, France**
4 **Simon Fraser University, Burnaby, Canada**
pkling@sfu.ca

Abstract

Plurality consensus considers a network of n nodes, each having one of k opinions. Nodes execute a (randomized) distributed protocol with the goal that all nodes adopt the *plurality* (the opinion initially supported by the most nodes). Communication is realized via the Gossip (or random phone call) model. A major open question has been whether there is a protocol for the complete graph that converges (w.h.p.) in polylogarithmic time and uses only polylogarithmic memory per node (local memory). We answer this question affirmatively.

We propose two protocols that need only mild assumptions on the bias in favor of the plurality. As an example of our results, consider the complete graph and an arbitrarily small constant multiplicative bias in favor of the plurality. Our first protocol achieves plurality consensus in $\mathrm{O}(\log k \cdot \log\log n)$ rounds using $\log k + \Theta(\log\log k)$ bits of local memory. Our second protocol achieves plurality consensus in $\mathrm{O}(\log n \cdot \log\log n)$ rounds using only $\log k + 4$ bits of local memory. This disproves a conjecture by Becchetti et al. (SODA'15) implying that any protocol with local memory $\log k + \mathrm{O}(1)$ has worst-case runtime $\Omega(k)$. We provide similar bounds for much weaker bias assumptions. At the heart of our protocols lies an *undecided state*, an idea introduced by Angluin et al. (Distributed Computing'08).

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases plurality consensus, voting, majority, distributed, gossip

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.136

1 Introduction

Reaching *plurality consensus* is a fundamental problem in distributed computing. We consider this problem in a networked setting, where a graph is given in which each node initially holds one of $k \in \mathbb{N}$ *opinions*. The objective is the design of an efficient distributed protocol that ensures that eventually all nodes agree on the initial *plurality opinion*, which is the opinion that is initially supported by the most nodes. This problem is also referred to as *majority consensus* or *proportionate agreement* [1, 3, 19]. In accordance with [5] and others, we prefer to refer to it as *plurality consensus*, so as to make clear that the opinion eventually to be attained by all nodes need not initially have been absolute majority. One need not stray too far from the core of distributed computing to come across direct applications of plurality consensus: the handling of fault tolerance in parallel computing or the implementation of

* Supported in part by the Pacific Institute for the Mathematical Sciences.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi; Article No. 136; pp. 136:1–136:14

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

majority-based conflict resolution for CRCW PRAMs (and derivative models) are immediate examples.

Natural metrics for plurality consensus protocols are running time and memory overhead. The latter is the additional amount of (local) memory needed by each node above and beyond the bits required to store its current opinion. Results are typically expressed in terms of the number of nodes, the number of initial opinions, and the initial bias (between plurality and remaining opinions). Of particular interest, to us and in general, are fast protocols with small memory overhead. As in [5] we assume a synchronous PULL-based GOSSIP communication model on the underlying graph. Here, in each discrete round, every node may contact one neighbor and query that neighbor's opinion. We mostly assume the complete graph, but using a construction of [5] we can easily extend our results to regular expanders. We define the (relative) plurality gap $\gamma > 1$ as the ratio between the plurality opinion and the second most common opinion. We analyze two protocols in this paper:

- A protocol with running time $\mathrm{O}\big(\log k \cdot \log\log_\gamma n + \log\log n\big)$ and memory overhead $\Theta\left(\log\log k\right)$.
- A protocol with running time $\mathrm{O}\big(\log n \cdot \log\log_\gamma n\big)$ and memory overhead 4.

Plurality consensus is a member of the class of population dynamics, which are of great interest in fields as varied as epidemiology, physics, statistics, biology, chemistry, or sociology. All these have in common an initial population of agents with some initial properties and a protocol (dynamics) that in some manner changes the properties of given agents usually based on those of other agents. Specific models are as varied as the problems themselves; for instance, we may or may not have an underlying graph structure, a prescribed timing model, or restrictions on the amount and nature of communication. Other related dynamics are the lately en vogue voting protocols and Moran-type processes.

1.1 Related Work

In [3], Aspnes et al. consider $k = 2$ initial opinions $\{x, y\}$ and the complete graph as neighborhood structure. They introduce a third state, referred to as *blank*, b, which is a crucial ingredient in their protocol and analysis. Their protocol works such that an activated node u picks another node v at random. Given the opinions of those two nodes, the transition now proceeds as follows: If u has a non-blank opinion and sees in v the other non-blank opinion then it changes to b, if it has a non-blank opinion and sees in v the same non-blank opinion or b then it maintains its opinion, and if it has the blank opinion then it just copies whatever it sees in v. The authors show that with high probability all n nodes reach consensus within $\mathrm{O}\left(n \log n\right)$ many interactions (corresponding to parallel convergence time $\mathrm{O}\left(\log n\right)$), and the consensus value is the plurality value provided its (absolute) initial bias is at least $\omega\left(\sqrt{n}\log n\right)$. Each node needs to be able to store one of three values, x, y, or b.

In the case of two opinions the plurality problem can be solved by calculating the median of the opinions. In [11] the authors present such a protocol that converges in $\mathrm{O}\left(\log n\right)$ rounds and has constant memory overhead, if the initial difference bias $c_1 - c_2$ is at least $\Omega\left(\sqrt{n \log n}\right)$. In [2] the authors consider the plurality problem in a sequential setting where only one node can change its opinion at a time. They present a new protocol called Average and Conquer (AVC) that solves plurality exactly, in sequential time $\mathrm{O}\left(n \log n/(s\epsilon) + \log n \log s\right)$, where ϵn ($\epsilon > 1/2$) is the size of the plurality opinion and s the number of states. In [7] the authors generalize the former result to general networks and k opinions. They introduce protocols that solve the plurality consensus problem that are based on an interesting relationship between plurality consensus and distributed load balancing.

In [5] Becchetti et al. take the model of [3] and generalise it to $k \geq 2$ initial opinions. They still use the blank state, which they however refer to as undecided. The authors express their results in terms of $\mathrm{md}(\bar{\mathbf{c}})$, *the monochromatic distance* of configuration c. Formally, $\mathrm{md}(\bar{\mathbf{c}}) = \sum_{i=1}^{k} c_i/c_1$, where c_i is the number of nodes with opinion i and $c_1 \geq c_2 \cdots \geq c_k$. Note that $\mathrm{md}(\bar{\mathbf{c}})$ is always between $\mathrm{O}(1)$ and $\mathrm{O}(k)$. Then authors show almost-tight bounds on convergence time. Formally, let $k = k(n)$ be any function such that $k = \mathrm{O}\left((n/\log n)^{1/3}\right)$, and consider any initial configuration with $c_1 \geq (1+\alpha) \cdot c_2$, where $\alpha \geq 0$ is any arbitrarily-small constant. Their protocol converges in $\mathrm{O}(\mathrm{md}(\bar{\mathbf{c}}) \cdot \log n)$, rounds (w.h.p.) and it has only constant memory overhead. They also show that for $k = \mathrm{O}\left((n/\log n)^{1/6}\right)$ and any initial configuration the convergence time of their protocol is (w.h.p.) linear in the monochromatic distance. Finally, they show how to adapt their results to regular expanders using random walks to sample the opinion of nodes. As with [3], they require one state more than is necessary to store the actual opinion values. They conjecture that any protocol using $\log k + \mathrm{O}(1)$ bits of memory has runtime at least linear in k in the worst case — as discussed later, we partially refute this conjecture. In [6] the authors consider the 3-majority dynamics. They show that for $k \leq n^{\alpha}$ (with constant α) the 3-majority dynamics converges to an *almost-consensus* state in time $\mathrm{O}\left((k^2\sqrt{\log n} + kl\log n)(k+\log n)\right)$. An almost-consensus state is defined as a state where all but a subset of size $\mathrm{O}(n^{\gamma})$ (for constant $\gamma < 1$) of the nodes support the same opinion. In [4] the authors consider the undecided dynamics in complete graphs in an asynchronous setting. They derive the time of convergence and an upper bound for the probability of error.

A line of research which is related to the plurality consensus problem is the voting problem. The setting is the same, a network with n nodes is given and initially every node has one of k opinions. Here the goal is that all nodes agree on one opinion, which is not necessarily the plurality opinion. A sequential version of the voter model was introduced in [16]. The parallel voter model was first analyzed in [15]. The authors of [15] bound the expected consensus time in terms of the expected meeting time T_m of two random walks and show a bound of $\mathrm{O}(T_m \cdot \log n) = \mathrm{O}\left(n^3 \log n\right)$. The authors of [8] provide an improved upper bound of $O\big(1/(1-\lambda_2) \cdot \log^4 n + \rho\big)$ on the expected consensus time for any graph G, where λ_2 is the second eigenvalue of the transition matrix of a random walk on G, and $\rho = \big(\sum_{u \in V(G)} d(u)\big)^2 / \sum_{u \in V(G)} d^2(u)$ is the ratio of the square of the sum of node degrees over the sum of the squared degrees. The authors of [9, 10] consider a modification of the standard voter model with two opinions, which they call *two-sample voting*. In every round, each node chooses two of its neighbors randomly and adopts their opinion only if they both agree. For regular graphs and random regular graphs, it is shown that two-sample voting has a consensus time of $\mathrm{O}(\log n)$ if the initial imbalance between the nodes having the two opinions is large enough. In [13] the authors consider a 2-choice voting protocol for k opinions in the complete graph. Their protocol converges to the majority opinion in time $\mathrm{O}(k \cdot \log n)$, with high probability, if k=$\mathrm{O}(n^{\epsilon})$ for some small $\epsilon > 0$, and the initial absolute gap between largest and second-largest opinion is $\Omega\left(\sqrt{n \log n}\right)$. They also show that there exist initial configurations where the $\Theta(k)$ bound on the run time is matched. Independently, they also give a protocol which is similar to our simple, first protocol (cf. Section 3) and has roughly the same voting time. Other related papers from literature about sensor networks include [12] (which considers *binary interval consensus*, which can be used to solve majorization) and [17] (which considers the plurality problem in a different distributed model and for constant k).

1.2 Our Contribution

In Section 3 we present and analyze protocol RepeatedCleanup, which in a complete graph of n nodes and k opinions works in $\mathrm{O}\left(\log k \cdot \log\log_{\gamma} n + \log\log n\right)$ rounds and has a memory

overhead of $O(\log\log k)$ bits. The protocol is a synchronized version of the protocol from [6], in which the nodes "lose" their opinion and adopt the state "undecided" whenever they sample a node with a different opinion. Then they will adopt the "real" opinion of the next (i.e., not undecided) node they sample. The nodes keep changing their opinion until all nodes agree on the plurality opinion. In contrast, our protocol works in $O\left(\log\log_\gamma n\right)$ many phases, each of length $\Theta(\log k)$. In the first round of every phase the nodes sample a node and lose the opinion if the sampled node has a different opinion. They then use the rest of the phase to find a new "real" opinion. In [5] the authors show that for $k = O\left((n/\log n)^{1/6}\right)$ and any initial configuration the convergence time of their protocol is linear in the monochromatic distance, which can be as large as $O(k)$. Hence, our protocol outperforms the lower bound for the protocol of [5]. Interestingly, the speed-up is reached by synchronizing the protocol, which can also be regarded as slowing it down. In our protocol a node that just found a real opinion again waits until the beginning of the next round to sample another node, instead of doing that immediately.

The drawback of our first protocol is that it uses a counter to determine the end of a phase. In Section 4 we present a protocol that works in $O\left(\log n \cdot \log\log_\gamma n\right)$ rounds and has a memory overhead of only 4 bits. The main idea of the protocol is to slow down the progress in the individual phases by having nodes toss a biased coin with success probability $1/n$, basically replacing the deterministic counter by a probabilistic counterpart. Our result shows that a conjecture by Becchetti et al. [5], implying that any protocol with constant memory overhead has worst-case runtime $\Omega(k)$, does not hold if nodes have access to such a coin. Note that the coin toss is not necessary if nodes can decide whether they sampled themselves (or a marked node/leader).

A very recent, independent result by Ghaffari and Parter [14] suggests a protocol for plurality consensus with similar time and memory bounds as ours. They employ the same basic idea of cleanup and decision-accumulation rounds (cf. Section 3), which they name selection and recovery steps. Their final protocol differs in that they use some of the undecided nodes as *clock nodes* (which use the $\log k$ bits normally used to store the opinion to count time) to help synchronize other nodes.

2 Model & Notation

We consider protocols in the complete graph with $n \in \mathbb{N}$ nodes. Each node u has one of $k \in \mathbb{N}$ opinions $\mathrm{op}_u \in \{1, 2, \ldots, k\}$. We write $\mathrm{op}_u = \perp$ to indicate that node u is undecided. Time is modelled in synchronous, discrete and parallel rounds and we assume a PULL-based GOSSIP model for communication (nodes can request information from one other node chosen uniformly at random). Note that each node needs at least $\lceil \log k \rceil$ bits of local memory (to store its current opinion). Any additional number of bits per node needed by a given protocol is called the protocol's *memory overhead*.

Notation. In the following, $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the L^1 and L^2 norms, respectively, that is, $\|\boldsymbol{x}\|_1 = \sum_{i=1}^n |x_i|$ and $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$ for an n-dimensional vector $\boldsymbol{x}$. For a real value $x > 0$, its binary logarithm is denoted by $\log x$ and its natural logarithm by $\ln x$. For an integer i, the shorthand $[i] \coloneqq \{1, 2, \ldots, i\}$ denotes the set of the first i integers. The phrase *with high probability* (w.h.p.) refers to probabilities of the form $1 - n^{-\Omega(1)}$.

At any point in time, the system can be described by a k-dimensional vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_k) \in \{0/n, 1/n, \ldots, 1\}^k \subseteq [0,1]^k$, where the i-th entry $x_i \in [0,1]$ denotes the current fraction of nodes with opinion i. We call such a vector $\boldsymbol{x}$ a *configuration*. Note

that $1 - \|\boldsymbol{x}\|_1 \in [0,1]$ is the fraction of undecided nodes. $X_i(t)$ is defined as the random variable that takes as its values the configuration produced by the protocol at the end of round $t \in \mathbb{N}_0$. We use $\bar{\boldsymbol{x}} \coloneqq \boldsymbol{X}(0)$ to denote the initial (fixed) configuration. The random variable $Y(t) \coloneqq 1 - \|\boldsymbol{X}(t)\|_1$ with values in $\in [0,1]$ denotes the fraction of undecided nodes at the end of round t.

To measure how far we are from plurality consensus, we define two *plurality gap* notions: Assuming (w.l.o.g.) that 1 is a most common opinion, let $\psi(\boldsymbol{x}) \coloneqq x_1 - \max_{i \neq 1} x_i \in [0,1]$ be the *absolute* plurality gap. Similarly, $\gamma(\boldsymbol{x}) \coloneqq x_1 / \max_{i \neq 1} x_i \geq 1$ is the *relative* plurality gap.

Assumptions. In order to guarantee convergence, we need some (mild) bias assumptions. Given the initial configuration $\bar{\boldsymbol{x}}$, without loss of generality (w.l.o.g.), we assume $\bar{x}_1 \geq \bar{x}_i$ for all $i \in [k]$ and refer to (the initially most common) opinion 1 as the *plurality opinion*. For most of the analysis we assume

$$\psi(\bar{\boldsymbol{x}}) = \omega\left(\frac{(\log n)^2}{\sqrt{n}}\right) \tag{1}$$

and

$$k = \mathrm{o}\left(\frac{\sqrt{n}}{(\log n)^2}\right). \tag{2}$$

While Condition (1) is essential[1], Condition (2) is without loss of generality (it can be achieved by merging small opinions). We define $\rho = \rho(n) \coloneqq (\log n)^2/\sqrt{n}$ and call a configuration $\boldsymbol{x}$ *biased* if $\psi(\boldsymbol{x}) = \omega(\rho)$. Our analysis assumes n to be at least some sufficiently large constant.

3 Plurality Consensus with $\log k + \Theta(\log\log k)$ Bits

We divide time into *phases*, each consisting of $T \coloneqq 5 + 2\log k$ rounds. In each round, every node u uses a Pull operation to sample a random node v and checks its opinion op_v. We distinguish two types of rounds:

- *Cleanup* rounds represent the first round of each phase. Here, u becomes undecided if op_v differs from u's own opinion op_u (and keeps its opinion otherwise).
- *Decision-accumulation* rounds make up the remaining $T - 1$ rounds of a phase. Here, only undecided nodes act and simply adopt the pulled opinion op_v.

The synchronization of the steps in which nodes lose their opinion and adopt opinions of sampled nodes is key to the fast convergence. As we will see in the analysis they ensure that each phase increases the relative plurality gap exponentially. We call this protocol RepeatedCleanup. See Listing 1 for a formal description. In the remainder of this section we prove the following theorem.

▶ **Theorem 1.** RepeatedCleanup *has a local memory overhead of* $\Theta(\log\log k)$ *bits. If started on a biased configuration* $\bar{\boldsymbol{x}}$ *with relative plurality gap* $\gamma \coloneqq \gamma(\bar{\boldsymbol{x}})$*, then (w.h.p.) plurality consensus is achieved in* $\mathrm{O}\big(\log(k) \cdot \log\log_\gamma n + \log\log n\big)$ *rounds.*

We use the shorthand $\boldsymbol{X}(\tau, t) \coloneqq \boldsymbol{X}((\tau - 1) \cdot T + t)$ to denote the configuration at the end of round t in phase τ. Similarly, $X_i(\tau, t)$ and $Y(\tau, t)$ denote the corresponding fractions

[1] Our results hold also under the slightly weaker condition $\psi(\bar{\boldsymbol{x}}) = \omega\big(\sqrt{\log(n)/n}\big)$. However, Condition (1) significantly simplifies some parts of the analysis.

```
1  sample a random node v
2  if t ≡ 0 (mod T):                                   {cleanup}
3      if op_v ≠ op_u:   op_u ← ⊥
4  else:                                               {decision−accumulation}
5      if op_u = ⊥:      op_u ← op_v
```

Listing 1 Protocol RepeatedCleanup as executed by node u in round $t \in \mathbb{N}$. It works in phases of length $T = \Theta(\log k)$ and has memory overhead $\Theta(\log\log k)$.

of nodes. The following definition identifies opinions that are supported by only a small fraction of nodes (and, thus, are likely to vanish in a cleanup round), and configurations that do not have too many undecided nodes.

- An opinion i in configuration $\boldsymbol{x}$ is *negligible* if $x_i \le \rho$ (recall that $\rho = (\log n)^2/\sqrt{n}$).
- A configuration $\boldsymbol{x}$ is *alive* if $\|\boldsymbol{x}\|_1 \ge 1 - e^{-1}$.

The following simple observation already hints at the basic change of configurations during a phase: in expectation, the cleanup round squares the relative plurality gap (but reduces the absolute support of each opinion), while the decision-accumulation rounds increase the absolute support of each opinion (not changing the relative plurality gap).

▶ **Observation 2.** *Consider a configuration $\boldsymbol{x}$ with $y \coloneqq 1 - \|\boldsymbol{x}\|_1$. If the configuration $\boldsymbol{X}$ results from a cleanup round on $\boldsymbol{x}$ and $Y = 1 - \|\boldsymbol{X}\|_1$, then*

$$\mathbb{E}[X_i] = x_i^2, \qquad \mathbb{E}[\|\boldsymbol{X}\|_1] = \|\boldsymbol{x}\|_2^2, \qquad \text{and} \qquad \mathbb{E}[Y] = 2y - y^2. \tag{3}$$

Similarly, if $\boldsymbol{X}$ results from a decision-accumulation round, then

$$\mathbb{E}[X_i] = (1+y)\cdot x_i, \qquad \mathbb{E}[\|\boldsymbol{X}\|_1] = 2\|\boldsymbol{x}\|_1 - \|\boldsymbol{x}\|_1^2, \qquad \text{and} \qquad \mathbb{E}[Y] = y^2\,. \tag{4}$$

Analysis Overview. The bound on the memory overhead in Theorem 1 immediately follows from the protocol description. The runtime bound is proven in three steps: Lemma 7 shows that all non-plurality opinions become negligible during the first $O\big(\log\log_{\gamma(\bar{\boldsymbol{x}})} n\big)$ phases. Lemma 8 proves that all these negligible opinions vanish within an additional constant number of phases. Lemma 9 shows that any remaining undecided nodes vanish in another $O(\log\log n)$ phases. Before we prove these key lemmas (Section 3.3), we show how configurations evolve during single rounds (Section 3.1) and single phases (Section 3.2).

3.1 Change During a Single Round

Our first two claims show concentration for the expected configuration change in cleanup and decision-accumulation rounds we saw in Observation 2. They follow by standard Chernoff bounds.

▶ **Claim 3** (Cleanup Round). *Consider a configuration $\boldsymbol{x}$ at the beginning of phase τ. Let $a > 0$ be a constant.*

(a) *Let $i \in [k]$ and $\delta \coloneqq \sqrt{3a\cdot\log(n)/n}\cdot x_i^{-1}$. Then*

$$\Pr\left[X_i(\tau,1) \ge \max\left(\rho^2, (1+\delta)\cdot x_i^2\right) \,\middle|\, \boldsymbol{X}(\tau,0) = \boldsymbol{x}\right] \le n^{-a} \qquad \text{and} \tag{5}$$

$$\Pr\left[X_i(\tau,1) \le (1-\delta)\cdot x_i^2 \,\middle|\, \boldsymbol{X}(\tau,0) = \boldsymbol{x}\right] \le n^{-a}. \tag{6}$$

(b) *Let $\delta \coloneqq \sqrt{3a\cdot\log(n)/n}\cdot\|\boldsymbol{x}\|_2^{-1}$. Then* $\Pr\left[\|\boldsymbol{X}(\tau,1)\|_1 \le (1-\delta)\cdot\|\boldsymbol{x}\|_2^2 \,\middle|\, \boldsymbol{X}(\tau,0) = \boldsymbol{x}\right] \le n^{-a}$.

▶ **Claim 4** (Decision-accumulation Round). *Consider a configuration $\boldsymbol{x}$ at the beginning of round $t+1>1$ in phase τ. Let $a>0$ be a constant and $y \coloneqq 1-\|\boldsymbol{x}\|_1$ the fraction of undecided nodes.*

(a) *Let $i \in [k]$, $\delta' \coloneqq \sqrt{13a \cdot \log n/(x_i y \cdot n)}$, and $\delta \coloneqq \max(\delta', \delta'^2)$. Then*

$$\Pr\left[X_i(\tau,t+1) \geq (1+\delta y)\cdot x_i \cdot (1+y) \mid \boldsymbol{X}(\tau,t)=\boldsymbol{x}\right] \leq n^{-a} \qquad \text{and} \tag{7}$$

$$\Pr\left[X_i(\tau,t+1) \leq (1-\delta' y)\cdot x_i \cdot (1+y) \mid \boldsymbol{X}(\tau,t)=\boldsymbol{x}\right] \leq n^{-a}\,. \tag{8}$$

(b) *Assume $y \leq 1-\rho$. Then* $\Pr\left[Y(\tau,t+1) \geq \max\left(\rho^2, y^{3/2}\right) \mid \boldsymbol{X}(\tau,t)=\boldsymbol{x}\right] \leq n^{-a}$.

3.2 Change During a Single Phase

Next, we use the effects of single rounds to show that (a) the property of being alive is (w.h.p.) invariant from phase to phase (Claim 5) and (b) the relative plurality gap of a biased configuration increases (w.h.p.) exponentially during a phase (Claim 6).

▶ **Claim 5.** *Consider a configuration $\boldsymbol{x}$ at the beginning of phase τ. Let $a>0$ be a constant. Assume $\|\boldsymbol{x}\|_1 \geq 1-e^{-1}$ and $\|\boldsymbol{x}\|_2^2 = \omega(\rho)$. Then*

$$\Pr\left[\|\boldsymbol{X}(\tau+1,0)\|_1 < 1-e^{-1} \;\middle|\; \boldsymbol{X}(\tau,0)=\boldsymbol{x}\right] = n^{-a}\,. \tag{9}$$

Proof. Let us first consider the effect of the cleanup round. Claim 3(b) gives

$$\Pr\left[\|\boldsymbol{X}(\tau,1)\|_1 \leq \|\boldsymbol{x}\|_2^2/2 \;\middle|\; \boldsymbol{X}(\tau,0)=\boldsymbol{x}\right] \leq n^{-a-1}\,, \tag{10}$$

where we used $\|\boldsymbol{x}\|_2^2 = \omega(\rho) = \omega\big(\sqrt{\log(n)/n}\big)$, such that the involved δ-term becomes $\mathrm{o}(1)$. Next, fix the configuration $\tilde{\boldsymbol{x}} = X(\tau,1)$ after the cleanup round. Let $\tilde{y} \coloneqq 1-\|\tilde{\boldsymbol{x}}\|_1$ and assume $\tilde{y} \leq 1-\|\boldsymbol{x}\|_2^2/2$ (this holds with probability at least $1-n^{-a-1}$ due to Equation (10)). With the claim's assumption $\|\boldsymbol{x}\|_2^2 = \omega(\rho)$, this implies $\tilde{y} \leq 1-\rho$ and we can apply Claim 4(b) to get

$$\Pr\left[Y(\tau,t+1) \geq \max(\rho^2, \tilde{y}^{3/2}) \;\middle|\; \boldsymbol{X}(\tau,1)=\tilde{\boldsymbol{x}}\right] \leq n^{-a-1} \tag{11}$$

for a decision-accumulation round $t+1>1$. A union bound over the $T-1=\mathrm{O}\big(\log n\big)$ decision-accumulation rounds, combined with Equation (10) via the law of total probability, gives

$$\Pr\left[Y(\tau,T) \geq \max\left(\rho^2, \left(1-\|\boldsymbol{x}\|_2^2/2\right)^{(3/2)^{T-1}}\right) \;\middle|\; \boldsymbol{X}(\tau,0)=\boldsymbol{x}\right] \leq n^{-a}\,. \tag{12}$$

The second term of the maximum is at most $\exp\big(-(3/2)^{T-1}\cdot\|\boldsymbol{x}\|_2^2/2\big)$. For this to be at most some value $z>0$, we need $T-1 \geq \log_{3/2}\big(2\ln(1/z)/\|\boldsymbol{x}\|_2^2\big)$. Choosing $z \coloneqq \exp\big(-k\cdot\|\boldsymbol{x}\|_2^2/\|\boldsymbol{x}\|_1^2\big)$ and remembering our choice of T (see beginning of Section 3), we calculate

$$\log_{3/2}\left(\frac{2\ln(1/z)}{\|\boldsymbol{x}\|_2^2}\right) = \log_{3/2}\left(2/\|\boldsymbol{x}\|_1^2\right) + \log_{3/2}(k) \leq 4+2\log(k) = T-1\,, \tag{13}$$

where we used the claim's assumption $\|\boldsymbol{x}\|_1 \geq 1-e^{-1}$ to bound $\log_{3/2}\big(2/\|\boldsymbol{x}\|_1^2\big) \leq 4$. Thus, Equation (12) implies that with probability at least $1-n^{-a}$ we have $Y(\tau,T) \leq \max(\rho^2,z) \leq e^{-1}$, where we used $\rho^2=\mathrm{o}(1)$ and the Cauchy-Schwarz inequality to get $z=\exp(-k\cdot\|\boldsymbol{x}\|_2^2/\|\boldsymbol{x}\|_1^2) \leq e^{-1}$. The claim follows with this from $\|\boldsymbol{X}(\tau+1,0)\|_1 = \|\boldsymbol{X}(\tau,T)\|_1 = 1-Y(\tau,T)$. ◀

▶ **Claim 6.** *Consider a configuration $\boldsymbol{x}$ at the beginning of phase τ. Let $a > 0$ be a constant. For an opinion $i \neq 1$ assume $x_1 - x_i = \omega(\rho)$ and let $\gamma_i \coloneqq \min(x_1/\rho, x_1/x_i)$. Then*

$$\Pr\left[\frac{X_1(\tau+1,0)}{X_i(\tau+1,0)} \leq \gamma_i^{3/2} \;\middle|\; \boldsymbol{X}(\tau,0) = \boldsymbol{x}\right] \leq n^{-a}\,. \tag{14}$$

Proof. For a configuration $\boldsymbol{x'}$ define the shorthands $\gamma_i(\boldsymbol{x'}) \coloneqq x'_1/x'_i$ and $\psi_i(\boldsymbol{x'}) \coloneqq x'_1 - x'_i$. We make an inductive argument over γ_i during the phase (using ψ_i as an auxiliary tool for the induction). Applying Claim 3(a) to both opinions (once the upper and once the lower bound) yields

$$\Pr\left[X_1(\tau,1) \leq (1-\delta_1)\cdot x_1^2 \;\middle|\; \boldsymbol{X}(\tau,0) = \boldsymbol{x}\right] \leq n^{-a-1} \qquad \text{and} \tag{15}$$

$$\Pr\left[X_i(\tau,1) \geq \max\left(\rho^2, (1+\delta_i)\cdot x_i^2\right) \;\middle|\; \boldsymbol{X}(\tau,0) = \boldsymbol{x}\right] \leq n^{-a-1}\,, \tag{16}$$

where $\delta_j \coloneqq \sqrt{3a\cdot\log(n)/n}\cdot x_j^{-1}$ for $j \in \{1, i\}$. Let $\delta_\rho \coloneqq \sqrt{3a\cdot\ln(n)/n}\cdot\rho^{-1}$ and $\delta \coloneqq \min(\delta_i, \delta_\rho)$. Note that $\delta_1 \leq \delta$, since $x_1 \geq x_i + \omega(\rho)$ by the claim's assumption. With this, we bound the right-hand side in the probability of Equation (15) by $(1-\delta)\cdot x_1^2$ and, similarly, the right-hand side in the probability of Equation (16) by $(1+\delta)\cdot\max(\rho^2, x_i^2)$. Using a union bound and the inequality $(1-x)/(1+x) \geq (1-2x)$ for $x \in [0,1]$, with probability at least $1 - 2n^{-a-1}$ we have $\gamma_i(\boldsymbol{X}(\tau,1)) \geq (1-2\delta)\cdot\gamma_i^2$. Similarly, we also have $\psi_i(\boldsymbol{X}(\tau,1)) = \omega\left(\rho^2\right)$. For $x_i \leq \rho$, this follows immediately from Equations (15) and (16) (since $x_1^2 = \omega\left(\rho^2\right)$ and $x_i^2 = \mathrm{O}\left(\rho^2\right)$). For $x_i > \rho$, we calculate

$$\begin{aligned}
\psi_i(\boldsymbol{X}(\tau,1)) &= X_1(\tau,1) - X_i(\tau,1) \geq (1-\delta_1)\cdot x_1^2 - (1+\delta_i)\cdot x_i^2 \\
&= \left(x_1^2 - x_i^2 - x_1\cdot\sqrt{\frac{3a\cdot\log n}{n}} - x_i\cdot\sqrt{\frac{3a\cdot\log n}{n}}\right) \\
&= (x_1+x_i)\cdot\left(x_1 - x_i - 2\sqrt{\frac{3a\cdot\log n}{n}}\right) = \omega(\rho)\cdot(\omega(\rho) - \mathrm{o}(\rho)) = \omega\left(\rho^2\right)\,.
\end{aligned} \tag{17}$$

Next, fix the configuration $\tilde{\boldsymbol{x}} = X(\tau,t)$ and let $\tilde{y} \coloneqq 1 - \|\tilde{\boldsymbol{x}}\|_1$. Assume $\psi_i(\tilde{\boldsymbol{x}}) = \omega\left(\rho^2\right)$ and $\tilde{x}_i \geq \rho^2$. Applying Claim 4(a) to both opinions (once the upper and once the lower bound) yields

$$\Pr\left[X_1(\tau,t+1) \leq (1-\tilde{\delta}'_1\tilde{y})\cdot\tilde{x}_1\cdot(1+\tilde{y}) \;\middle|\; \boldsymbol{X}(\tau,t) = \tilde{\boldsymbol{x}}\right] \leq n^{-a-1} \qquad \text{and} \tag{18}$$

$$\Pr\left[X_i(\tau,t+1) \geq (1+\tilde{\delta}_i\tilde{y})\cdot\tilde{x}_i\cdot(1+\tilde{y}) \;\middle|\; \boldsymbol{X}(\tau,t) = \tilde{\boldsymbol{x}}\right] \leq n^{-a-1}\,, \tag{19}$$

where $\tilde{\delta}'_j \coloneqq \sqrt{13a\cdot\log n/(\tilde{x}_j\tilde{y}\cdot n)}$ and $\tilde{\delta}_j \coloneqq \max(\tilde{\delta}'_j, \tilde{\delta}'^2_j)$ for $j \in \{1, i\}$. Note that $\tilde{\delta}'_1 \leq \tilde{\delta}_i$, since $\tilde{x}_1 \geq \tilde{x}_i + \omega\left(\rho^2\right)$ by our assumption. Thus, as before we can combine Equations (18) and (19) via a union bound to get that with probability at least $1 - 2n^{-a-1}$ we have $\gamma_i(\boldsymbol{X}(\tau,t+1)) \geq (1-2\tilde{\delta}_i\tilde{y})\cdot\gamma_i(\tilde{\boldsymbol{x}})$ and – analogous to the calculation in Equation (17) – $\psi_i(\boldsymbol{X}(\tau,t+1)) = \omega\left(\rho^2\right)$. Now, define the error $\tilde{\delta} \coloneqq 2\tilde{\delta}_i\tilde{y}$ and note that

$$\tilde{\delta} = \max\left(\sqrt{\frac{13a\cdot\log n}{\tilde{x}_i n}}\cdot\sqrt{\tilde{y}}, \frac{13a\cdot\log n}{\tilde{x}_i n}\right) \leq \sqrt{\frac{13a\cdot\log n}{\tilde{x}_i n}} = \mathrm{O}\left((\log n)^{-3/2}\right)\,, \tag{20}$$

where we used the assumption $\tilde{x}_i \geq \rho^2 = \omega(\log(n)/n)$. In particular, this implies that (w.h.p.) we have $\gamma_i(\boldsymbol{X}(\tau,t+1)) \geq (1-\tilde{\delta})\cdot\gamma_i(\tilde{\boldsymbol{x}})$, $\psi_i(\boldsymbol{X}(\tau,t+1)) = \omega\left(\rho^2\right)$, and $X_i(\tau,t+1) \geq X_i(\tau,t) = \tilde{x} \geq \rho^2$. Moreover, the error $\tilde{\delta}$ is non-increasing in t (since $\tilde{x}_i$ is non-decreasing in

t), such that we get the largest error for $t = 1$, such that we can apply the above recursively. Applying this via the chain rule to the $T - 1 = \mathrm{O}(\log n)$ decision accumulation rounds yields

$$\Pr\left[\gamma_i(\boldsymbol{X}(\tau,T)) \geq (1-\tilde{\delta})^{T-1} \cdot \frac{\tilde{x}_1}{\max(\rho^2, \tilde{x}_i)} \;\middle|\; \boldsymbol{X}(\tau,1) = \tilde{\boldsymbol{x}}\right] \geq 1 - \mathrm{O}(\log n) \cdot 2n^{-a-1}\,. \quad (21)$$

Since $\tilde{\delta} = \mathrm{O}\big((\log n)^{-3/2}\big)$, we can bound the error term $(1-\tilde{\delta})^{T-1} \geq \exp(-(T-1)\cdot 2\tilde{\delta}) \geq 1 - 2T\cdot\tilde{\delta}$. That is, the error due to the decision-accumulation rounds is $1-\delta'$ for $\delta' \coloneqq 2T\cdot\tilde{\delta} = \mathrm{O}\big((\log n)^{-1/2}\big)$. We now combine our result for the cleanup round and the decision-accumulation rounds via the law of total probability to get that with probability at least $1-n^{-a}$ we have $\gamma_i(\boldsymbol{X}(\tau,T)) \geq (1-2\delta)(1-\delta')\cdot\gamma_i^2$.

Using $\check{\delta} \coloneqq 2\max(2\delta,\delta') = \mathrm{O}\big(\log(n)\cdot\sqrt{\log(n)/n}\big)\cdot\min(x_i^{-1},\rho^{-1})$, we get $\gamma_i(\boldsymbol{X}(\tau,T)) \geq (1-\check{\delta})\cdot\gamma_i^2$. It merely remains to verify that $1-\check{\delta} \geq \gamma_i^{-1/2}$. This is equivalent to $\check{\delta} \leq 1-\gamma_i^{-1/2}$. If $\gamma_i = \Omega(1)$, this holds trivially since $\check{\delta} = \mathrm{o}(1)$. So assume $\gamma_i = 1+\varepsilon$ for a suitable $\varepsilon \in (0,1]$. For this range, we have $1-\gamma_i^{-1/2} \geq \varepsilon/4$, such that it is sufficient to show $\check{\delta} \leq \varepsilon/4$. By the claim's assumption $x_1 - x_i = \omega(\rho)$, we get $\varepsilon = \gamma_i - 1 = x_1/x_i - 1 = \omega(\rho)\cdot x_i^{-1}$. On the other hand, we have $\check{\delta} = \mathrm{O}\big(\log(n)\cdot\sqrt{\log(n)/n}\big)\cdot\min(x_i^{-1},\rho^{-1}) \leq \mathrm{O}\big(\sqrt{(\log n)^3/n}\big)\cdot x_i^{-1} = \mathrm{O}(\rho)\cdot x_i^{-1}$. This finishes the proof. ◀

3.3 Wrapping up the Analysis

We now have the tools to prove the three key lemmas mentioned before (which immediately imply Theorem 1). We first show that after $\mathrm{O}\big(\log\log_{\gamma(\bar{\boldsymbol{x}})} n\big)$ phases, (w.h.p.) all opinions $i \geq 2$ are negligible and the configuration is still alive.

▶ **Lemma 7.** *Consider an initial configuration $\bar{\boldsymbol{x}}$ that is alive and for which $\psi(\bar{\boldsymbol{x}}) = \omega(\rho)$. Define $\tau_1 \coloneqq \log_{3/2}\log_{\gamma(\bar{\boldsymbol{x}})} n$. Then*

$$\Pr\left[\bigcap_{i=2}^{k}\big(X_i(\tau_1,0) \leq \rho\big) \wedge \|\boldsymbol{X}(\tau_1,0)\|_1 \geq 1-e^{-1} \;\middle|\; \boldsymbol{X}(1,0) = \bar{\boldsymbol{x}}\right] \geq 1 - n^{-2}\,.$$

Proof. Fix a phase $\tau \in \mathbb{N}$ and let $\boldsymbol{x}$ denote the configuration at the beginning of phase τ. Assume $\boldsymbol{x}$ to be alive, biased, and 1 to be the plurality opinion. Let $\gamma \coloneqq \min(x_1/\rho, \gamma(\boldsymbol{x}))$. By Claim 5, with probability at least $1-n^{-3}$ we have $\|\boldsymbol{X}(\tau+1,0)\|_1 \geq 1-e^{-1}$. Combined with Claim 6 via a union bound over all opinions we get

$$\Pr\left[\gamma(\boldsymbol{X}(\tau+1,0)) \leq \gamma^{3/2} \vee \|\boldsymbol{X}(\tau+1,0)\|_1 < 1-e^{-1} \;\middle|\; \boldsymbol{X}(\tau,0) = \boldsymbol{x}\right] \leq 2n^{-3}\,. \quad (22)$$

Thus, as long as there is at least one non-negligible opinion $i \geq 2$, (w.h.p.) the relative plurality gap increases exponentially and the configuration stays alive. Moreover, note that $\boldsymbol{X}(\tau+1,0)$ being alive and the increased relative gap between x_1 and $\max(\rho, x_i)$ for *any* other opinion i implies $\psi(\boldsymbol{X}(\tau+1,0)) = \omega(\rho)$. Thus, we can iterate this argument. To this end, for any $\tau \in \mathbb{N}$ define the event

$$\mathcal{E}_{\tau+1} \coloneqq \Big(\gamma(\boldsymbol{X}(\tau+1,0)) > \gamma_\tau^{3/2}\Big) \wedge (\|\boldsymbol{X}(\tau+1,0)\|_1 \geq 1-e^{-1}) \wedge \psi(\boldsymbol{X}(\tau+1,0)) = \omega(\rho)\,, \quad (23)$$

where $\gamma_\tau \coloneqq \min(X_1(\tau,0)/\rho, \gamma(\boldsymbol{X}(\tau,0)))$. Above we proved $\Pr[\mathcal{E}_{\tau+1} \mid \mathcal{E}_\tau, \boldsymbol{X}(\tau,0) = \boldsymbol{x}] \geq 1-2n^{-3}$. Using the definition of conditional probability, we get $\Pr[\bigcap_{\tau\leq\tau_1}\mathcal{E}_\tau] \geq \big(1-2n^{-3}\big)^{\tau_1} \geq 1-n^{-2}$, where we used (by Assumption (1)) $\tau_1 \leq 2\log\frac{\log n}{\log\gamma(\bar{\boldsymbol{x}})} \leq 2\log\frac{\log n}{\log(1+\psi(\bar{\boldsymbol{x}}))} \leq 2\log\frac{2\log n}{\psi(\bar{\boldsymbol{x}})} = \mathrm{o}(\log n)$. Finally, our choice of τ_1 guarantees $\gamma(\bar{\boldsymbol{x}})^{(3/2)^{\tau_1}} = n \geq \rho^{-1}$, such that all opinions are negligible at the start of phase τ_1. ◀

The next lemma shows that once we are in a configuration that is alive and where all opinions $i \geq 2$ are negligible, (w.h.p.) all these negligible opinions vanish within a constant number of additional phases.

▶ **Lemma 8.** *Consider a configuration $\boldsymbol{x}$ that is alive and for which $x_i \leq \rho$ for all $i \neq 1$. Define $\tau_2 \coloneqq \tau_1 + 3$. Then* $\Pr\left[\sum_{i=2}^{k} X_i(\tau_2, 0) = 0 \wedge \|\boldsymbol{X}(\tau_2, 0)\|_1 \geq 1 - e^{-1} \;\middle|\; \boldsymbol{X}(\tau_1, 0) = \boldsymbol{x}\right] \geq 1 - n^{-2}$.

Proof. Applying Claim 6 yields $\Pr\left[\bigcup_{i\neq 1}\left(\frac{X_1(\tau_1+1,0)}{X_i(\tau_1+1,0)} \leq \gamma_\rho^{3/2}\right) \;\middle|\; \boldsymbol{X}(\tau_1, 0) = \boldsymbol{x}\right] \leq n^{-3}$, where $\gamma_\rho = x_1/\rho$ as in Claim 6. Since $\|\boldsymbol{x}\|_1 \geq 1 - e^{-1}$ but $x_i \leq \rho$ for all $i \geq 2$, we must have $x_1 = \Omega(1)$ (or $\|\boldsymbol{x}\|_1 \leq k \cdot \rho + \mathrm{o}(1) = \mathrm{o}(1)$ would contradict $\boldsymbol{x}$ being alive). In particular, we get $\gamma_\rho = \Omega(1/\rho)$. Combining this with Claim 5 via a union bound, with probability at least $1 - 2n^{-3}$ configuration $\boldsymbol{X}(\tau_1 + 1, 0)$ is alive and $X_i(\tau_1 + 1, 0) < \gamma_\rho^{-3/2} = \mathrm{O}(\rho^{3/2})$ for all $i \geq 2$. Now consider the cleanup round of phase $\tau_1 + 1$ for an opinion $i \geq 2$ with $X_i(\tau_1 + 1, 0) = \mathrm{O}(\rho^{3/2})$. The probability that even one node of such an opinion remains decided after the cleanup round is at most $X_i(\tau_1+1, 0) \cdot n \cdot X_i(\tau_1+1, 0) = \mathrm{O}(\rho^3 n)$. Repeating this for a constant number of phases (note that we can use our high probability bounds to guarantee that the configuration stays alive and the plurality gap high enough) and applying the geometric distribution, we get that the probability for an opinion $i \geq 2$ to survive c more phases is at most $\mathrm{O}(\rho^{3c} n)$. The probability that even one of the $k - 1 \leq \sqrt{n}$ negligible opinions survives these c phases is $\mathrm{O}(\rho^{3c} \cdot n^{3/2}) = \mathrm{o}(n^{-(c-1)\cdot 5/4})$. The claim's statement follows for $c = 2$. ◀

Our last lemma shows that once we reached a configuration that is alive and only the plurality opinion is left, (w.h.p.) all nodes adopt the plurality in $\mathrm{O}(\log \log n)$ phases.

▶ **Lemma 9.** *For $\boldsymbol{x}$ with $\sum_{i\geq 2} x_i = 0$ and $\|\boldsymbol{x}\|_1 = x_1 \geq 1 - e^{-1}$ let $\tau_3 \coloneqq \tau_2 + \ln \log n + 1 = \tau_2 + \mathrm{O}(\log \log n)$. Then* $\Pr[X_1(\tau_3, 0) = 1 \mid \boldsymbol{X}(\tau_2, 0) = \boldsymbol{x}] \geq 1 - n^{-2}$.

Proof. We will show that the number of undecided nodes decreases exponentially from phase to phase, until their number is so low that (w.h.p.) they vanish within a constant number of decision-accumulation rounds. To this end, we use two basic high probability bounds: for cleanup rounds of a phase τ on any configuration $\boldsymbol{x'}$ with $\sum_{i\geq 2} x'_i = 0$ and $\|\boldsymbol{x'}\|_1 = x'_1 \geq 2/3$, we use

$$\Pr\left[Y(\tau, 1) \geq \max(\rho^2, 2.05 \cdot y') \;\middle|\; \boldsymbol{X}(\tau, 0) = \boldsymbol{x'}\right] \leq n^{-a}. \tag{24}$$

To see this, first note that[2] we have $\Pr[Y(\tau, 1) \geq (1 + \delta x'_1) \cdot y' \cdot (1 + x'_1) \mid \boldsymbol{X}(\tau, 0) = \boldsymbol{x'}] \leq n^{-a}$ for $\delta' \coloneqq \sqrt{13a \cdot \log n/(x'_1 y' \cdot n)}$ and $\delta \coloneqq \max(\delta', \delta'^2)$. Equation (24) then follows by distinguishing whether $y' \geq \rho^2/4$ or not.

To show the exponential decrease of the undecided nodes, assume we are given a configuration $\boldsymbol{x''}$ with $y'' < \max(\rho^2, 2.05 \cdot y')$ after the cleanup round of a phase τ. We apply Claim 4(b) to the first $t_* \coloneqq 5 \leq T - 1$ decision-accumulation rounds of phase τ and use a union bound to get $\Pr\left[Y(\tau, 1 + t_*) \geq \max\left(\rho^2, y''^{(3/2)^{t_*}}\right) \;\middle|\; \boldsymbol{X}(\tau, 1) = \boldsymbol{x''}\right] \leq t_* \cdot n^{-a}$. Since $y' \leq e^{-1}$, we get $(2.05 \cdot y')^{(3/2)^5} \leq y'^2$. Combining these observations with Equation (24)

[2] This follows exactly like Claim 4(a) when switching x_i for y' and y for x'_1. If only one opinion is left, undecided nodes in cleanup rounds increase exactly as decided nodes in decision-accumulation rounds.

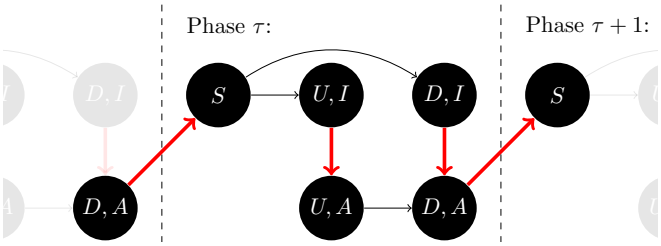


Figure 1 Illustration how nodes transition through phases. Red (thick) state transitions highlight slow transitions that ensure that nodes do not get out of sync too much.

we get $\Pr\left[Y(\tau+1,0) \geq \max\left(\rho^2, y'^2\right) \mid \boldsymbol{X}(\tau,0) = \boldsymbol{x'}\right] \leq 6n^{-a}$. In particular, the resulting configuration is still alive at the start of phase $\tau+1$, so we can iterate to get

$$\Pr\left[Y(\tau_2+t,0) \geq \max\left(\rho^2, y^{2^t}\right) \mid \boldsymbol{X}(\tau_2,0) = \boldsymbol{x}\right] \leq 6t \cdot n^{-a}\,. \tag{25}$$

Since $y \leq e^{-1}$, for $t = \ln\ln n$ we get $\max\left(\rho^2, y^{2^t}\right) = \rho^2$. Now, consider a configuration $\boldsymbol{x'}$ with $y' < \rho^2$. Equation (24) yields $\Pr\left[Y(\tau_2+t,1) \geq 2.05\rho^2 \mid \boldsymbol{X}(\tau_2+t,0) = \boldsymbol{x'}\right] \leq n^{-a}$. If $Y(\tau_2+t,1) < 2.05\rho^2$, then the probability that even one undecided node remains undecided after the first decision-accumulation round is at most $Y(\tau_2+t,1)n \cdot Y(\tau_2+t,1) \leq 5\rho^4 n \leq n^{-3/4}$. Similar to the proof of Lemma 8, we boost this probability using a geometric random variable by considering the first $4 \leq T-1$ consecutive decision-accumulation rounds, such that $\Pr\left[Y(\tau_2+t,5) \geq 0 \mid \boldsymbol{X}(\tau_2+t,0) = \boldsymbol{x'}\right] \leq n^{-a} + n^{-3}$. Combined with Equation (25), we get the desired statement. ◀

4 Plurality Consensus with $\log k + 4$ Bits

The non-constant memory overhead of RepeatedCleanup is due to the round counter used to synchronize phases. We now present a protocol that avoids this counter. Each node u stores its opinion $\mathrm{op}_u \in [k]$, a *phase counter* $\mathrm{p}_u \in \mathbb{N}$, and a *state variable* $\mathrm{s}_u \in \{S, (U,I), (D,I), (U,A), (D,A)\}$. Our description and analysis assume p_u to be an arbitrary integer. While this would result in a non-constant memory overhead, we will prove that (w.h.p.) $|\mathrm{p}_u - \mathrm{p}_v| \leq 1$ for any two nodes u, v and any round. Thus, the actual implementation can restrict p_u to $\{0,1,2\}$, such that we get a memory overhead of $\log(3 \cdot 5) \leq 4$ bits. We call this protocol ConstOverhead. Our main result is the following theorem:

▶ **Theorem 10.** ConstOverhead *has a local memory overhead of* 4 *bits (15 states). If started on a biased configuration* $\bar{\boldsymbol{x}}$ *with relative plurality gap* $\gamma := \gamma(\bar{\boldsymbol{x}})$*, then (w.h.p.) plurality consensus is achieved in* $\mathrm{O}\left(\log n \cdot \log\log_\gamma n\right)$ *rounds.*

The rest of this section describes the protocol as well as the underlying idea and gives a sketch of the analysis. More details can be found in the full version.

Protocol Overview. Initially, each node u starts with $\mathrm{p}_u = 1$ and $\mathrm{s}_u = S$. We call u *undecided* if $\mathrm{s}_u = (U,\cdot)$, *decided* if $\mathrm{s}_u = (D,\cdot)$, *inactive* if $\mathrm{s}_u = (\cdot,I)$, and *active* if $\mathrm{s}_u = (\cdot,A)$. At the start of a round, each node u samples a random node v and uses v's data to transition through its phase. Listing 2 gives the formal protocol description and Figure 1 an illustration.

We start with a high level description of the protocol which we gradually refine. Basically, ConstOverhead mimics the synchronized behavior of RepeatedCleanup. As in our first protocol, there are two ways to transition through a phase: if a node u samples another

```
1  sample a random node v
2  if s_u = S:                                                    {cleanup}
3      if p_v ≥ p_u or (p_v = p_u − 1 and s_v = (D, A)):
4          if op_v = op_u:   s_u ← (D, I)
5          else:             s_u ← (U, I)
6  elseif s_u ∈ { (U, I), (D, I) }:                               {slow transition}
7      if (p_v = p_u and v active) or p_v > p_u or Ber(1/n) = 1:
8          become active
9  elseif s_u = (U, A):                                           {decision−accumulation}
10     if (p_v = p_u and v decided) or p_v > p_u:
11         op_u ← op_v and become decided
12 elseif s_u = (D, A):                                           {slow transition}
13     if p_v > p_u or Ber(1/n) = 1:   (p_u, s_u) ← (p_u + 1, S)
```

Listing 2 Protocol ConstOverhead as executed by node u in round $t \in \mathbb{N}$. This description assumes p_u to be an arbitrary integer.

node v of the same opinion, it keeps its opinion and has to wait for the other nodes to catch up. While RepeatedCleanup implemented this waiting via a counter, ConstOverhead uses *slow transitions* (highlighted red/thick in Figure 1). Otherwise, if u samples a node v of a different opinion, it becomes undecided and keeps sampling nodes until it finds a new opinion.

Becoming decided or undecided is modeled by entering the *inactive decided* state (D, I) or *inactive undecided* state (U, I), respectively. This transition from S to one of these inactive states corresponds to cleanup rounds of RepeatedCleanup. Now, there is a slow transition to the *active decided* state (D, A) and *active undecided* state (U, A). This transition from inactive to active ensures that (a) decided nodes do not enter the next phase too early (which could require a large phase counter) and (b) undecided nodes do not sample decided nodes too early (which could result in a skewed distribution, since the number of decided nodes might be too small). Once an undecided node u becomes active, it keeps sampling nodes until it finds a new opinion. This corresponds to decision-accumulation rounds of RepeatedCleanup. Note that the transition from (D, A) to S (at which the phase counter is increased) is slow. This ensures that not too many nodes enter the next phase before all undecided nodes found a new opinion (which could, as before, require a large phase counter).

Some Subtleties. While the above reflects the basic behavior of our protocol, we omitted some details. Let us make a few important and useful observations:

- A node u always checks whether the sampled node v is not too far behind.
- Nodes do not explicitly forget their opinion when becoming undecided but simply overwrite their old opinion when they find a new opinion. In particular, if a node from an earlier state asks for u's opinion while u is undecided, u answers with its most recent opinion (the provision of which does not cost us extra in terms of memory as we simply retain the information where it was; the undecided state is seperately encoded in the already accounted-for $\Theta(1)$ additional bits).
- Slow transitions $s \to s'$ between two states s and s' basically simulate Pull-rumor spreading [18]: Nodes in state s' or later are "informed", while all other nodes are "uninformed". When u samples an informed node (a node in s' or beyond), it can cross the slow transition (independent of the sampled opinion). The Bernoulli trial $\mathrm{Ber}(1/n)$ in slow transitions ensures that, eventually, at least one node is "informed". Without it, no node could cross such a transition.

Analysis Overview. Given that ConstOverhead is based on the same principle as RepeatedCleanup, namely employing synchronized cleanup rounds to increase the relative plurality gap exponentially from phase to phase, it is natural to use a similar analysis. The major difficulty stems from the fact that our synchronization primitive is now probabilistic (slow transitions) instead of deterministic (counter). In particular, in the case of ConstOverhead there is no guarantee that nodes wait at slow transitions for other nodes to catch up; in fact, there will typically be a few nodes that proceed early on over slow transitions. This might disturb our analysis in two ways: (a) if nodes could proceed arbitrarily far ahead, their phase counter could become arbitrarily high, resulting in a non-constant memory overhead, and (b) if a small group of nodes with a non-plurality opinion were "lucky" and proceeded fast, these nodes might cause more and more latecomers to adopt a non-plurality opinion.

The major tool to address both of these issues are two probabilistic synchronization results. The first shows that (w.h.p.) there is a period/stage of $\mathrm{O}(\log n)$ consecutive rounds such that:

- If $n - \mathrm{polylog}(n)$ nodes are in state (D, A) of phase $\tau - 1$ or state S of phase τ, and all remaining nodes are in one of the two inactive states $(\cdot, I)$ of phase τ,
- then at the end of this period, $n - \mathrm{polylog}(n)$ nodes are in one of the two inactive states $(\cdot, I)$ of phase τ, and all remaining nodes are in one of the two active states $(\cdot, A)$ of phase τ.

The second result shows that (w.h.p.) there is a period/stage of $\mathrm{O}(\log n)$ consecutive rounds such that:

- If $n - \mathrm{polylog}(n)$ nodes are in one of the two inactive states $(\cdot, I)$ of phase τ, and all remaining nodes are in one of the two active states $(\cdot, A)$ of phase τ,
- then at the end of this period, $n - \mathrm{polylog}(n)$ nodes are in state (D, A) of phase τ or state S of phase $\tau + 1$, and all remaining nodes are in one of the two inactive states $(\cdot, I)$ of phase τ.

Note that the final condition of the first stage fits perfectly into the assumption of the second stage and vice versa. We call the first stage the *cleanup stage* and the second stage the *decision-accumulation stage* (in the style of the corresponding round in RepeatedCleanup). Since these stages are well-sparated, we can prove an analogue of Claim 3 and an analogue of Claim 4. Equipped with these, the remainder of the proof is basically identical to the proof of RepeatedCleanup (which was completely based on these concentration bounds).

References

1. Mohammed Amin Abdullah and Moez Draief. Majority consensus on random graphs of a given degree sequence. *CoRR*, abs/1209.5025, 2012. URL: `http://arxiv.org/abs/1209.5025`.
2. Dan Alistarh, Rati Gelashvili, and Milan Vojnovic. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 47–56, 2015. `doi:10.1145/2767386.2767429`.
3. Dana Angluin, James Aspnes, and David Eisenstat. A simple population protocol for fast robust approximate majority. *Distributed Computing*, 21(2):87–102, 2008. `doi:10.1007/s00446-008-0059-z`.
4. Arta Babaee and Moez Draief. Distributed multivalued consensus. *The Computer Journal*, 57(8):1132–1140, 2014. `doi:10.1093/comjnl/bxt026`.
5. Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, and Riccardo Silvestri. Plurality consensus in the gossip model. In *Proceedings of the 26th*

Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pages 371–390, 2015. doi:10.1137/1.9781611973730.27.

6 Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Stabilizing consensus with many opinions. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 620–635, 2016. doi:10.1137/1.9781611974331.ch46.

7 P. Berenbrink, T. Friedetzky, P. Kling, F. Mallmann-Trenn, and C. Wastell. Plurality Consensus via Shuffling: Lessons Learned from Load Balancing. *ArXiv e-prints*, February 2016. arXiv:1602.01342.

8 Colin Cooper, Robert Elsässer, Hirotaka Ono, and Tomasz Radzik. Coalescing random walks and voting on graphs. In *Proceedings of the 31st ACM Symposium on Principles of Distributed Computing (PODC)*, pages 47–56, 2012. doi:10.1145/2332432.2332440.

9 Colin Cooper, Robert Elsässer, and Tomasz Radzik. The power of two choices in distributed voting. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 435–446, 2014. doi:10.1007/978-3-662-43951-7_37.

10 Colin Cooper, Tomasz Radzik, Nicolas Rivera, and Takeharu Shiraga. Coalescing walks on rotor-router systems. In *Proceedings of the 22nd International Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 444–458, 2015. doi:10.1007/978-3-319-25258-2_31.

11 Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler. Stabilizing consensus with the power of two choices. In *Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 149–158, 2011. doi:10.1145/1989493.1989516.

12 Moez Draief and Milan Vojnović. Convergence speed of binary interval consensus. *SIAM Journal on Control and Optimisation*, 50(3):1087–1109, 2012.

13 Robert Elsässer, Tom Friedetzky, Dominik Kaaser, Frederik Mallmann-Trenn, and Horst Trinker. Efficient k-party voting with two choices. *CoRR*, abs/1602.04667, 2016. URL: http://arxiv.org/abs/1602.04667.

14 Mohsen Ghaffari and Merav Parter. A polylogarithmic gossip algorithm for plurality consensus. In *Proceedings of the 35th ACM Symposium on Principles of Distributed Computing (PODC)*, 2016. to appear.

15 Y. Hassin and D. Peleg. Distributed probabilistic polling and applications to proportionate agreement. *Information and Computation*, 171(2):248–268, 2001. doi:10.1006/inco.2001.3088.

16 R. Holley and T. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *The Annals of Probability*, 3(4):643–663, 1975. URL: http://www.jstor.org/stable/2959329.

17 K. Jung, B. Y. Kim, and M. Vojnovic. Distributed ranking in networks with limited memory and communication. In *Proc. of the 2012 IEEE Int'l Symposium on Information Theory Proceedings (ISIT)*, pages 980–984, 2012. doi:10.1109/ISIT.2012.6284710.

18 Richard M. Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vöcking. Randomized rumor spreading. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 565–574, 2000. doi:10.1109/SFCS.2000.892324.

19 Etienne Perron, Dinkar Vasudevan, and Milan Vojnovic. Using three states for binary consensus on complete graphs. In *Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM)*, pages 2527–2535, 2009. doi:10.1109/INFCOM.2009.5062181.

Fast, Robust, Quantizable Approximate Consensus*

Bernadette Charron-Bost[1], Matthias Függer[2], and Thomas Nowak[3]

1 **CNRS, École polytechnique, Paris, France**
`charron@lix.polytechnique.fr`
2 **Max-Planck-Institut für Informatik, Saarbrücken, Germany**
`mfuegger@mpi-inf.mpg.de`
3 **Université Paris-Sud, Paris, France**
`thomas.nowak@lri.fr`

Abstract

We introduce a new class of distributed algorithms for the approximate consensus problem in dynamic rooted networks, which we call *amortized averaging algorithms*. They are deduced from ordinary averaging algorithms by adding a value-gathering phase before each value update. This results in a drastic drop in decision times, from being exponential in the number n of processes to being polynomial under the assumption that each process knows n. In particular, the *amortized midpoint algorithm* is the first algorithm that achieves a linear decision time in dynamic rooted networks with an optimal contraction rate of $1/2$ at each update step.

We then show robustness of the amortized midpoint algorithm under violation of network assumptions: it gracefully degrades if communication graphs from time to time are non rooted, or under a wrong estimate of the number of processes. Finally, we prove that the amortized midpoint algorithm behaves well if processes can store and send only quantized values, rendering it well-suited for the design of dynamic networked systems. As a corollary we obtain that the 2-set consensus problem is solvable in linear time in any dynamic rooted network model.

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

Keywords and phrases approximate consensus, dynamic networks, averaging algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.137

1 Introduction

This paper studies the problem of *approximate consensus*: given a set of processes, each of them with an initial real value, for any positive real number ε, every process has to achieve ε*-consensus*, i.e., decides on one value that lies in the range of the initial values and that differs from the other decision values by at most ε. This problem, which is a weakening of *exact* ($\varepsilon = 0$) *consensus*, has a large variety of applications in distributed computing or multi-agent control (e.g., clock synchronization or geometric coordination tasks) and often has to be solved in the context of dynamic networks, using local information, and under adverse constraints.

In recent work [8], we proved that approximate consensus is solvable in a dynamic network model if and only if the directed time-varying communication graph is always rooted, i.e.,

* This work was partially supported by an X-DGA contract and the Austrian Science Foundation (FWF) project SIC (P26436).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 137; pp. 137:1–137:14

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

contains a rooted spanning tree. The spanning tree can change completely from time to time. Hence no stability condition is required to solve approximate consensus.

In fact, we showed that when addressing the solvability issue of approximate consensus, it suffices to consider the class of *averaging algorithms* where each process maintains a single scalar state variable and repeatedly updates to a weighted average of the values it has just received. Indeed, the problem is solvable if and only if it can be solved by an averaging algorithm.

We proved that, if there is a positive lower bound ϱ on the weights used in averaging steps, then the averaging algorithm achieves ε-agreement in $O\big(n\varrho^{-n}\log\frac{1}{\varepsilon}\big)$ rounds where n is the number of processes. Moreover, we showed that the classical averaging algorithm, called the *equal-neighbor* algorithm, exhibits an exponentially large decision time in the case of the *Butterfly network model*.

Contribution. In this work we present a new approximate consensus algorithm that we show to be *fast*, being the first algorithm in highly dynamic networks with linear time complexity in the number of processes, *robust* to violation of network assumptions, and allowing for *quantization* of its variables – leading to a space efficient algorithm.

We list the main contributions in detail in the following:

(i) We consider the property of *ϱ-safety* for averaging algorithms originally introduced by Chazelle [12], which is a generalization of the lower bound condition on positive weights. This property focuses on the set of transmitted *values* and not on the linear *functions* (stochastic matrices) applied in the averaging steps, as done classically. It thus captures the essential properties needed for contracting the range of current values in the system. Using a graph-theoretic reduction that already played a key role in [8], we first give simple proofs of correctness and of upper bounds on the contraction rate of ϱ-safe averaging algorithms.
More importantly, this approach allows us to propose a new averaging algorithm for approximate consensus in dynamic rooted networks, that we call the *midpoint algorithm*, with an optimal contraction rate of $1/2$.

(ii) We introduce the notion of *amortization* of averaging algorithms, which consists in inserting a value-gathering phase before each averaging step. This additional phase transforms ϱ-safe averaging algorithms into "turbo versions" of themselves in that decision times pass from being exponential [8] to being polynomial in the number n of processes.
Amortization assumes that each process knows the size n of the network. Furthermore, processes ought to be able to accumulate and to relay values from the past, as opposed to averaging algorithms that are memoryless in the sense that processes deal only with fresh values. More precisely, we show that the amortized version of a ϱ-safe averaging algorithm solves approximate consensus with polynomial decision times in any dynamic rooted network, and *a priori* under the conditions that each process knows the size of the network, and can store and transmit in every message up to $n(n-1)$ scalar values. Then we combine the two above ideas in the design of the *amortized midpoint algorithm*, which achieves ε-agreement in $O\big(n\log\frac{1}{\varepsilon}\big)$ rounds. Apart its linear decision times, the positive features of this amortized averaging algorithm are that it works in anonymous networks (without process identifier) and that processes are required to store and to transmit in each message only *two* scalar values, whatever the size of the network is.

(iii) The correctness proof of amortized averaging algorithms relies on two fundamental hypotheses: common knowledge of the number n of processes and rootedness of all

occurring communication graphs. However we show that, even with an erroneous estimate on n or when communication graphs sometimes fail to be rooted, the amortized version of a ϱ-safe averaging algorithm still solves approximate consensus, albeit with larger decision time. This graceful degradation property demonstrates that the robustness of a ϱ-safe averaging algorithm carries over to some extent to its amortized versions.

(iv) Finally we show that if processes can only use variables with finite precision, then the amortized midpoint algorithm achieves ε-consensus in linear time whenever possible, i.e., when ε is larger than the variables' precision. The decision time with quantization does not degrade and remains in the same order as without quantization. The robustness to rounding errors seems to be specific to the amortized midpoint algorithm as our proof does not work for general amortized ϱ-safe averaging algorithms.

As a corollary, we get that the 2-*set consensus problem* – another weakening of exact consensus where disagreement is limited to at most two different decision values – is solvable in linear time in all dynamic rooted network models.

While the practical implications of a versatile approximate consensus algorithm which is both fast and robust are clearly appealing for the design of man-made dynamic networked systems, our results go even beyond that and are interesting also from a different, more analytic, perspective.

Many natural phenomena, like bird flocking [11], synchronization of coupled oscillators [27, 20], opinion dynamics [16], or firefly synchronization [19], can be modeled with agents that execute averaging algorithms. However, there is a large mismatch between the fast agreement times observed in nature and the theoretical worst-case times, which may be exponential [11]. In this sense, our results provide a step towards closing this gap, suggesting that the agents in these natural systems in fact operate on a different, slower, time scale than their environments, and that this inertia actually contributes positively to faster consensus.

Related work. The time complexity of approximate consensus has been extensively studied in the past. Several papers (e.g., [20, 6, 1, 5, 2]) handle the case of time-varying communication graphs and consider averaging algorithms with decision times which are exponential in the number of processes.

The first algorithm with polynomial decision times has been proposed by Olshevsky and Tsitsiklis [24], based on ideas from load balancing. They proved that their algorithm achieves ε-consensus in $O\big(n^3 \log \frac{1}{\varepsilon}\big)$ rounds with a time-varying *bidirectional* topology under the quite strong stability condition that graphs do not change during each 3 round slot $3k$, $3k+1$, $3k+2$. A refined analysis of this algorithm [22] results in an improved quadratic bound. In the meantime, Chazelle [12] proposed a remarkable averaging algorithm with cubic decision times, which works in any bidirectional connected network model.

Another approach to speed up decision times – also developed in the context of load balancing – consists in considering *second-order* algorithms where the update rule for process p at round $k+1$ involves the old value held by p at round $k-1$, in addition to the values at round k of p's neighbors; see [21]. Following this approach, Olshevsky [23] has recently proposed a linear decision time algorithm that requires each process to know n, and to be able to store and to transmit in every message two scalar values, like the amortized midpoint algorithm. Unfortunately Olshevsky's algorithm works only under the assumption of a *fixed bidirectional* communication graph. Another linear approximate consensus algorithm has been proposed in [28] in the case of a fixed topology, possibly non-bidirectional, but it assumes processes with strong computational power and large memory (processes ought to store all values of the past). The sum of these efforts makes the time-linearity of the

relatively simple amortized midpoint algorithm in arbitrarily dynamic directed anonymous networks all the more so striking.

As for the effects of quantization, they have been studied in a number of papers, but most of them assume a fixed topology to bound the time complexity; see [18, 15]. In their seminal paper [22], Nedić et al. analyzed the quantized version of their quadratic averaging algorithm for a time-varying bidirectional topology and proved that decision times with quantized values are of the same order as with continuous values.

2 Approximate consensus and averaging algorithms

We assume a distributed, round-based computational model in the spirit of the Heard-Of model [9]. A system consists in a set of processes $[n] = \{1, \dots, n\}$. Computation proceeds in *rounds*: In a round, each process sends its state to its outgoing neighbors, receives values from its incoming neighbors, and finally updates its state based. The value of the updated state is determined by a deterministic algorithm, i.e., a transition function that maps the values in the incoming messages to a new state value. Rounds are communication closed in the sense that no process receives values in round k that are sent in a round different from k.

Communications that occur in a round are modeled by a directed graph $G = ([n], E(G))$ with a self-loop at each node. The latter requirement is quite natural as a process can obviously communicate with itself instantaneously. Such a directed graph is called a *communication graph*. We denote by $\mathrm{In}_p(G)$ and $\mathrm{Out}_p(G)$ the sets of incoming and outgoing neighbors of p, respectively.

The *product* of two communication graphs G and H, denoted $G \circ H$, is the directed graph with an edge from (p, q) if there exists $r \in [n]$ such that $(p, r) \in E(G)$ and $(r, q) \in E(H)$. We easily see that $G \circ H$ is a communication graph and, because of the self-loops, $E(G) \cup E(H) \subseteq E(G \circ H)$.

A *communication pattern* is a sequence $(G(k))_{k \geqslant 1}$ of communication graphs. For a given communication pattern, $\mathrm{In}_p(k)$ and $\mathrm{Out}_p(k)$ stand for $\mathrm{In}_p(G(k))$ and $\mathrm{Out}_p(G(k))$, respectively.

Each process p has a *local state* s_p the value of which at the end of round $k \geqslant 1$ is denoted by $s_p(k)$. Process p's initial state, i.e., its state at the beginning of round 1, is denoted by $s_p(0)$. Let the *global state* at the end of round k be the collection $s(k) = (s_p(k))_{p \in [n]}$. The *execution* of an algorithm from global initial state $s(0)$, with communication pattern $(G(k))_{k \geqslant 1}$ is the unique sequence $(s(k))_{k \geqslant 0}$ of global states defined as follows: for each round $k \geqslant 1$, process p sends $s_p(k-1)$ to all the processes in $\mathrm{Out}_p(k)$, receives $s_q(k-1)$ from each process q in $\mathrm{In}_p(k)$, and computes $s_p(k)$ from the incoming messages, according to the algorithm's transition function.

When the structure of states allows each process to record and to relay information it has received during any period of K rounds for some positive integer K, we may be led to modify time-scale and to consider blocks of K consecutive rounds, called *macro-rounds*: macro-round ℓ is the sequence of rounds $(\ell - 1)K + 1, \dots, \ell K$ and the corresponding information flow graph, called *communication graph at macro-round* ℓ, is the product of the communication graphs $G((\ell - 1)K + 1) \circ \dots \circ G(\ell K)$.

2.1 Approximate consensus

A crucial problem in distributed systems is to achieve agreement among local process states from arbitrary initial local states. It is a well-known fact that this goal is not easily achievable in the context of dynamic network changes [26, 8], and restrictions on communication patterns

are required for that. We define a *network model* as a non-empty set $\mathcal{N}$ of communication graphs, those that may occur in communication patterns.

We now consider the above round-based algorithms in which the local state of process p contains two variables x_p and dec_p. Initially the range of x_p is $[0,1]$ and $\mathrm{dec}_p = \perp$ (which informally means that p has not decided yet).[1] Process p is allowed to set dec_p to the current value of x_p, and so to a value v different from $\perp$, only once; in that case we say that p *decides* v.

For any $\varepsilon \geqslant 0$, an algorithm *achieves ε-agreement with communication pattern* $(G(k))_{k\geqslant 1}$ if each execution from a global initial state as specified above and with the communication pattern $(G(k))_{k\geqslant 1}$ fulfills the following three conditions:

ε-Agreement. The decision values of any two processes are within ε.

Validity. All decided values are in the range of the initial values of processes.

Termination. All processes eventually decide.

An algorithm *solves approximate consensus* in a network model $\mathcal{N}$ if for any $\varepsilon > 0$, it achieves ε-agreement with each communication pattern formed with graphs all in $\mathcal{N}$. It *solves exact consensus* in $\mathcal{N}$ if f it achieves 0-agreement with each communication pattern with graphs in $\mathcal{N}$.

In a previous paper [8], we proved the following characterization of network models in which approximate consensus is solvable:

▶ **Theorem 1** ([8]). *The approximate consensus problem is solvable in a network model $\mathcal{N}$ if and only if each graph in $\mathcal{N}$ has a rooted spanning tree.*

2.2 Averaging algorithms

We focus on *averaging algorithms* in which at each round k, every process p updates x_p to some weighted average of the values it has just received: formally, if $m_p(k-1)$ is the minimum of the values $\{x_q(k-1) \mid q \in \mathrm{In}_p(k)\}$ received by p in round k and $M_p(k-1)$ is its maximum, then

$$m_p(k-1) \leqslant x_p(k) \leqslant M_p(k-1)\,.$$

In other words, at each round k, every process adopts a new value within the interval formed by the values of its incoming neighbors in the communication graph $G(k)$.

Since we strive for distributed implementations of averaging algorithms, weights in the average update rule of process p are required to be locally computable by p. They may depend only on the set of values received by p at round k, as is the case, for instance, with the update rule of the *mean-value algorithm*:

$$x_p(k) = \frac{1}{|V_p(k)|} \sum_{v\in V_p(k)} v \tag{1}$$

where $V_p(k) = \{x_q(k-1) \mid q \in \mathrm{In}_p(k)\}$. In contrast, even in anonymous networks, weights in the update rule for p may depend on the *multiset* of values received by p at round k counted with their multiplicities: as an example, p's update rule in the *equal-neighbor algorithm* is given by:

$$x_p(k) = \frac{1}{|\,\mathrm{In}_p(k)|} \sum_{q\in \mathrm{In}_p(k)} x_q(k-1)\,. \tag{2}$$

[1] In the case of *binary consensus*, x_p is restricted to be initially from $\{0,1\}$.

Observe that the decision rule is not specified in the above definition of averaging algorithms: the decision time immediately follows from the number of rounds that is proven to be sufficient to reach ε-agreement.

Let $\varrho \in]0, 1/2]$; an averaging algorithm is *ϱ-safe in $\mathcal{N}$* if at any round k of each of its executions with communication patterns in $\mathcal{N}$, each process adopts a new value within the interval formed by its neighbors in $G(k)$ not too close to the boundary:

$$\varrho M_p(k-1) + (1-\varrho)m_p(k-1) \leqslant x_p(k) \leqslant (1-\varrho)M_p(k-1) + \varrho\, m_p(k-1)\,.$$

Clearly if an averaging algorithm is ϱ-safe, then it is ϱ'-safe for any $\varrho' \leqslant \varrho$. We also easily check the following property of the equal-neighbor and mean-value algorithms.

▶ **Proposition 2.** *In any network model with n processes, the equal-neighbor algorithm and the mean-value algorithm are both $(1/n)$-safe.*

Finally, an averaging algorithm is *α-contracting in $\mathcal{N}$* if at each round k of any of its executions with communication patterns in $\mathcal{N}$, we have

$$\delta\big(x(k)\big) \leqslant \alpha\, \delta\big(x(k-1)\big)$$

where δ is the seminorm on $\mathbb{R}^n$ defined by $\delta(x) = \max_p(x_p) - \min_p(x_p)$.

▶ **Proposition 3.** *In any network model, every α-contracting averaging algorithm with $\alpha \in [0, 1[$ solves approximate consensus and achieves ε-agreement in $\left\lceil \log_{\frac{1}{\alpha}} \left(\frac{1}{\varepsilon}\right)\right\rceil$ rounds.*

3 Averaging algorithms, nonsplitness, and contraction rates

In a previous paper [8], we proved solvability of approximate consensus in rooted network models by a reduction to *nonsplit* network models: a directed graph is *nonsplit* if any two nodes have a common incoming neighbor. The crucial point of nonsplitness lies in the following result:

▶ **Theorem 4.** *In a nonsplit network model, a ϱ-safe averaging algorithm is $(1-\varrho)$-contracting. Thus it solves approximate consensus and achieves ε-agreement in $\left\lceil \log_{\frac{1}{1-\varrho}} \left(\frac{1}{\varepsilon}\right)\right\rceil$ rounds.*

Combined with Proposition 2, we obtain an elementary proof of the classical result [4, 7] that approximate consensus is solvable in rooted network models, which does not make use anymore of Dobrushin's coefficients of scrambling matrices [14], which can be defined as $\delta(A) = \sup_{\delta(x)\neq 0} \delta(Ax)/\delta(x)$.

As an immediate consequence of Theorem 4, we deduce that any ϱ-safe algorithm is at best $(1/2)$-contracting since by definition the safety coefficient ϱ cannot be larger than $1/2$. Indeed, in Section 4 we shall present an averaging algorithm that is $(1/2)$-safe in any nonsplit network model. But do there exist other averaging algorithms with a contraction rate less than $1/2$? As stated in the following theorem, the answer is no in the network model of nonsplit communication graphs, except in the case of two processes for which the best contraction rate is $1/3$.

▶ **Theorem 5.** *In the network model of nonsplit communication graphs with n processes, there is no averaging algorithm that is α-contracting for any $\alpha > 1/3$ if $n = 2$ and for any $\alpha > 1/2$ if $n \geqslant 3$.*

Both lower bounds in Theorem 5 are tight. Indeed the *midpoint algorithm* that we will introduce in Section 4.3 is $(1/2)$-safe, and so $(1/2)$-contracting in the network model of nonsplit communication graphs. For a two process network, we check that Algorithm 1 has a contraction rate of $1/3$.

Algorithm 1 Averaging algorithm for two processes with contraction rate 1/3

Initialization:

$x_p \in [0,1]$

In round $k \geqslant 1$ **do:**

send x_p to other process and receive x_q if $(q,p) \in E(k)$

if x_q was received **then**

  $x_p \leftarrow x_p/3 + 2x_q/3$

end if

4 Speedup by amortization

In [8] we proved the following reduction from rooted to nonsplit network models to show that averaging algorithms solve approximate consensus in exponential time.

▶ **Proposition 6** ([8]). *Every product of $n-1$ rooted graphs with n nodes and self-loops at all nodes is nonsplit.*

We next show that one can in fact push this reduction to the extreme, obtaining significantly faster algorithms: Proposition 6 and Theorem 4 suggest to consider a new type of distributed algorithms, which we call *amortized averaging algorithms*, in which each process repeatedly first collects values during $n-1$ rounds, and then computes a weighted average of values it has just collected. In other words, an amortized averaging algorithm is an averaging algorithm with the granularity of macro-rounds consisting in blocks of $n-1$ consecutive rounds.

We now make this notion more precise. First let us fix some notation. Macro-round ℓ is the sequence of rounds $(\ell-1)(n-1)+1,\dots,\ell(n-1)$. We consider algorithms for which every variable x_p is updated only at the end of macro-rounds; $x_p(\ell)$ will denote the value of x_p at the end of round $\ell(n-1)$, as no confusion can arise. Given some communication pattern $(G(k))_{k\geqslant 1}$, the communication graph at macro-round ℓ is equal to:

$$\hat{G}(\ell) = G((\ell-1)(n-1)+1) \circ \dots \circ G(\ell(n-1))\,.$$

Each process p can record the set of values it has received during macro-round ℓ, namely the set $V_p(\ell) = \{x_q(\ell-1) \mid q \in \mathrm{In}_p\left(\hat{G}(\ell)\right)\}$, but in anonymous networks, p cannot determine the set of its incoming neighbors in $\hat{G}(\ell)$. This is in contrast to networks with unique process identifiers where each process p can determine the membership of $\mathrm{In}_p\left(\hat{G}(\ell)\right)$ by piggybacking the name of the sender onto every message, and so can compute the set $W_p(\ell) = \{\left(q, x_q(\ell-1)\right) \mid q \in \mathrm{In}_p\left(\hat{G}(\ell)\right)\}$. Each process can then determine the multiset of values that it has received during a macro-round, counted with their multiplicities.

In consequence in any anonymous network, we can define the *amortized version of* an averaging algorithm $\mathcal{A}$ with weights in update rules that depend only on the sets of received values: at the end of every macro-round ℓ, each process p adopts a new value by applying the same update rule as in $\mathcal{A}$ with the macro-set $V_p(\ell)$. Based on the above discussion, this definition can be extended to averaging algorithms with update rules involving the sets of incoming neighbors when processes have unique identifiers. For instance, the amortized version of the mean-value algorithm is defined in any anonymous network while the amortized equal-neighbor algorithm requires to have unique process identifiers.

In both cases, the new value adopted by process p at the end of macro-round ℓ lies within the interval formed by the values of its incoming neighbors in the communication graph $\hat{G}(\ell)$: if $\hat{m}_p(\ell-1)$ is the minimum of the values in $V_p(\ell)$ and $\hat{M}_p(\ell-1)$ is the maximum, then

$$\hat{m}_p(\ell-1) \leqslant x_p(\ell) \leqslant \hat{M}_p(\ell-1)\,.$$

Algorithm 2 Amortized equal-neighbor algorithm

Initialization:

1: $x_p \in [0,1]$ and $W_p \leftarrow \{(p, x_p)\}$

In round $k \geqslant 1$ **do:**

2: send W_p to all processes in $\mathrm{Out}_p(k)$ and receive W_q from all processes q in $\mathrm{In}_p(k)$

3: $W_p \leftarrow \bigcup_{q \in \mathrm{In}_p(k)} W_q$

4: **if** $k \equiv 0 \mod n-1$ **then**

5: $\quad x_p \leftarrow \frac{1}{|W_p|} \sum_{(q,v) \in W_p} v$

6: $\quad W_p \leftarrow \{(p, x_p)\}$

7: **end if**

If the original averaging algorithm is ϱ-safe, then we have the additional guarantee that

$$\varrho \hat{M}_p(\ell-1) + (1-\varrho)\hat{m}_p(\ell-1) \leqslant x_p(\ell) \leqslant (1-\varrho)\hat{M}_p(\ell-1) + \varrho\, \hat{m}_p(\ell-1)\,.$$

We can then combine Proposition 6 and Theorem 4 to derive the following central result for amortized averaging algorithms.

▶ **Theorem 7.** *In any rooted network model, the amortized version of a ϱ-safe averaging algorithm solves approximate consensus and achieves ε-agreement in $(n-1)\left\lceil \log_{\frac{1}{1-\varrho}} \left(\frac{1}{\varepsilon}\right)\right\rceil$ rounds.*

In order to fix decision times, we have assumed from the beginning that each process knows the number n of processes or at least a common upper bound on n. However, regarding the *asymptotic consensus* problem, obtained by substituting limit values for decision values in the specification of approximate consensus, this common knowledge on n is actually useless for averaging algorithms.

In contrast, update rules in amortized averaging algorithms require that the number of processes in the network is known to all processes. In other words, amortized averaging algorithms are defined only under the assumption of this common knowledge in the network, even for solving asymptotic consensus. In fact, we can adapt the definition of amortized averaging algorithms to the case where n is a fixed parameter and then, because of the self-loops, obtain that Theorem 7 still holds when n is only an upper bound on the number of processes.

4.1 A quadratic-time algorithm in rooted networks with process identifiers

In [8], we proposed an approximate consensus algorithm with a quadratic decision time. The algorithm (cf. Algorithm 2) does not work in anonymous networks as it uses process identifiers so that each process can determine whether some value that it has received several times during a macro-round is originated from the same process or not.

The update rule (line 5) rewrites as:

$$x_p(\ell) = \frac{1}{\left|\mathrm{In}_p\left(\hat{G}(\ell)\right)\right\}\!\right|} \sum_{q \in \mathrm{In}_p\left(\hat{G}(\ell)\right)} x_q(\ell-1)\,.$$

Hence Algorithm 2 is actually the amortized version of the equal-neighbor algorithm.

From Proposition 2 and Theorem 7, it follows that Algorithm 2 solves approximate consensus. Moreover, because of the inequality $\log(1-a) \leqslant -a$ when $0 \leqslant a$ and because $\delta\left(x(0)\right) \leqslant 1$, it follows that if $\ell \geqslant n \log \frac{1}{\varepsilon}$, then $\delta\left(x(\ell)\right) \leqslant \varepsilon$. Hence Algorithm 2 achieves ε-agreement in $O\left(n^2 \log \frac{1}{\varepsilon}\right)$ rounds.

4.2 A quadratic-time algorithm in anonymous rooted networks

Interestingly, Algorithm 2 still works when processes just collect values without taking into account processes from which they originate.

Algorithm 3 Amortized mean-value algorithm

Initialization:

1: $x_p \in [0,1]$
2: $V_p \subseteq V$, initially $\emptyset$
In round $k \geqslant 1$ **do:**
3: send V_p to all processes in $\mathrm{Out}_p(k)$ and receive V_q from all processes q in $\mathrm{In}_p(k)$
4: $V_p \leftarrow \bigcup_{q \in \mathrm{In}_p(k)} V_q$
5: **if** $k \equiv 0 \mod n-1$ **then**
6: $\quad x_p \leftarrow \frac{1}{|V_p|} \sum_{v \in V_p} v$
7: $\quad V_p \leftarrow \emptyset$
8: **end if**

At each macro-round ℓ, the update rule in the resulting algorithm for anonymous networks (cf. Algorithm 3, line 6) coincides with the update rule in the mean-value algorithm. In other words, Algorithm 3 is the amortized version of the mean-value algorithm. Since the latter algorithm is a $(1/n)$-safe averaging algorithm (Proposition 2), we may apply Theorem 7 and obtain the following result.

▶ **Theorem 8.** *In a rooted network model of n processes, the amortized mean-value algorithm solves approximate consensus and achieves ε-agreement in $O(n^2 \log \frac{1}{\varepsilon})$ rounds.*

4.3 A linear-time algorithm in anonymous rooted networks

We improve the above quadratic upper bound on decision times with a linear amortized averaging algorithm which differs from our previous algorithm in the update rule: each process adopts the midpoint of the range of values it has received during a macro-round (cf. Algorithm 4).

Algorithm 4 Amortized midpoint algorithm

Initialization:

1: $x_p \in [0,1]$
2: $m_p \in [0,1]$, initially x_p
3: $M_p \in [0,1]$, initially x_p
In round $k \geqslant 1$ **do:**
4: send (m_p, M_p) to all processes in $\mathrm{Out}_p(k)$ and receive (m_q, M_q) from all processes q in $\mathrm{In}_p(k)$
5: $m_p \leftarrow \min\{m_q \mid q \in \mathrm{In}_p(k)\}$
6: $M_p \leftarrow \max\{M_q \mid q \in \mathrm{In}_p(k)\}$
7: **if** $k \equiv 0 \mod n-1$ **then**
8: $\quad x_p \leftarrow (m_p + M_p)/2$
9: $\quad m_p \leftarrow x_p$
10: $\quad M_p \leftarrow x_p$
11: **end if**

Let us now consider the *midpoint algorithm* that is the simple averaging algorithm with the midpoint update rule. In other words, Algorithm 4 is the amortized version of the midpoint algorithm.

By definition, the midpoint algorithm is $(1/2)$-safe. By Theorem 4, it follows that this algorithm is a $(1/2)$-contracting approximate consensus algorithm in any nonsplit network

model, with a constant decision time. This improves the linear time complexity of the equal neighbor averaging algorithm [8] for this type of network models and demonstrates that the $1/2$ lower bound in Theorem 5 for networks with $n \geqslant 3$ processes is tight.

Furthermore Theorem 7 implies that Algorithm 4 solves approximate consensus in any rooted network model with a decision time in $O(n \log \frac{1}{\varepsilon})$ rounds.

▶ **Theorem 9.** *In a rooted network model of n processes, the amortized midpoint algorithm solves approximate consensus and achieves ε-agreement in $(n-1)\lceil \log_2 \frac{1}{\varepsilon} \rceil$ rounds.*

Hence the amortized mean-value algorithm and the amortized midpoint algorithm both work in any anonymous network model, under the assumption that the number of processes is known to all processes. However while the first algorithm has quadratic decision times and requires each process to store n values per round and to transmit n values per message, the amortized midpoint algorithm solves approximate consensus in linear-time and with only a constant number of (namely, two) values per process and per message.

5 Robustness of amortized averaging algorithms

In this section, we discuss the resiliency of our amortized averaging algorithms against a wrong estimate of the number of processes or against a partial failure of the assumption that communication graphs are permanently rooted.

Firstly consider some communication pattern in which only part of communication graphs are rooted: suppose that $N-1$ communication graphs in any macro-round of $n-1$ consecutive rounds are guaranteed to be rooted. Then there are at least $n-1$ rooted communication graphs in any sequence of $L = \left\lceil \frac{n-1}{N-1} \right\rceil$ macro-rounds of length $n-1$, and every product of L communication graphs of macro-rounds is nonsplit.

Secondly suppose that the network model is indeed rooted but processes do not know the exact number n of processes and only knows an estimate N on n. Macro-rounds in the amortized averaging algorithms then consist of $N-1$ rounds (instead of $n-1$), and so the cumulative communication graphs in such macro-rounds may be not nonsplit when $N < n$. However the communication graph over any block of $L = \left\lceil \frac{n-1}{N-1} \right\rceil$ macro-rounds of length $N-1$ is nonsplit.

In both cases, the above discussion leads to introduce the notion of K-*nonsplit* network model defined as any network model $\mathcal{N}$ such that every product of K communication graphs from $\mathcal{N}$ is nonsplit. Theorem 4 can then be extended as follows:

▶ **Theorem 10.** *In a K-nonsplit network model, a ϱ-safe averaging algorithm is $(1-\varrho^K)$-contracting over each block of K consecutive rounds, i.e., for every non negative integer k, we have*

$$\delta\big(x(k+K)\big) \leqslant (1-\varrho^K)\,\delta\big(x(k)\big)\,.$$

As an immediate corollary of Theorem 10, we obtain that the amortized version of a ϱ-safe averaging algorithm is resilient against a wrong estimate of the number of processes or against a partial failure of the assumption of a rooted network model, and its decision times are multiplied by a factor in $O\big(L\varrho^{-L}\big)$. Note that the theorem measures the number of macro-rounds, and not the number of rounds.

▶ **Theorem 11.** *The amortized version of a ϱ-safe averaging algorithm solves approximate consensus even with an erroneous number of processes or with a communication pattern*

where only part of communication graphs are rooted. Moreover it achieves ε-agreement in $O\left(L\varrho^{-L} \cdot \log\left(\frac{1}{\varepsilon}\right)\right)$ macro-rounds if N denotes the estimate on process number or if only $N-1$ rounds in each block of $n-1$ consecutive rounds are guaranteed to have a rooted communication graph and where $L = \left\lceil \frac{n-1}{N-1} \right\rceil$.

6 Quantization

In this section, we take into account the additional constraint that processes can only store and transmit quantized values. This model provides a good approximation for networks with storage constraints or with finite bandwidth channels.

We include this constraint by quantizing each averaging update rule. For that, we fix some positive integer Q and choose a rounding function, denoted $[\,.\,]$, which rounds down (or rounds up) to the nearest multiple of $1/Q$. Then the quantized update rule for process p at round k writes

$$x_p(k) = \left[\sum_{q \in \mathrm{In}_p(k)} w_{qp}(k)\, x_q(k-1) \right] \tag{3}$$

where the $w_{qp}(k)$ denote the weights in the average of the original algorithm. Besides we assume that all initial values are multiples of $1/Q$.

6.1 Quantization and midpoint

Nedić et al. [22] proved that in any strongly connected network model, every quantized averaging algorithm with the update rule (3) solves exact consensus (and so approximate consensus). Because of the impossibility result for exact consensus [8], their result does not hold if the strong connectivity assumption is weakened into the one of rooted network models.

For the same reason, if $\varepsilon < 1/Q$, then ε-consensus cannot be generally achieved in a rooted network model by a quantized averaging algorithm or its amortized version. In this section, we prove that the quantization of the amortized midpoint algorithm indeed achieves $1/Q$-agreement in any rooted network model.

▶ **Theorem 12.** *In a rooted network model, quantization of the amortized midpoint algorithm achieves $1/Q$-agreement by round $(n-1)\left(\lfloor \log_2(Q-2) \rfloor + 2\right)$. Moreover in every execution, the sequence of values of every process p converges to a limit x_p^* that is a multiple of $1/Q$ in finite time, and for every pair of processes p, q, we have either $x_p^* = x_q^*$ or $\left|x_p^* - x_q^*\right| = 1/Q$.*

We see that the decision times of the quantized and the non-quantized versions of the amortized midpoint algorithm are in the same order for $\varepsilon = 1/Q$. Further, one can show that for $\varepsilon > 2/Q$, ε-agreement is achieved earlier, namely in round $(n-1)\left(\left\lfloor \log_2 \frac{Q-2}{Q\varepsilon-2} \right\rfloor + 1\right)$. When increasing precision, i.e., for $Q \to \infty$, this reduces to $O(n \log \frac{1}{\varepsilon})$.

6.2 Approximate consensus versus 2-set consensus

We now consider the *2-set consensus problem* which is another natural generalization of the consensus problem. Instead of requiring that processes agree to within any positive real-valued tolerance ε, processes have to decide on at most 2 different values:

Agreement. There are at most two different decision values.

Formally, each process starts with an input value from the set V of multiples of $1/Q$ and has to output a decision value from V in such a way that the termination and validity conditions in the consensus specification as well as the above agreement condition are satisfied.

The 2-set consensus problem naturally reduces to approximate consensus: processes round off their $1/Q$-agreement output values. Unfortunately, the use of averaging procedures to solve approximate consensus leads processes to exchange values out of the set V in the resulting 2-set consensus algorithms. The quantized amortized midpoint algorithm allows us to overcome this problem, and Theorem 12 shows that in any rooted network model, this algorithm achieves 2-set consensus in $(n-1)\left(\lfloor \log_2(Q-2) \rfloor + 2\right)$ rounds.

The above discussion shows that the 2-set consensus problem is solvable in a dynamic network model if all the communication graphs are rooted. In particular, 2-set consensus is solvable in a asynchronous complete network with a minority of faulty senders since nonsplit rounds can be implemented in this model. Combined with the impossibility result in [13] in the case of a strict majority of faulty processes, we obtain an exact characterization of the sender faulty models for which 2-set consensus is solvable in asynchronous systems if the number of processes n is odd and a small gap (namely $n/2$ faulty processes) when n is even.

Our positive result can be interestingly compared with the 2 faulty processes boundary [3, 17, 25] between possibility and impossibility of the original (and stronger) 2-set consensus problem [10] where decision values ought to be initial values, instead of being in the range of the initial values. That points out the crucial role of the validity condition on the solvability of 2-set consensus.

7 Conclusion

This paper presented a linear-time approximate consensus algorithm in rooted dynamic network models. For that, we introduced the amortization technique to speed up classical averaging algorithms from exponential to polynomial time, while preserving their inherent robustness. Careful study of the properties that make averaging algorithms contracting lead us to identify the midpoint algorithm as having the optimal contraction rate of $1/2$. Central to our analysis of the amortized midpoint algorithm was the switch from the commonly employed matrix-based to a value-based view. An important property of the amortized midpoint algorithm is that it can be used almost unaltered with quantized values, which allows to decide on at most two neighboring values, thereby a fortiori solving 2-set consensus. Interestingly, it only needs to store and send two values in each round, which makes it space-efficient and viable for implementation.

References

1 David Angeli and Pierre-Alexandre Bliman. Stability of leaderless discrete-time multi-agent systems. *Mathematics of Control, Signals, and Systems*, 18(4):293–322, 2006.

2 Pierre-Alexandre Bliman, Angelia Nedic, and Asuman E. Ozdaglar. Rate of convergence for consensus with delays. In *Proceedings of the 47th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC)*, pages 2226–2231. IEEE, New York City, 2008.

3 Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 91–100. ACM, New York City, 1993.

4 Ming Cao, A. Stephen Morse, and Brian D. O. Anderson. Reaching a consensus in a dynamically changing environment: a graphical approach. *SIAM Journal on Control and Optimization*, 47(2):575–600, 2008.

5 Ming Cao, A. Stephen Morse, and Brian D. O. Anderson. Reaching a consensus in a dynamically changing environment: convergence rates, measurement delays, and asynchronous events. *SIAM Journal on Control and Optimization*, 47(2):601–623, 2008.

6 Ming Cao, Daniel A. Spielman, and A. Stephen Morse. A lower bound on convergence of a distributed network consensus algorithm. In Hannes Frey, Xu Li, and Stefan Rührup, editors, *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC)*, pages 2356–2361. IEEE, New York City, 2005.

7 Bernadette Charron-Bost. Orientation and connectivity based criteria for asymptotic consensus. arXiv:1303.2043v1 [cs.DC], 2013.

8 Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 528–539. Springer, Heidelberg, 2015.

9 Bernadette Charron-Bost and André Schiper. The Heard-Of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, 2009.

10 Soma Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, 1993.

11 Bernard Chazelle. The convergence of bird flocking. `arXiv:0905.4241 [cs.CG]`, 2009.

12 Bernard Chazelle. The total s-energy of a multiagent system. *SIAM Journal on Control and Optimization*, 49(4):1680–1706, 2011.

13 Roberto De Prisco, Dahlia Malkhi, and Michael K. Reiter. On k-set consensus problems in asynchronous systems. In *Proceedings of the 18th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 257–265. ACM, New York City, 1999.

14 Roland L. Dobrushin. Central limit theorem for non-stationary Markov chains I. *Theory of Probability & Its Applications*, 1(1):65–80, 1956.

15 Paolo Frasca, Ruggero Carli, Fabio Fagnani, and Sandro Zampieri. Average consensus on networks with quantized communication. *International Journal of Robust and Nonlinear Control*, 19(16):1787–1816, 2009.

16 R. Hegselmann and U. Krause. Opinion dynamics and bounded confidence models, analysis, and simulation. *Journal of artificial societies and social simulation*, 5(3):1–33, 2002.

17 Maurice Herlihy and Nir Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 111–120. ACM, New York City, 1993.

18 Akshay Kashyap, Tamer Basar, and R. Srikant. Quantized consensus. *Automatica*, 43(7):1192–1203, 2007.

19 Renato E. Mirollo and Steven H. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM Journal on Applied Mathematics*, 50(6):1645–1662, December 1990.

20 Luc Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on Automatic Control*, 50(2):169–182, 2005.

21 S. Muthukrishnan, Bhaskar Ghosh, and Martin H. Schultz. First- and second-order diffusive methods for rapid, coarse, distributed load balancing. *Theory of Computing Systems*, 31(4):331–354, 1998.

22 Angelia Nedic, Alexander Olshevsky, Asuman E. Ozdaglar, and John N. Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

23 Alex Olshevsky. Linear time average consensus on fixed graphs. *IFAC-PapersOnLine*, 48(22):94–99, 2015.

24 Alex Olshevsky and John N. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Review*, 53(4):747–772, 2011.

25 Michael E. Saks and Fotios Zaharoglou. Wait-free k-set agreement is impossible: the topology of public knowledge. In *Proceedings of the 25th ACM Symposium on Theory of Computing (STOC)*, pages 101–110. ACM, New York City, 1993.

26 Nicola Santoro and Peter Widmayer. Time is not a healer. In B. Monien and R. Cori, editors, *Proceedings of the 6th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 349 of *LNCS*, pages 304–313. Springer, Heidelberg, 1989.

27 Steven H. Strogatz. From Kuramoto to Crawford: Exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1–4):1–20, 2000.

28 Y. Yuan, G.-B. Stan, L. Shi, M. Barahona, and J. Goncalves. Decentralized minimum-time consensus. *Automatica*, 49(5):1227–1235, 2013.

Leader Election in Unreliable Radio Networks

Mohsen Ghaffari[1] and Calvin Newport[2]

1 **MIT, Cambridge, MA, USA**
ghaffari@mit.edu
2 **Georgetown University, Washington, DC, USA**
cnewport@cs.georgetown.edu

Abstract

The *dual graph* model describes a radio network that contains both reliable and unreliable links. In recent years, this model has received significant attention by the distributed algorithms community [12, 4, 8, 10, 9, 16, 1, 14]. Due to results in [10], it is known that leader election plays a key role in enabling efficient computation in this difficult setting: a leader can synchronize the network in such a manner that most problems can be subsequently solved in time similar to the classical radio network model that lacks unreliable links. The *feasibility* of efficient leader election in the dual graph model, however, was left as an important open question. In this paper, we answer this question. In more detail, we prove new upper and lower bound results that characterize the complexity of leader election in this setting. By doing so, we reveal a surprising dichotomy: (1) under the assumption that the network size n is in the range 1 to N, where N is a large upper bound on the maximum possible network size (e.g., the ID space), leader election is fundamentally hard, requiring $\tilde{\Omega}(\sqrt{N})$ rounds to solve in the worst-case; (2) under the assumption that n is in the range 2 to N, however, the problem can be solved in only $\tilde{O}(D)$ rounds, for network diameter D, matching the lower bound for leader election in the standard radio network model (within log factors) [7].

1998 ACM Subject Classification C.2.1 Wireless Communication

Keywords and phrases Radio Networks, Leader Election, Unreliability, Randomized Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.138

1 Introduction

The *dual graph* model is a generalization of the classical radio network model [5, 3] that augments a connected network topology of *reliable* links with an additional set of *unreliable* links. These latter links can come and go on a round-by-round basis as determined links. These latter links can come and go on a round-by-round basis as determined by an unknown adversarial process. The model is motivated by the observation that in real radio networks link quality can change unpredictably due to many different factors (e.g., [17]). Upper bounds proved in this general model are therefore more robust in real world deployments, while lower bounds help formalize the complexity induced by link unreliability.

The dual graph model was introduced in [11] and has since been well-studied in the distributed algorithms community [12, 4, 8, 10, 9, 16, 1, 14]. In this paper, we present new upper and lower bounds for the problem of leader election in this setting. As we elaborate below, by doing so we resolve an important open question from [10] and provide a primitive that enables efficient solutions to many problems in this difficult model.

The Power of Rerandomization. In more detail, the dual graph model describes a radio network consisting of n processes connected by two network topology graphs $G = (V, E)$

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 138; pp. 138:1–138:14

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

and $G' = (V, E')$, where the nodes in V correspond to the n processes, E describes reliable links, and E' describes unreliable links. We assume G is connected and use D to describe its diameter. We consider randomized algorithms executing in synchronous rounds. In each round, the network topology is fixed to include all edges in E and some subset of the edges in E', where the latter decision is made by an unknown adversary. Once the topology is fixed, communication follows the standard multi-hop multiple access channel rules (see Section 2).

An important assumption in the dual graph model is the strength of the adversary that selects the unreliable links to include in each round. Early work on this model assumed the adversary was *strongly adaptive* (i.e., it knows the processes' random bits), which led, not surprisingly, to pessimistic lower bounds [11, 12, 4, 8, 16]. In response to these negative results, we argued in [10] that it makes more sense to assume an *oblivious* adversary, as this still allows for unpredictable link behavior, but it removes the unrealistic assumption that link quality can somehow adapt to the processes' private bits.

To validate the claim that the oblivious dual graph model is tractable, we demonstrated that *if* you can synchronize a group of processes with new random bits generated after the execution begins (and therefore unknown to the adversary), these processes can then efficiently communicate. The high-level idea is that transmitters use the shared bits to randomly permute the broadcast probabilities used by the standard *Decay* contention management routine [3] in a manner that is hard for the adversary to predict and therefore is hard for the adversary to impede with its selection of unreliable links.

The key to enabling efficient communication in the dual graph model, therefore, is to efficiently spread shared random bits among the processes in the network. In [10], we took a first step toward this goal by showing that if you are provided a single leader in the network, this leader can generate the bits itself, and then efficiently disseminate them to all processes in $O(D \log n + \log^2(n))$ rounds (with high probability in n)—nearly matching the lower bound of $\Omega(D \log (n/D) + \log^2 n)$ for one-to-all broadcast in the standard radio network model [2, 13, 15]. The high-level idea is that as the new bits spread, the processes that learn them are synchronized and can therefore run the efficient *Decay* variation to efficiently spread them to the next hop.

This strategy reduces the challenge of efficient communication in the dual graph model to electing a single leader. *Whether or not it is possible to efficiently solve leader election in this setting, however, was left as key open question in [10].* In this paper, we answer it.

Our Results. We prove lower and upper bounds that characterize the complexity of leader election in the dual graph model with an oblivious adversary. In doing so, we reveal a surprising dichotomy: (1) under the assumption that the network size n is in the range 1 to N, where N is a large upper bound on the maximum possible network size (e.g., the ID space), leader election is fundamentally *hard*, requiring $\tilde{\Omega}(\sqrt{N})$ rounds to solve in the worst-case; (2) under the assumption that n is in the range 2 to N, however, the problem can be solved in only $\tilde{O}(D)$ rounds, matching the lower bound for leader election in the standard radio network model (within log factors) [7].

Put another way: the promise that you are not alone in the world renders the problem of leader election vastly more tractable. We are unaware of other similar settings for which the split between $n \geq 1$ and $n \geq 2$ holds such significance. In the standard radio network model, for example, the best known leader election algorithms (which nearly match the relevant lower bounds) work equally well for all network sizes [7].

Our lower bound focuses on the easier problem of *loneliness detection*, in which the goal is for each process in a network to correctly determine whether or not it is alone. We construct

a network of maximum size N that consists of $\approx \sqrt{N}$ *outsider* processes each connected to an *insider* process in G. We then connect the insider processes to satisfy the model requirement that G is connected. The graph G' describing unreliable links is fully connected. We then demonstrate how an oblivious adversary can construct a schedule for adding and removing G' edges between insider and outsider processes that delays some outsider processes from receiving messages for a long period. During this period, these outsiders cannot distinguish this execution from those in networks where they are alone. The challenge in constructing this schedule is that the insider processes might coordinate after the execution begins, creating correlated behavior that is hard to predict when constructing our G' schedule. To sidestep this problem we connect each insider process to a line of length $\approx \sqrt{N}$ such that all these lines connect at their far ends. This allows us to keep insider behavior independent for the $\Omega(\sqrt{N})$ rounds that pass before any two insider processes can learn common information.

Our upper bound, by contrast, leverages a novel strategy that necessitates that $n \geq 2$. In particular, the algorithm works in phases. In each phase, it attempts to elect a single leader by having each process flip a weighted coin and become a candidate leader if it comes up heads. For each weight tried, the processes run a series of *experiments* to detect whether they succeeded in electing a single leader.

At a high-level, in these experiments, each candidate leader begins to grow a *territory* of processes that it synchronizes with new random bits. These bits allow them to communicate efficiently using the rerandomization strategy from [10] described above. For territory A to detect if it neighbors some different territory B (indicating the election failed), processes within each territory watch the success rate of their internal communication. If the collision rate is higher than expected, this is evidence that another territory, using different random bits, exists nearby. This strategy requires that each candidate leader has at least one neighbor to recruit for this testing—generating our requirement that $n \geq 2$. We emphasize that this cooperative approach to interference detection is new in the dual graph literature.

Related Work. A close predecessor to the dual graph model of unreliable radio communication was introduced by Clementi et al. [6]. The model in its current form was identified by Kuhn et al. [11]. It has since been well-studied by the distributed algorithms community [12, 4, 8, 10, 9, 16, 1, 14]. Under the pessimistic assumption of a strongly adaptive adversary controlling the unreliable links, results are known for global broadcast [11, 12], local broadcast [8], and graph structuring algorithms [4, 16]. In [10], we suggested the assumption of an oblivious adversary and produced near optimal bounds for global broadcast. We also proved that local broadcast *could not* be solved in $o(D)$ rounds—establishing the $\tilde{O}(D)$ round strategy for enabling efficient communication presented in this paper as optimal with respect to its dependence on D. Recent work by Lynch and Newport [14] explored a different route to efficient communication in this setting by describing how to enable efficient communication in $o(D)$ time under the *additional assumption* that G and G' satisfy strong geographic constraints. The algorithm described here makes no assumption on the network topologies and the graphs used in our lower bound do not satisfy the constraints of [14].

2 Model and Problem Statement

Model. We study the dual graph model of unreliable networks with an oblivious adversary. In more detail, we model a network consisting of n processes connected by a topology consisting of both *reliable* and *unreliable* links. We describe the reliable links with a connected graph $G = (V, E)$, and the unreliable links with a graph $G' = (V, E')$. We use

D to describe the diameter of G. An algorithm in our setting assigns one computational process to each node in V, and we assume processes are provided no advance information about G or G'. In the following, we will use the terms *node* and *process* interchangeably.

Executions are divided into synchronous rounds. At the beginning of an execution, an oblivious adversary determines which edges from G' to include in the network topology for all upcoming rounds (e.g., it generates an infinite sequence of graphs, $G_1, G_2, ...$, where each $G_i = (V, E_i)$, where E_i includes every edge in E and a subset of edges from E'). The adversary does not have access to the nodes' private random bits in generating this sequence. In each round r, we call the edges in G_r the active edges for the round. At the beginning of the round, each node decides to listen or transmit. A node u receives a message m in r if and only if: (a) u decides to listen; and (b) exactly one neighbor of u in the active edges (i.e., neighbor in G_r) transmits in r and it transmits m. In this case, u learns whether the message came from a reliable neighbor. Put another way, if two or more neighbors transmit, all messages are lost at u due to collision. We assume that collisions cannot be distinguished from silence (i.e., no collision detection).

Problem Statements. The problem of *leader election* requires each node v to output a binary value b_v such that, with high probability, exactly one node outputs 1. This node is called the leader. This problem specification suffices for our lower bound. For our algorithmic result, we strengthen the problem by also requiring that each node other than the leader receive at least one message from the leader. The problem of *loneliness detection* requires that each node v outputs a binary value b'_v such that, with high probability, if v is not alone in the network, $b'_v = 0$, and if v is alone, then $b'_v = 1$.

3 The Loneliness Detection Lower Bound

Here, we show that detecting whether a node is alone in the network or not requires $\tilde{\Omega}(\sqrt{N})$ rounds, where N is the best known upper bound on n, e.g., the size of the ID space. That is, if all that nodes know about the current number n of nodes in the network is that $n \in [1, N]$, then leader election in the dual graph model requires much more than its analogue in the reliable model, where $\tilde{O}(D)$ round suffices. Note that in the reliable model, detecting loneliness can be solved easily in $O(\log^2 N)$ rounds, with high probability, using the classical Decay protocol. In contrast, as we show here, the same problem in the unreliable model requires $\tilde{\Omega}(\sqrt{N})$ rounds. Formally, we prove the following result:

▶ **Theorem 1.** *Loneliness detection in the oblivious-adversary dual graph model requires $\tilde{\Omega}(\sqrt{n})$ rounds. More specifically, there is a dual graph radio network (G, G') with at most N nodes in which, for any execution of any distributed algorithm $\mathcal{A}$, some nodes cannot distinguish this execution from one in which they are alone.*

We first start with describing the structure of the dual graph radio network (G, G') that proves this lower bound. We then provide intuitive discussions about the adversary's oblivious strategy for each given algorithm $\mathcal{A}$. At the end, we provide the formal attack strategy and prove that it ensures that some of the nodes in the network experience an execution identical to one in the setting where they are alone in the network.

The Lower Bound Network. We first explain the reliable graph G. We have $\lfloor\sqrt{N}\rfloor$ paths, each consisting of $\lfloor\frac{N-1}{\sqrt{N}}\rfloor \approx \sqrt{N}$ nodes. We call these support lines, and we label them 1, 2, ..., $\lfloor\sqrt{N}\rfloor$. For each line, we call one endpoint "right" and the other "left". We connect all

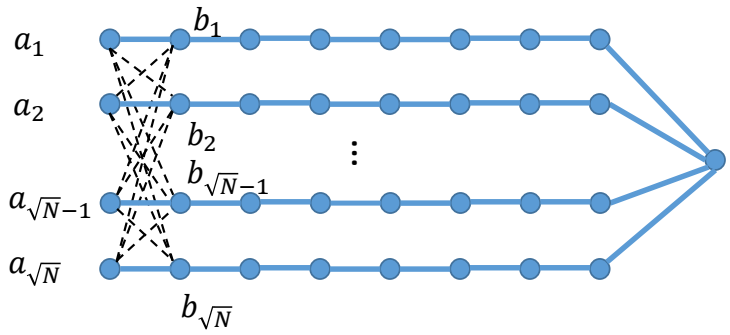


Figure 1 An illustration of the lower bound network. The reliable G-edges are depicted as solid blue lines, and the dashed black lines indicate the unreliable G'-edges.

the right endpoints to one node t. For each line i, we call the first node a_i and the second node b_i. The unreliable graph G' is then defined as a complete bipartite graph between $A = \{a_i | i \in [\lfloor\sqrt{N}\rfloor]\}$ and $B = \{b_i | i \in [\lfloor\sqrt{N}\rfloor]\}$, except for the first path edges a_i–b_i, for each path i, which are reliable and in G. For each i, we refer to a_i and b_i as pairs of each other. Figure 1 shows an illustration of this network.

3.1 Intuitive Discussions about the Adversary's Strategy

The adversary's plan is to ensure that by round $O(\sqrt{N}/\log N)$, with high probability, there are still at least half of the nodes of A which have not received any message from any other node. Thus, for each of them, the execution is identical to one in which the node is alone.

The Adversarial Strategy – Take 1. We start with an insufficient but instructive strategy. For the moment, think of b_i nodes as detached from the rest of their respective lines and let us focus on just the bipartite graph $A \times B$. Let us call a node $v \in A \cup B$ *compromised* if it has received at least one message from some other node of $A \cup B$ or some node in $A \cup B$ has received a message of v. Otherwise, we call v uncompromised.

The adversary uses its knowledge of the algorithm $\mathcal{A}$ to calculate the expected number of nodes in $A \cup B$ that will transmit in each given round. If this expectation is greater than $\Omega(\log N)$, then the adversary will aim for creating a collision for all nodes, by making all edges of G' active. The idea is that, if we have independence between the random decision of nodes in $A \cup B$ transmitting in this round, then due to this high expectation, with high probability, two or more nodes transmit. That would mean all nodes in $A \cup B$ experience collisions and none of them receives a message. Thus, in these high-expectation rounds, no nodes gets compromised, with high probability. On the other hand, if the expected number of transmitting nodes in $A \cup B$ is at most $O(\log N)$, then the adversary will work in the other extreme, and it does not activate any of the unreliable G' edges. Assuming independence between different nodes in $A \cup B$ transmitting, with high probability, at most $O(\log N)$ nodes transmit. This means, with high probability, there are at most $O(\log N)$ many a_i–b_i pairs for which some message exchange happens between a_i and b_i. Hence, in these low-expectation rounds, at most $O(\log N)$ nodes get compromised. Therefore, per round, $O(\log N)$ nodes get compromised, with high probability. That means, even after $O(\sqrt{N}/\log N)$ rounds—for small enough constants—at least half of A remain uncompromised.

The Problem in the Take 1 Strategy. The key difficulty with the above strategy is that, whether different nodes of $A \cup B$ transmit or not in a given round are not independent. This creates a challenge for concentration arguments around the expectation. For instance, there can be dependency between nodes that are compromised as there might be some communication that happened between them. Furthermore, note that the adversary cannot know which nodes are compromised in each round, as that depend on the random decisions of nodes with respect to transmitting or not.

The Adversarial Strategy – Take 2. The adversary will base its decision on the expected number X^r of uncompromised nodes that transmit, in each given round. Notice that the adversary does not know which nodes are compromised. Hence, in this expectation, whether a node is compromised or not is also a random event. Note that $X^r = \sum_i x^r_{a_i} + x^r_{b_i}$ where, for instance, $x^r_{a_i}$ is an indicator random variable for the event of a_i remaining uncompromised up to round r and then transmitting in round r.

The reason that we can focus the attack on only uncompromised nodes is that the behavior of compromised nodes will not impact the new nodes that get compromised, with high probability. Particularly, if $X^r = \Omega(\log N)$ and we have independence, then with high probability, two or more uncompromised nodes transmit. Thus activating all G' edges makes every node in $A \cup B$ experience collision. On the other hand, if $X^r = O(\log N)$ and we have independence, then with high probability, at most $O(\log N)$ uncompromised nodes transmit. Hence, if no G'-edge is activated, only nodes which their pair transmitted can be compromised. Thus, per round, at most $O(\log N)$ new nodes get compromised. Hence, even after $O(\sqrt{N}/\log N)$ rounds, at least half of the nodes in A remain uncompromised.

The Problem in the Take 2 Strategy. It remains to explain how the adversary computes the probabilities of the indicator random variables $x^r_{a_i}$ and $x^r_{b_j}$. Furthermore, the events of these variables are not precisely independent, because there is dependency between different nodes becoming compromised. For instance, if a node transmits alone in a high-expectation round in $A \cup B$, all nodes are compromised. Also, if a node gets compromised in a low-expectation round, so does its pair. We need to explain that we have sufficient independence between the random variables to use concentration arguments.

3.2 The Formal Adversarial Strategy and its Analysis

We now describe our exact adversarial strategy for determining the active edges of G' per round. We describe this process algorithmically. We later prove that for any loneliness detection algorithm $\mathcal{A}$, under this oblivious adversarial strategy, at least half of the A-nodes remain uncompromised. the schedule produced by this process works well.

In more detail, let $T = c\sqrt{n}/\log n$, for a sufficiently small constant $c > 0$, to be fixed later. As mentioned before, the adversary uses only two choices regarding which edges of G' are active: The first option is include no extra G'-edges. We call these rounds *sparse*. The second option is to include all G' edges. We call these rounds *dense*. We can, therefore, describe the adversary's behavior with a binary array S of size T, where $S[r] = 0$ indicates round r will be sparse and $S[r] = 1$ indicates it will be dense.

Oblivious Link Schedule Generation for Duration T

// `Initialization`
α is a positive constant that we use in defining our sparse/dense threshold
A is binary array of size T initialized to 0 in all positions
$N(u)$, for $u \in A \cup B$, is a real value initialized to 1
$P_r(u)$, for $u \in A \cup B$ and $r \in [T]$, is the probability node u transmits in r if it is still uncompromised at r (this depends on the definition of $\mathcal{A}$)
// `Assigning Values to Array` S
for $r = 1$ to T:
  $x \leftarrow \sum_{u \in A \cup B} N(u) \cdot P_r(u)$
  if $x \geq \alpha \log n$ **then**
    $S[r] \leftarrow 1$ // `round` r `is dense` (so we assume no compromises)
  else
    $S[r] \leftarrow 0$ // `round` r `is sparse`
    foreach $u \in A \cup B$
      $N(u) \leftarrow N(u) \cdot [(1 - P_r(u)) \cdot (1 - P_r(v)) + P_r(u) \cdot P_r(v)]$
        // `updating the probability that` u `is compromised, where` v `is the pair of` u `in the respective path.`

We must first verify that the above schedule can be generated by the oblivious adversary prior to the execution. The only non-trivial information in this effort is how the adversary generates $P_r(u)$ in advance, which we discuss next.

Computing Transmission Probabilities of Uncompromised Nodes in $A \cup B$. For $u \in A$, this follows directly from the definition of the algorithm: given the algorithm and a single node u, it is straightforward to calculate the probability that the algorithm broadcasts in any round r, given that node u is isolated.

For $u \in B$, where $u = b_i$ the adversary computes $P_r(u)$ by recursively simulating the following modified graph under a fixed pattern of delivering messages. The graph is simply the line starting from b_i and ending with the left endpoint of the path. The simulation is done (recursively, round by round) under the setting that whenever $S[r] = 1$, node b_i does not receive any message even if it is listening and its left neighbor transmits, whenever $S[r] = 0$, the reception of b_i is the same as it when this detached path is running alone. Simulating this detached line under this fixed reception strategy to b_i allows us to compute the $P_r(b_i)$ for each round $r \leq T$. This is because, if b_i is not compromised, the only difference between this simulated ditached line and the line in our network would be the connection to node t at the far right end. However, that connection cannot affect the behavior of b_i in the first $T = c\sqrt{N}/\log N$ rounds, as it is $\sqrt{N}$ hops away from b_i and due to the synchronous rounds, the causal dependency propagates with a speed of one hop per round at most.

Analysis. Next we analyze the performance of any algorithm $\mathcal{A}$ when executed in our network with the G' edges following the adversarial strategy captured by the array S as generated above.

▶ **Lemma 2.** *With high probability, in each of the first $T = c\sqrt{N}/\log N$ rounds, at most $O(\sqrt{N})$ new nodes get compromised.*

Proof. By induction on r, we show that with high probability, (1) in each dense round, all nodes in $A \cup B$ experience collision and thus no new node gets compromised, and (2) in each sparse round, at most $O(\log N)$ nodes get compromised. This directly proves the lemma.

The claim is trivially correct for round $r = 1$ because in this round, all nodes are uncompromised and $N(u) = 1$ for all of them. So, the parameter x in the adversary's calculations is the correct expectation of the number of transmitting nodes in $A \cup B$ in this round. Since we have independence between these events in the first round, Chernoff bound gives that with high probability, dense rounds are in collision for all nodes in $A \cup B$, and sparse rounds compromise at most $O(\log N)$ nodes.

Next, we prove the inductive step. Suppose that the claim is true for all rounds $r' \leq r-1$. Note that the claim implies that, modulo an inverse-polynomially small term (for the negligible change of the claim being broken), in round r, the variable $N(u)$ in the pseudo-code correctly captures the probability of node u remaining uncompromised. This is because, the calculation ignores the dense rounds, in which we know the u would not get compromised there, and it takes the transmission probability of the node and its pair into account in sparse rounds, as only the transmission of exactly one of these two nodes can compromise u.

Define N_r be random variable indicating the set of uncompromised nodes in r. Also, define the random variable X_r to be the number of uncompromised nodes in $A \cup B$ that transmit in r. Let $Y_r(u)$ be the indicator variable that is 1 if and only if u is uncompromised and transmits in r. We clearly have $E[X_r] = \sum_{u \in A \cup B} Y_r(u) = \sum_{u \in A \cup B} P_r(u) \cdot \Pr[u \in N_r]$.

In the following, we use the notation x_r to refer to the value of x in the pseudo-code run by our oblivious scheduler algorithm in round r. By the argument above, and thanks to the hypothesis of induction, we know that the computations of $N(u)$ for the probability of not being compromised was correct up to round $r-1$.

▶ **Claim 3.** *By way of induction, assume that in every dense round $r' \leq r-1$, all nodes of $A \cup B$ experience collision and thus, none of them receives a message. It follows that for each $v, u \in A \cup B$ where $(u, v) \notin G$, and each $r \leq T$, the events $Y_r(u)$ and $Y_r(v)$ are independent.*

Claim Proof. Given the inductive assumption that collisions are enforced in dense rounds $r' \leq r-1$, for each $u \in A \cup B$, we have the following: If it is in A, until u is compromised, it sees an execution identical to those in which it is alone. Also, if it is in B, until u is compromised, it experiences an execution identical to those in which the related detached line is under the reception adversary discussed above when computing $P_r(u)$. Particularly, u has received no information from outside its support line, and therefore its transmission decisions are independent of nodes in other lines.

Hence, the only way for a node to be compromised for the first time is if the node u or its pair transmit during a sparse round, in the past. However, these transmissions (prior to being compromised) are independent events between nodes of different pairs (i.e., lines). ◀

By the independence provided by the above claim, we can prove that the inductive assumption of having collisions in dense rounds is correct also for the next round r. This is because, under the inductive assumption that dense rounds prior to round r were all in collisions, the calculations $N(u)$ in the pseudo-code correctly captures the probability of a node u being uncompromised up to and including round r. This is equal to the probability of remaining uncompromised up to round $r-1$, which we know was correctly computed by indcution, and then having either none of u and its pair v transmit, or both transmit. Now, given that these events $Y_r(u)$ for different $u \in A \cup B$ are independent, except for each depending on one other event in the same set, we can finish the argument as follows: we use the standard extensions of Chernoff bound to settings with bounded (constant) dependency-graph degree and conclude the following: if the expectation $E[X_r]$ is $\Omega(\log N)$, then with high probability, $X_r \geq 2$. Hence, with high probability, the dense round r also is in collision. Similarly, if the expectation $E[X_r]$ is $O(\log N)$, then with high probability

$X_r \leq O(\log N)$ and thus, in round r, at most $O(\log N)$ new nodes get compromised. This finishes the proof of the inductive proof and thus also the proof of the lemma. ◀

4 The Algorithm

Here, we explain a distributed algorithm that computes a leader in $\tilde{O}(D)$ rounds, in the dual graph model with an oblivious adversary, when nodes are given the promise that $n \geq 2$.

▶ **Theorem 4.** *There is a distributed algorithm that in any dual graph network (G, G') with an oblivious adversary, and where $n \geq 2$ and G has diameter D, elects a leader in at most $O(D \log^4 N)$ rounds, with probability at least $1 - 1/N^c$, for any desired constant $c \geq 2$.*

4.1 Outline

Suppose that all nodes know an upper bound D on the diameter of the reliable graph G, that is within a constant factor of the diameter. We will explain how to remove this assumption using the standard doubling technique, at the end of this section.

We will try $O(\log N)$ estimates of the form $\eta = 2^i$ for n, where $i \in [1, \log N]$. For each of these, we run $O(\log N)$ experiments, where in each experiment each node is marked with probability $1/\eta = 2^{-i}$. Each marked node is considered as a candidate for leadership and an experiment is *successful* if and only if there is exactly one candidate. It is easy to see that, with high probability, there is at least one experiment in which there is exactly one marked node. The challenge is to identify experiments with exactly one candidate.

We run the $O(\log^2 N)$ experiments in parallel. Particularly, we divide time into epochs of length $O(\log^2 N)$ and the j^{th} round of each epoch belongs to the j^{th} experiment. Let us focus on one experiment. The problem we need to solve in a single experiment is to detect whether in this experiment, there is exactly one candidate or not, in $\tilde{O}(D)$ rounds.

For now, we focus on one fixed arbitrary experiment. In this experiment, we perform a special broadcast procedure starting from each of the candidates. At any point in time, we call the set of nodes that receive (only) the identifier of a candidate v the *territory* of v. Nodes that have received two or more candidate identifiers in this experiment are called *in conflict*, and nodes that have not received any candidate identifier in this experiment are called *leaderless*. We define the *conflict* and *leaderless* territories accordingly.

We first explain in Section 4.2 the simple broadcast scheme that starts from each candidate and tries to grow its territory. The guarantee will be that, if the candidate is alone, the message reaches all nodes in $O(D \log^2 N)$ rounds. In Section 4.3, we then explain how to detect whether the broadcast has already reached all nodes or not. The guarantee will be that an experiment terminates if and only if there is exactly one candidate, and in that case, it terminates in $\Theta(D \log^2 N)$ rounds. Hence, once an experiment terminates, if we wait for a (sufficiently large) constant factor more time, all those experiment that can terminate do so. At the end, among these successful experiments, the one with the smallest experiment-number wins. That means the single candidate of that experiment is elected as the leader.

4.2 Growing Territories

Each candidate v initiates a special single-message broadcast procedure. Included in this message is the identifier of the candidate v, as well as $O(\log^2 N \log \log N)$ bits of randomness. The broadcast procedure then proceeds in D phases, where in each phase, we perform the following procedure:

Rerandomized Decay Protocol. Each phase consists of $L = \alpha \log^2 N$ consecutive rounds, for a large enough constant α. In each round of the phase, all nodes that have received the message of a candidate by the start of the phase use the randomness shared in the message to pick a shared random number $j \in_{\mathcal{U}} [1, 2 \log N]$. The details of this part will be discussed shortly. Then, each node decides to transmit with probability 2^{-j}, using its own private randomness, and remains silent otherwise. At the end of the phase, each node only keeps the messages that it received from its reliable neighbors, or those that it had before. Note that the former uses the assumption that each node is able to distinguish successful receptions from reliable and unreliable neighbors. This will later simplify the process of removing the assumption of knowing an upper bound on the diameter D.

We note that the above is an adaptation of the classical Decay procedure[3], with the key change being in the publicly random choice of the transmission probability. This property helps crucially in defeating the oblivious adversary.

We show that if we have a single candidate, after $O(D \log^2 N)$ rounds, its message reaches all nodes, w.h.p. This follows immediately from the following simple lemma, which particularly implies that if there is only one candidate, its broadcast grows at a speed of one G-hop per round, thus reaching all nodes in $O(D \log^2 N)$ rounds.

▶ **Lemma 5.** *Suppose that node w has at least one G-neighbor that had received the message of a candidate v by the start of the phase. Furthermore, assume that no G'-neighbor of w has received a message from any candidate other than v. Then, regardless of the behavior of the adversary, w.h.p., node w receives the message of candidate v during this phase.*

Proof. Consider each round r of the phase and let G^r be the graph fixed by the adversary for round r. Let d^r be the number of active neighbors of w in round r that have received the message of candidate v, by the start of the phase. Notice that all potentially-active neighbors of w that transmit have received the message of candidate v and act according to the randomness bits received in the related message. Particularly, this shared randomness, they pick a common $j \in_{\mathcal{U}} [1, 2 \log N]$ for this round r. With probability $\frac{1}{2 \log N}$, the random value $j \in_{\mathcal{U}} [1, 2 \log N]$ will be such that $j = \lceil \log d^r \rceil$. If that happens, the probability that exactly one active neighbor of w transmit in round r is at least $\frac{d^r}{2^j}(1 - \frac{1}{2^j})^{d^r - 1} \geq 1/5$. Thus, per round, the probability that node w receives the message of candidate v from one of its active neighbors is at least $\frac{1}{10 \log N}$. By a simple Chernoff bound, this means that during the $\alpha \log^2 N$ rounds of the phase, node w receives the message of v with high probability. ◀

4.3 Detecting Loneliness for Candidates

Note that if in an experiment, we have two or more candidates, their territories will grow and eventually reach each other. At that point, the territories can obstruct each other from growing and covering all nodes. The key task will be to determine if the broadcast of a candidate has reached all nodes or not.

We call a node u in the territory of candidate v a *boundary node* of this territory if u has at least one G-neighbor outside the territory of v. Notice that, if the broadcast of v has not reached all nodes, then there is at least one boundary node in the territory of node v. On the other hand, if v was the only candidate, once the broadcast reaches all nodes, there will not be any boundary nodes.

The boundary nodes have the responsibility for detecting whether the broadcast has reached all nodes or not. They will deliver their indications to the candidate, which then decides whether the experiment was successful or not. The existence of boundary nodes will

be taken as an indication that the territory does not cover all nodes. Let us first explain the key idea for detecting boundary nodes.

Boundary Detection. As the lower bound from the previous section suggests, even the rudimentary problem of each node receiving a single message from any nearby node can be hard. Thus we cannot rely on the boundary nodes receiving a message from a node outside their territory. We turn the problem around. Instead of insisting on actually receiving a message from outside the territory, we take lack of a frequent receptions from the node's own territory as an indication for collisions, implying that the node is potentially a boundary node. More concretely, the following two are indications for the bound w being boundary and thus that the related broadcast has not reached all nodes: (1) if w receives a message from the territory of a different leader, or more cruicially (2) if w *does not receive* messages from its own territory frequently enough. Let us make this formal.

The boundary detection algorithm consists of a single phase—i.e., $L = \alpha \log^2 N$ rounds. Each node that has received a message from exactly one candidate v performs one phase of its Rerandomized Decay Protocol, exactly as explained above. On the other hand, all other nodes (those in conflict or leaderless) transmits a special NOISE message in all of the rounds of the phase. A node will consider itself boundary if it receives messages from its own territory in less than $\frac{\alpha \log N}{15}$ of the rounds of this phase. The next two lemmas, the second of which is the key lemma in this whole approach, explain why this criterion correctly identifies the boundary nodes:

▶ **Lemma 6.** *If a node w is not alone in its territory and it does not have a G'-neighbor out of its territory, then throughout the phase, it will receive at least $\frac{\alpha \log N}{15}$ messages from its own territory.*

Proof. The proof is similar to Lemma 5. We easily see that per round, w receives a message from its own territory with probability at least $\frac{\log N}{10}$. Thus, by a Chernoff bound, node w receives messages from its own territory in at least $\frac{\alpha \log N}{15}$ rounds, with high probability. ◀

▶ **Lemma 7.** *[Key Lemma] If w has at least one G-neighbor out of its territory, then w.h.p., it will either receive a message from a different territory, or it receives less than $\frac{\alpha \log N}{25}$ messages from its own territory.*

Proof. First note that if w has a G-neighbor in the conflict or leaderless territory, it will not receive any message from its own territory and hence it detects that it is a boundary. Next, suppose that each G-neighbor of w has received a message from exactly one marked node.

Consider a given round r of the phase, and fix the edges incident on w made active for this round by the adversary. Let A_r be the set of active neighbors of w in its own territory, and let B_r be the set of active neighbors of w in other territories. Note that by the assumption that w has at least one G-neighbor out of its territory, B_r is nonempty. Suppose that the set B_r is composed of nodes of t territories, $B_r = B_r^1 \cup B_r^2 \cup \ldots B_r^t$, where without loss of generality, B_r^t is not the conflict or leaderless territory. Define p_r to be the probability of the event that no node in B_r^1 to B_r^{t-1} transmits. We have the following simple observations regarding the transmission probabilities in these sets:

(1) The probability that no node in A_r transmits is at least $1/(2e)$.
(2) The probability that exactly one node of A_r transmits is in at most $\frac{5}{\log N}$.
(3) The probability that no node in any of B_r^1 to B_r^t transmits is at most p_r.
(4) The probability that exactly one node of B_r transmits, and it is in B_r^t, is at least $\frac{p_r}{10 \log N}$.

Note that the events of different territories are independent. Hence, by (2) and (3), we have that the probability that w receives a message from its own territory is at most $\frac{5p_r}{\log N}$. Also, by (1) and (4), we have that the probability that w receives a message from another territory is at least $\frac{p_r}{20e \log N}$.

Therefore, over all the L rounds of the phase, the expected number of messages that w receives from other territories is at least $\sum_{r=1}^{L} \frac{p_r}{20e \log N}$. If $\sum_{r=1}^{L} \frac{p_r}{20e \log N} \geq 5 \log N$, then with high probability, w receives a message from another territory. On the other hand, if $\sum_{r=1}^{L} \frac{p_r}{20e \log N} \leq 5 \log N$, the expected number of messages that w receives from its own territory is at most $\sum_{r=1}^{L} \frac{5p_r}{\log N} \ll \frac{\alpha \log N}{30}$, as the constant α is chosen to be large enough. Thus, in this case, with high probability, the number of rounds in which w receives a message from its own territory is less than $\frac{\alpha \log N}{25}$. This completes the proof. ◀

Delivering the Boundary Signal to the Candidate. If for a candidate v, there is a node u in the territory of v for which the condition of Lemma 7 is satisfied, then v knows that its experiment was not successful. We next describe how to let the candidate v know whether there is such a "boundary" node u in its territory. The basis for this again will be mimicking *collision detection*, where lack of frequent receptions from a node's own territory is regarded as a collision.

We now explain how we deliver the indicator about the existence of boundaries to the candidate, and how they react. Note that the broadcast algorithm, explained in Section 4.2, grows the territory at a speed of one G-hop per phase, when there is a single candidate, and at a (potentially) slower speed, when there are two or more. Hence, all nodes of the territory of a candidate v are within its $O(D)$ hops in G. After one phase of boundary detection, if the territory of v does not include all nodes, there will be at least one node u in this territory that satisfies the condition of Lemma 7. Thus, this node provides an excuse for not announcing the experiment successful, and potentially continuing the broadcast.

To deliver this information to the candidate, we run $O(D)$ additional phases of broadcast. Here, if a node has noticed a boundary (or is in conflict or leaderless), it constantly transmits NOISE in all the rounds of the phase. Otherwise, the node broadcasts simply according to its rerandomized decay protocol, using the shared randomness provided in the candidate's message. At the end of each phase, each node w believes that there is a boundary node in its territory if it believed so before or if receives a message from outside territory or a NOISE message, or more critically, if it does not receive messages from the node's own territory frequently enough (less than $\frac{\alpha \log N}{25}$ times). If this happens, node w constantly transmits NOISE in the rounds of the next phases. The following lemma follows by straightforward application of Lemma 7.

▶ **Lemma 8.** *If there is at least one boundary in the territory of candidate v, then by the end of these $O(D)$ phases, candidate v will be informed about it. Moreover, if the territory includes all nodes, then v will not receive such an indication.*

Finally, if after all these $\Theta(D)$ phases, a candidate v received no indication of a boundary, it considers this experiment successful. It then initiates a "*successful experiment*" announcement. Other candidate do not initiate any announcement. We run a broadcast from the candidates, in $\tilde{O}(D)$ rounds, spreading this success announcement. If the experiment was successful, then with high probability, the candidate is alone and thus, this success announcement gets delivered to all nodes of the network.

Removing the knowledge of diameter. Recall that above, we assumed that nodes know a constant factor upper bound D on the diameter of graph G. Here, we remove this assumption using a standard doubling method.

Effectively, we try 2-factor estimates of diameter. We divide the time into $\log N$ scales, where the k^{th} is made of $\tilde{O}(2^k)$ consecutive rounds. In the k^{th} scale, we imagine 2^k to be the upper bound on diameter. We thus grow the territories up to a radius of 2^k by running the broadcast (from scratch) for 2^k phases. Then we have one boundary detection phase, and then we spend 2^k additional phases to deliver the indicator about existence of boundaries to the candidate. A final set of 2^k spreads the potential announcements of successful experiments. Note that since the length of the scales form a geometric sum, and as when 2^k reaches D we will have the first successful experiment, we will observe the first successful experiment in $O(D)$ phases, considering all the scales. In the other experiments prior to this scale, the broadcasts will not be able to reach all nodes, as it spreads at a speed of one G-hop per round. Hence, those experiments will not be announced successful.

Consider the first scale for which some candidate announces a successful experiment. In this scale, we take the candidate of these successful experiments that has the smallest experiment number as the global leader.

References

1. Mohamad Ahmadi, Abdolhamid Ghodselahi, Fabian Kuhn, and Anisur Rahaman Molla. The cost of global broadcast in dynamic radio networks. In *Proceedings of the International Conference on Principles of Distributed Systems*, 2015.
2. N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A Lower Bound for Radio Broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, 1991.
3. Reuven Bar-Yehuda, Oded Goldreigch, and Alon Itai. On the Time-Complexity of Broadcast in Multi-Hop Radio Networks: An Exponential Gap between Determinism and Randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.
4. Keren Censor-Hillel, Seth Gilbert, Fabian Kuhn, Nancy Lynch, and Calvin Newport. Structuring unreliable radio networks. *Distributed Computing*, 27(1):1–19, 2014.
5. I. Chlamtac and S. Kutten. On Broadcasting in Radio Networks–Problem Analysis and Protocol Design. *IEEE Transactions on Communications*, 33(12):1240–1246, 1985.
6. A. E. F. Clementi, A. Monti, and R. Silvestri. Round Robin is Optimal for Fault-Tolerant Broadcasting on Wireless Networks. *Journal of Parallel and Distributed Computing*, 64(1):89–96, 2004.
7. Mohsen Ghaffari and Bernhard Haeupler. Near optimal leader election in multi-hop radio networks. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2013.
8. Mohsen Ghaffari, Bernhard Haeupler, Nancy Lynch, and Calvin Newport. Bounds on contention management in radio networks. In *The International Symposium on Distributed Computing (DISC)*, 2012.
9. Mohsen Ghaffari, Erez Kantor, Nancy Lynch, and Calvin Newport. Multi-message broadcast with Abstract MAC layers and unreliable links. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2014.
10. Mohsen Ghaffari, Nancy Lynch, and Calvin Newport. The cost of radio network broadcast for different models of unreliable links. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2013.
11. Fabian Kuhn, Nancy Lynch, and Calvin Newport. Brief announcement: Hardness of broadcasting in wireless networks with unreliable communication. In *Proceedings of the ACM Symposium on the Principles of Distributed Computing (PODC)*, 2009.

12 Fabian Kuhn, Nancy Lynch, Calvin Newport, Rotem Oshman, and Andrea Richa. Broadcasting in unreliable radio networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2010.

13 E. Kushilevitz and Y. Mansour. An Ω(D\log(N/D)) Lower Bound for Broadcast in Radio Networks. *SIAM Journal on Computing*, 27(3):702–712, 1998.

14 Nancy Lynch and Calvin Newport. A (truly) local broadcast layer for unreliable radio networks. In *Proceedings of the ACM Conference on Distributed Computing*, 2015.

15 Calvin Newport. Lower bounds for radio networks made easy. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, 2014.

16 Calvin Newport. Lower bounds for structuring unreliable radio networks. In *Proceedings of the International Symposium on Distributed Computing (DISC)*, 2014.

17 Calvin Newport, David Kotz, Yougu Yuan, Robert S Gray, Jason Liu, and Chip Elliott. Experimental Evaluation of Wireless Simulation Assumptions. *Simulation*, 83(9):643–661, 2007.

Faster Deterministic Communication in Radio Networks*†

Artur Czumaj[1] and Peter Davies[2]

1 **Department of Computer Science and Centre for Discrete Mathematics and its Applications, University of Warwick, Warwick, United Kingdom**
A.Czumaj@warwick.ac.uk

2 **Department of Computer Science and Centre for Discrete Mathematics and its Applications, University of Warwick, Warwick, United Kingdom**
P.W.Davies@warwick.ac.uk

Abstract

In this paper we improve the deterministic complexity of two fundamental communication primitives in the classical model of ad-hoc radio networks with unknown topology: broadcasting and wake-up. We consider an unknown radio network, in which all nodes have no prior knowledge about network topology, and know only the size of the network n, the maximum in-degree of any node Δ, and the eccentricity of the network D.

For such networks, we first give an algorithm for wake-up, in both directed and undirected networks, based on the existence of small universal synchronizers. This algorithm runs in $O(\frac{\min\{n, D\Delta\} \log n \log \Delta}{\log \log \Delta})$ time, improving over the previous best $O(n \log^2 n)$-time result across all ranges of parameters, but particularly when maximum in-degree is small.

Next, we introduce a new combinatorial framework of block synchronizers and prove the existence of such objects of low size. Using this framework, we design a new deterministic algorithm for the fundamental problem of *broadcasting*, running in $O(n \log D \log \log \frac{D\Delta}{n})$ time. This is the fastest known algorithm for this problems, improving upon the $O(n \log n \log \log n)$-time algorithm of De Marco (2010) and the $O(n \log^2 D)$-time algorithm due to Czumaj and Rytter (2003), the previous fastest results for directed networks, and is the first to come within a log-logarithmic factor of the $\Omega(n \log D)$ lower bound due to Clementi et al. (2003).

Our results have also direct implications on the fastest *deterministic leader election* and *clock synchronization* algorithms in both directed and undirected radio networks, tasks which are commonly used as building blocks for more complex procedures.

1998 ACM Subject Classification C.2.1 Distributed Networks

Keywords and phrases Radio networks, Communication networks, Broadcasting, Wake-Up, Deterministic algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.139

1 Introduction

1.1 Model of communication networks

We consider the classical model of *directed ad-hoc radio networks* with *unknown structure*. A *radio network* is modeled by a *directed* network $\mathfrak{N} = (V, E)$, where the set of nodes

* Due to space limitations, some details are deferred to the full version, available at https://arxiv.org/abs/1506.00853.

† Research partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP).

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 139; pp. 139:1–139:14

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

corresponds to the set of transmitter-receiver stations. The nodes of the network are assigned different identifiers (IDs), and throughout this paper we assume that all IDs are distinct numbers in $\{1, \ldots, |V|\}$. A directed edge $(v, u) \in E$ means that node v can send a message directly to node u. To make propagation of information feasible, we assume that every node in V is reachable in $\mathfrak{N}$ from any other.

In accordance with the standard model of unknown (ad-hoc) radio networks (for more elaborate discussion about the model, see, e.g., [1, 2, 5, 9, 10, 12, 17, 19, 22]), we make the assumption that a node does not have any prior knowledge about the topology of the network, its in-degree and out-degree, or the set of its neighbors. We assume that the only knowledge of each node is the *size* of the network n, the *maximum in-degree* of any node Δ, and the *eccentricity* of the network D, which is the maximum distance from the source node to any node in $\mathfrak{N}$. For a discussion of these assumptions, see the full version of this paper.

Nodes operate in discrete, synchronous time steps, but we do not need to assume knowledge of a global clock. When we refer to the "running time" of an algorithm, we mean the number of time steps which elapse before completion (i.e., we are not concerned with the number of calculations nodes perform within time steps). In each time step a node can either *transmit* a message to all of its out-neighbors at once or can remain silent and *listen* to the messages from its in-neighbors. We do not make any restriction on the size of messages.

The distinguishing feature of radio networks is the interfering behavior of transmissions. In the most standard radio networks model, the *model without collision detection* (see, e.g., [1, 2, 10, 22]), which is studied in this paper, if a node v listens in a given round and precisely one of its in-neighbors transmits, then v receives the message. In all other cases v receives nothing; in particular, the lack of collision detection means that v is unable to distinguish between zero of its in-neighbors transmitting and more than one.

The model without collision detection describes the most restrictive interfering behavior of transmissions; also considered in the literature is a less restrictive variant, the model with collision detection, where a node listening in a given round can distinguish between zero of its in-neighbors transmitting and more than one (see, e.g., [12, 22]).

1.2 Communications primitives: broadcasting and wake-up

In this paper we consider two fundamental communications primitives, namely *broadcasting* and *wake-up*, and consider *deterministic protocols* for each of these tasks.

1.2.1 Broadcasting

Broadcasting is one of the most fundamental problems in communication networks and has been extensively studied for many decades (see, e.g., [22] and the references therein).

The premise of the broadcasting task is that one particular node, called the *source*, has a message which must become known to all other nodes. We assume that all other nodes start in a dormant state and do not participate until they are "woken up" by receiving the source message (this is referred to in some works as the "no spontaneous transmissions" rule). As a result, while the model does not assume knowledge of a global clock, we can make this assumption in practice, since the current time can be appended to the source message as it propagates, and therefore will be known be all active nodes. This is important since it allows us to synchronize node behavior into fixed-length *blocks*.

1.2.2 Wake-up

The *wake-up problem* (see, e.g., [15]) is a related fundamental communication problem that arises in networks where there is no designated "source" node, and no synchronized time-step at which all nodes begin communicating. The goal is for all nodes to become "active" by receiving some transmission. Rather than a single source node which begins active, we instead assume that some subset of nodes spontaneously become active at arbitrary time-steps. The task can be seen as a variant of broadcast, with possibly multiple sources, and without the ability to assume a global clock. This last point is important, and results in wake-up protocols being slower than those for broadcast, since nodes cannot co-ordinate their behavior.

1.3 Related work

As a fundamental communications primitive, the task of *broadcasting* has been extensively studied for various network models for many decades.

For the model studied in this paper, directed radio networks with unknown structure and without collision detection, the first sub-quadratic deterministic broadcasting algorithm was proposed by Chlebus et al. [5], who gave an $O(n^{11/6})$-time broadcasting algorithm. After several small improvements (cf. [6, 21]), Chrobak et al. [9] designed an almost optimal algorithm that completes the task in $O(n \log^2 n)$ time, the first to be only a poly-logarithmic factor away from linear dependency. Kowalski and Pelc [17] improved this bound to obtain an algorithm of complexity $O(n \log n \log D)$ and Czumaj and Rytter [11] gave a broadcasting algorithm running in time $O(n \log^2 D)$. Finally, De Marco [20] designed an algorithm that completes broadcasting in $O(n \log n \log \log n)$ time steps. Thus, in summary, the state of the art result for deterministic broadcasting in directed radio networks with unknown structure (without collision detection) is the complexity of $O(n \min\{\log n \log \log n, \log^2 D\})$ [11, 20]. The best known lower bound is $\Omega(n \log D)$ due to Clementi et al. [10].

Broadcasting has been also studied in various related models, including undirected networks, randomized broadcasting protocols, models with collision detection, and models in which the entire network structure is known. For example, if the underlying network is undirected, then Kowalski [16] gave a deterministic broadcasting algorithm running in time $O(n \log D)$. If spontaneous transmissions are allowed and a global clock available, then deterministic broadcast can be performed in $O(n)$ time in undirected networks [5]. Randomized broadcasting has been also extensively studied, and in a seminal paper, Bar-Yehuda et al. [2] designed an almost optimal broadcasting algorithm achieving the running time of $O((D + \log n) \cdot \log n)$. This bound has been later improved by Czumaj and Rytter [11], and independently Kowalski and Pelc [18], who gave optimal randomized broadcasting algorithms that complete the task in $O(D \log \frac{n}{D} + \log^2 n)$ time with high probability, matching a known lower bound from [19]. In the model with collision detection, an $O(D + \log^6 n)$-time randomized algorithm due to Ghaffari et al. [12] is the first to exploit collisions and surpass the algorithms (and lower bound) for broadcasting without collision detection.

For more details, see e.g., [22] and the references therein.

The *wake-up problem* (see, e.g., [15]) is a related communication problem that arises in networks where there is no designated "source" node, and no synchronized time-step at which all nodes begin communicating. Before any more complex communication can take place, we must first require all nodes to be "active," i.e., aware that they should be communicating. This is the goal of wake-up, and it is a fundamental starting point for most other tasks in this setting, for example leader election and clock synchronization [8].

The first sub-quadratic deterministic wake-up protocol was given in by Chrobak et al. [8], who introduced the concept of *radio synchronizers* to abstract the essence of the problem.

They give an $O(n^{5/3} \log n)$-time protocol. Since then, there have been two improvements in running time, both making use of the radio synchronizer machinery: firstly to $O(n^{3/2} \log n)$ [4], and then to $O(n \log^2 n)$ [3]. Unlike for the problem of broadcast, the fastest known protocol for directed networks is also the fastest for undirected networks. A recent survey of the current state of research on the wake-up problem is given in [15].

1.4 Our results

In this paper we present a *new construction of universal radio synchronizers* and introduce and analyze a *new concept of block synchronizers* to improve the deterministic complexity of two fundamental communication primitives in the model of ad-hoc radio networks with unknown topology: broadcasting and wake-up.

By applying the analysis of block synchronizers, we present a new deterministic broadcasting algorithm (**Algorithm 1**) in directed ad-hoc radio networks with unknown structure, without collision detection, that for any directed network $\mathfrak{N}$ with n nodes, with eccentricity D, and maximum in-degree Δ, completes broadcasting in $O(n \log D \log\log \frac{D\Delta}{n})$ time-steps. This result almost matches a lower bound of $\Omega(n \log D)$ due to Clementi et al. [10], and improves upon the previous fastest algorithms due to De Marco [20] and due to Czumaj and Rytter [11], which require $O(n \log n \log\log n)$ and $O(n \log^2 D)$ time-steps, respectively.

Our result reveals that a non-trivial speed-up can be achieved for a broad spectrum of network parameters. Since $\Delta \leq n$, our algorithm has the complexity at most $O(n \log D \log\log D)$. Therefore, in particular, it significantly improves the complexity of broadcasting for shallow networks, where $D \ll n^{O(1)}$. Furthermore, the dependency on Δ reduces the complexity even further for networks where the product $D\Delta$ is near linear in n including sparse networks which can appear in many natural scenarios.

Our broadcasting result has also direct implications on the fastest *deterministic leader election algorithm* in directed and undirected radio networks. It is known that leader election can be completed in $O(\log n)$ times broadcasting time (see, e.g., [9, 13]) (assuming the broadcast algorithm extends to multiple sources, which is the case here as long as we have a global clock), and so our result improves the bound to achieve a deterministic leader election algorithm running in $O(n \log n \log D \log\log \frac{D\Delta}{n})$ time. For undirected networks the best result is $O(n \log^{3/2} n \sqrt{\log\log n})$ time [7] (we note that the $O(n \log D)$ broadcast protocol of [16] cannot be used at a $\log n$ slowdown for leader election, since it relies on token traversal and does not extend to multiple sources). Our result therefore favorably compares for shallow networks (for small D) even in undirected networks.

We also present a deterministic algorithm (**Algorithm 2**) for the related task of wake-up. We show the existence of universal radio synchronizers of delay $g(k) = O(\frac{n \log n \log k}{\log\log k})$, and demonstrate that this yields a wake-up protocol taking time $O(\frac{\min\{n, D\Delta\} \log n \log \Delta}{\log\log \Delta})$. This improves over the previous best result, the $O(n \log^2 n)$-time protocol of [3]; the improvement is largest when Δ is small, but even when it is polynomial in n, our algorithm is a $\log\log n$-factor faster.

Our improved result for wake-up has direct applications to communication algorithms in networks that do not have access to a global clock, where wake-up is an essential starting point for most more complex communication tasks. For example, wake-up is used as a subroutine in the fastest known protocols for fundamental tasks of *leader election* and *clock synchronization* (cf. [8]). These are two fundamental tasks in networks without global clocks, since they allow initially unsynchronized networks to be brought to a state in which synchronization can be assumed, and results from the better-understood setting with a global

clock can then be applied. Our wake-up protocol yields $O(\frac{\min\{n,D\Delta\}\log^2 n\log\Delta}{\log\log\Delta})$-time leader election and clock synchronization algorithms, which are the fastest known in both directed and undirected networks.

1.5 Previous approaches

Almost all deterministic broadcasting protocols with sub-quadratic complexity (that is, since [5]) have made use of the concept of *selective families* (or some similar variant thereof, such as selectors). These are families of sets for which one can guarantee that any subset of $[n] := \{1, 2, \ldots, n\}$ below a certain size has an intersection of size exactly 1 with some member of the family. They are useful in the context of radio networks because if the members of the family are interpreted to be the set of nodes which are allowed to transmit in a particular time-step, then after going through each member, any node with an active in-neighbor and an in-neighborhood smaller than the size threshold will be informed. Most of the recent improvements in broadcasting time have been due to a combination of proving smaller selective families exist, and finding more efficient ways to apply them (i.e., choosing which size of family to apply at which time).

One of the drawbacks of selective-family based algorithms is that applying them requires coordination between nodes. For the problem of broadcast, this means that some time may be wasted waiting for the current selective family to finish, and also that nodes cannot alter their behavior based on the time since they were informed, which might be desirable. For the problem of wake-up, this is even more of a difficulty; since we cannot assume a global clock, we cannot synchronize node behavior and hence cannot use selective families at all.

To tackle this issue, Chrobak et al. [8] introduced the concept of *radio synchronizers*. These are a development of selective families which allow nodes to begin their behavior at different times. A further extension to *universal synchronizers* in [4] allowed effectiveness across all in-neighborhood sizes. However, the adaptability to different node start times comes at a cost of increased size, meaning that synchronizer-based wake-up algorithms were slightly slower than selective family-based broadcasting algorithms .

The proofs of existence for selective families and synchronizers follow similar lines: a probabilistic candidate object is generated by deciding on each element independently at random with certain carefully chosen probabilities, and then it is proven that the candidate satisfies the desired properties with positive probability, and so such an object must exist. The proofs are all non-constructive (and therefore all resulting algorithms non-explicit; cf. Indyk [14] for an explicit construction of selective families).

Returning to the problem of broadcasting, a breakthrough came in 2010 with a paper by De Marco [20] which took a new approach. Rather than having all nodes synchronize their behavior, it instead had them begin their own unique pattern, starting immediately upon being informed. These behavior patterns were collated into a transmission matrix. The existence of a transition matrix with appropriate selective properties was then proven probabilistically. The ability for a node to transmit with a frequency which decayed over time allowed De Marco's method to inform nodes with a very large in-neighborhood faster, and this in turn reduced total broadcasting time from $O(n\log^2 D)$ [11] to $O(n\log n\log\log n)$.

A downside of this new approach is that having nodes begin immediately, rather than wait until the beginning of the next selector, gives rise to a far greater number of possible starting-time scenarios that have be accounted for during the probabilistic proof. This caused the logarithmic factor in running time to be $\log n$ rather than $\log D$. Further, the method was comparatively slow to inform nodes of low in-degree, compared to a selective family of appropriate size. These are the difficulties that our approach will have to overcome.

1.6 Overview of our approach

Our wake-up result follows a similar line to the previous works; we prove the existence of smaller universal synchronizers than previously known, using the probabilistic method. Our improvement stems from new techniques in analysis rather than method, which allow us to gain a log-logarithmic factor by choosing what we believe are the optimal probabilities by which to construct a randomized candidate.

Our broadcasting result takes a new direction, some elements of which are new and some of which can be seen as a compromise between selective family-type objects and the transmission schedules of De Marco [20]. We first note that nodes of small in-degree can be quickly dealt with by repeatedly applying $(n, \frac{n}{D})$-selective families "in the background" of the algorithm. This allows us to tailor the more novel part of the approach to nodes of large in-degree. We have nodes perform their own behavior patterns with decaying transmission frequency over time, but they are semi-synchronized to "blocks" of length roughly $\frac{n}{D}$, in order to cut down the number of circumstances we must consider. This idea is formalized by the concept of *block synchronizers,* combinatorial objects which can be seen as an extension of the radio synchronizers used for wake-up.

An important new concept used in our analysis of block synchronizers (and also in our proof of small universal synchronizers) is that of *cores*. Cores reduce a set of nodes and starting times to a (usually smaller) set of nodes which are active during a critical period. In this way we can combine many different circumstances into a single case, and demonstrate that for our purposes they all behave in the same way.

The most technically involved part of both of the proofs is the selection of the probabilities with which we generate a randomized candidate object (universal synchronizer or block synchronizer). Intuitively, when thinking about radio networks, a node in our network is aiming to inform its out-neighbors, and it should assume that as time goes on, only those with large in-neighborhoods will remain uninformed (because these nodes are harder to inform quickly). Therefore a node should transmit with ever-decreasing frequency, roughly inversely proportional to how large it estimates remaining uninformed neighbors' in-neighborhoods must be. However, these in-neighborhoods cannot be estimated exactly, and so we must tweak the probabilities slightly to cover the possible range. In block synchronizers we do this using phases of length $O(\log\log\frac{D\Delta}{n})$ during which nodes halve their transmission probability every step, but since behavior must be synchronized to achieve this we cannot do the same for radio synchronizers. Instead, we allow our estimate to be further from the true value, and require more time-steps around the same value to compensate.

As with previous results based on selective families, synchronizers, or similar combinatorial structures, the proofs of the structures we give are non-constructive, and therefore the algorithms are non-explicit.

2 Combinatorial tools

Our communications protocols rely upon the existence of objects with certain combinatorial properties, and we will separate these more abstract results from their applications to radio networks. In this section, we will define the combinatorial objects we will need to make use of, and prove their existence. Next, in Sections 3–4, we will demonstrate in details how these combinatorial objects can be used to obtain fast algorithms for broadcasting and wake-up.

2.1 Selective families

We begin with a brief discussion about *selective families*, whose importance in the context of broadcasting was first observed by Chlebus et al. [5]. A selective family is a family of subsets of $[n] := \{1, \ldots, n\}$ such that every subset of $[n]$ below a certain size has intersection of size exactly 1 with a member of the family. For the sake of consistency with successive definitions, rather than defining the family of subsets S_i, we will instead use the equivalent definition of a set of binary sequences S^v (that is, $S^v_i = 1$ if and only if $v \in S_i$).

For some $m \in \mathbb{N}$, let each $v \in [n]$ have its own length-m binary sequence $S^v = S^v_0 S^v_1 S^v_2 \ldots S^v_{m-1}$.

▶ **Definition 1.** $S = \{S^v\}_{v\in[n]}$ is an (n, k)-**selective family** if for any $X \subseteq [n]$ with $1 \leq |X| \leq k$, there exists $j \in [0, m)$ such that $\sum_{v\in X} S^v_j = 1$. (We say that such j *hits* X.)

2.1.1 Application to radio networks

During the course of radio network protocols we can "apply" a selective family S on an n-node network by having each node v transmit in time-step j if and only if v has a message it wishes to transmit and $S^v_j = 1$ (see, e.g., [5, 10]). Some previous protocols involved nodes starting to transmit immediately if they were informed of a message during the application of a selective family (or a variant called a selector designed for such a purpose), but here we will require nodes to wait until the current selective family is completed before they start participating. That is, nodes only attempt to transmit their message if they knew it at the beginning of the current application.

The result of applying an (n, k)-selective family is that any node u which has between 1 and k active neighbors before the application will be informed of a message upon its conclusion. This is because there must be some time-step j which hits the set of u's active neighbors, and therefore exactly one transmits in that time-step, so u receives a message. This method of selective family application in radio networks was first used in [5].

2.1.2 Existence of small selective families

The following standard lemma (see, e.g., [10]) posits the existence of (n, k)-selective families of size $O(k \log \frac{n}{k})$. This has been shown to be asymptotically optimal [10].

▶ **Lemma 2 (Small selective families).** *For some constant c and for any $1 \leq k \leq n$ there exists an (n, k)-selective family of size at most $m = ck \log \frac{n}{k}$.*

2.2 Radio synchronizers

Radio synchronizers are an extension of selective families designed to account for nodes in a radio network starting their behavior patterns at different times, and without access to a global clock. They were first introduced in [8] and used in an algorithm for performing wake-up, and this is also the purpose for which we will apply them.

To define radio synchronizers, we first define the concept of *activation schedule*.

▶ **Definition 3.** An n-**activation schedule** is a function $\omega : [n] \to \mathbb{N}$.

We will extend the definition to subsets $X \subseteq [n]$ by setting $\omega(X) = \min_{v\in X} \omega(v)$.

As for selective families, let each $v \in [n]$ have its own length-m binary sequence $S^v = S^v_0 S^v_1 S^v_2 \ldots S^v_{m-1}$. We then define radio synchronizers as follows:

▶ **Definition 4.** $S = \{S^v\}_{v\in[n]}$ is an (n, k, m)-**radio synchronizer** if for any activation schedule ω and for any $X \subseteq [n]$ with $1 \leq |X| \leq k$, there exists $j \in [\omega(X), \omega(X) + m)$ such that $\sum_{v\in X} S^v_{j-\omega(v)} = 1$.

One can see that the definition is very similar to that of selective families (Definition 1), except that now each v's sequence is offset by the value $\omega(v)$. To keep track of this shift in expressions such as the sum in the definition, we will call such values j *columns*. As with selective families, we say that any column j satisfying the condition in Definition 4 *hits* X.

In [4], this concept was extended to universal radio synchronizers which cover the whole range of k from 1 to n. Let $g : [n] \to \mathbb{N}$ be a non-decreasing function, which we will call the *delay* function.

▶ **Definition 5.** $S = \{S^v\}_{v\in[n]}$ is an (n, g)-**universal radio synchronizer** if for any activation schedule ω, and for any $X \subseteq [n]$, there exists column $j \in [\omega(X), \omega(X) + g(|X|))$ such that $\sum_{v\in X} S^v_{j-\omega(v)} = 1$.

2.2.1 Application of universal radio synchronizers to radio networks

One can apply universal radio synchronizers to the problem of wake-up in radio networks by having $\omega(v)$ represent the time-step in which node v becomes active during the course of a protocol (either spontaneously or by receiving a transmission). Subsequently, v interprets S^v as the pattern in which it should transmit, starting immediately from time-step $\omega(v)$. That is, in each time-step j after activation, v checks the next value in S^v (i.e., $S^v_{j-\omega(v)}$), transmits if it is **1** and stays silent otherwise. Then, the selective property specified by the definition guarantees that any node u with an in-neighborhood of size q hears a transmission within at most $g(q)$ steps of its first in-neighbor becoming active.

2.2.2 Existence of small universal radio synchronizers

We will prove existence of small universal synchronizers.

▶ **Theorem 6.** *For some constant c and for any $n \in \mathbb{N}$ there exists an (n, g)-**universal radio synchronizer** with $g(q) = \frac{cq \log q \log n}{\log \log q}$.*

The proof of Theorem 6 is deferred to the full version, but here we present its main ideas.

Our universal synchronizer must hit any set X within $g(X)$ columns of the set's start column $\omega(X)$. Our first step is to narrow down the amount of sets and activation schedules we must consider by defining the *core* of a set. The core removes elements of X which only become active after we must already have hit X (i.e. after column $\omega(X) + g(X)$), and then shifts all values of ω so that $\omega(X) = 0$. Multiple different sets X under multiple different activation schedules can have the same core, and therefore behave the same way during the critical period. So, by considering cores instead of the original sets and activation schedules, we can reduce the amount of cases we must account for.

We then probabilistically generate a candidate universal synchronizer, and prove that it does indeed satisfy the required property with positive probability. The crucial part of this method is our choice of probabilities with which we include each element in the candidate.

We expect to be able to hit a core of size q within $\frac{cq \log q \log n}{\log \log q}$ columns. Therefore, for a particular element v and for columns greater than $\omega(v) + \frac{cq \log q \log n}{\log \log q}$, all remaining cores containing v that we have left to hit should be larger than q. If we want to hit a set of size roughly q by randomly selecting each element of $[n]$, then we optimize our probability of doing so by selecting each element with probability roughly $\frac{1}{q}$. This translates to setting

$S^v_{\frac{cq\log q\log n}{\log\log q}} = 1$ with probability roughly $\frac{1}{q}$. Substituting $j = \frac{cq\log q\log n}{\log\log q}$, this works out to give the probability of $S^v_j = 1$ to be roughly $\frac{\log j\log n}{j\log\log j}$. However, as it transpires, this is not quite the optimal choice.

To see what we should choose instead, we must first examine how we analyze our candidate. We define the *load* $f_C(j)$ of a column j with respect to a core C; this is the expected number of elements of the core selected in that column. We show that the probability of hitting the core on j is at least $f_C(j)2^{-f_C(j)}$. Ideally, we want $f_C(j)$ to be constant for cores of roughly the size we should currently be hitting, and then this hitting probability is also constant. However, if we use the probabilities $\mathbf{P}[S^v_j = 1] = \frac{\log j\log n}{j\log\log j}$, we in fact find that $f_C(j)$ can be between $\Omega(1)$ and $O(\log j)$, depending upon the times at which its elements became active. This $\log j$-sized gap cannot be closed, since we can easily find cores which do indeed have loads at either end of the range. What we can do, however, is tweak the probabilities to shift this range down and maximize hitting probability over it.

Specifically, if we shift probabilities down by a factor of $\frac{\log j}{\log\log j}$, i.e. to $\mathbf{P}[S^v_j = 1] \approx \frac{\log n}{j}$, we shift the range of possible loads to be between $\Omega(\frac{\log\log j}{\log j})$ and $O(\log\log j)$, ensuring that the hitting probability is $\Omega(\frac{\log\log j}{\log j})$. This is as close to constant as we can get; the gap is what necessitates the $\frac{\log q}{\log\log q}$ factor in our delay function g.

With this bound on the probability of hitting a particular core at a particular column, we can use independence to obtain an lower bound for hitting a core over the whole range of valid columns, and then use a union bound to show that we can hit all cores with positive probability.

2.3 Block synchronizers

Next, we introduce *block synchronizers*, which are a new type of combinatorial object designed for use in a fast broadcasting algorithm. They can be seen as an extension of both radio synchronizers and the transmission matrix formulation of De Marco [20].

Let each $v \in [n]$ have its own length-m binary sequence $S^v = S^v_0 S^v_1 S^v_2 \dots S^v_{m-1}$. Define a function $\mu_B : \mathbb{N} \to \mathbb{N}$, for some fixed B, which rounds its input up to the next multiple of B, that is, $\mu_B(x) = \min\{pB : p \geq \frac{x}{B}, p \in \mathbb{N}\}$; we will call $s(v) := \mu_B(\omega(v))$ the *start column* of v. We extend s to subsets of $[n]$ in the obvious way, $s(X) = \mu_B(\omega(X))$.

▶ **Definition 7.** $S = \{S^v\}_{v\in[n]}$ is an (n, Δ, r, B)-**block synchronizer** if for any activation schedule ω and any set $X \subseteq [n]$ with $|X| \leq \Delta$, there exists column $j \in [s(X), s(X) + B\lceil\frac{|X|}{r}\rceil)$ such that $\sum_{v\in X} S^v_{j-s(v)} = 1$.

Block synchronizers differ from radio synchronizers in two ways: Firstly, on top of the offsetting effect of the activation schedule, there is also the function μ_B that effectively "snaps" behavior patterns to blocks of size B, hence the name block synchronizer. Secondly, the size of the range in which we must hit X is linearly dependent on $|X|$ rather than being fixed. The parameter r is the increment by which each block increases the size of sets we can hit.

2.3.1 Application of block synchronizers to radio networks

The idea of our broadcasting algorithm will be that any node v waits until the start of the first block after its activation time $\omega(v)$, and then begins its transmission pattern S^v. The definition of block synchronizer aims to model this scenario. The hitting condition ensures that any node with an in-neighborhood of size $q \leq \Delta$ will be informed within $B\lceil\frac{q}{r}\rceil$ time-steps of the start of the block in which its first in-neighbor begins transmitting.

2.3.2 Existence of small block synchronizers

We prove the following theorem.

▶ **Theorem 8.** *For some constant c and for any $n, D, \Delta \in \mathbb{N}$ with D, $\Delta \leq n < D\Delta$, there exists an $(n, \Delta, \frac{n}{D}, c\frac{n}{D} \log D \log\log \frac{D\Delta}{n})$-**block synchronizer**.*

The proof of this theorem is deferred to the full version, but we outline its main ideas.

We begin in a similar way to that of Theorem 6, in that we define the concept of a core to narrow down the range of possibilities we must consider. The difference is that now elements v have their behavior patterns snapped to blocks, so the core need only store the number of the first block v becomes active, rather than the exact column. This significantly reduces the number of possible cores, and is essential for obtaining the $\log D$ factor in size, rather than $\log n$. As before, cores also shift the activation schedule to begin at 0.

We wish to proceed in a similar manner as in the proof of small universal synchronizers, generating a candidate probabilistically and then proving that it satisfies the desired property with positive probability. While we could do this directly for block synchronizers using the same method, we would obtain a size of $\Theta(n \log D \log\log \Delta)$. Our improved bound of $O(n \log D \log\log \frac{D\Delta}{n})$ results from noting that we can afford to hit cores smaller than $\frac{n}{D}$ using selective families, leaving a narrower range of cores to be hit by our randomized candidate object. Therefore, we define an *upper block synchronizer* which need only hit cores above a certain size threshold (in our case, $\frac{n}{D}$), and this is the object we prove the existence of with the probabilistic method.

Once again, the most technical aspect of the proof lies in the probabilities with which we choose elements in our candidate object. Again, we define the load of a column $f_{\mathcal{C}}(j)$ to be the expected number of elements in a core $\mathcal{C}$ which are selected in column j, and show that the probability of hitting the core at j is at least $f_{\mathcal{C}}(j)2^{-f_{\mathcal{C}}(j)}$. By choosing our probabilities to be roughly inversely proportional to the size of cores we expect to remain un-hit, we can guarantee that on a constant fraction of columns we have $f_{\mathcal{C}}(j) = \Omega(1)$ and $f_{\mathcal{C}}(j) = O(\log \frac{Dj}{n}) = O(\log \frac{D\Delta}{n})$. However, the way in which we deal with this range differs from the case of universal synchronizers.

Since our elements' behavior is snapped to blocks, we can synchronize changes in probability of selection between all elements. Specifically, we have $O(\log\log \frac{D\Delta}{n})$-length phases in which the selection probability of all elements halves every consecutive column (in addition to the slight decrease that occurs between columns naturally). This means that $f_{\mathcal{C}}(j)$ also roughly halves every column within phases, and so there is at least one column within each phase in which it is within some constant range (we use the interval $(\frac{1}{3}, 1)$). Then, the probability of hitting the core at that column is also at least some constant.

With this bound, we can again use independence of columns and a union bound to show that our candidate upper block synchronizer hits all cores larger than $\frac{n}{D}$ with positive probability. Then, it remains only to insert a $(n, \frac{n}{D})$-selective family at the start of every block in order to hit smaller cores, and we meet the conditions of a block synchronizer.

3 Algorithms for broadcasting and wake-up

In this section we present our algorithms for broadcasting and wake-up in radio networks.

3.1 Broadcasting

We will assume that $D\Delta > n$, otherwise an earlier $O(D\Delta \log \frac{n}{\Delta})$-time protocol from [10] can be used to achieve $O(D\Delta \log \frac{n}{\Delta}) = O(n \log D)$ time.

Let $\mathcal{S}$ be an $(n, \Delta, \frac{n}{D}, \mathcal{B})$-block synchronizer, with $\mathcal{B} = c\frac{n}{D}\log D \log\log \frac{D\Delta}{n}$, and recall that $\mu_{\mathcal{B}}(x) = \min\{p\mathcal{B} : p \geq \frac{x}{\mathcal{B}}, p \in \mathbb{N}\}$. We will say that the source node becomes active at time-step 0, and any other node v becomes active in a time-step i if it received its first transmission at time-step $i-1$. Our broadcasting algorithm is the following (Algorithm 1):

Algorithm 1 Broadcast at a node v

Let i be the time-step in which v becomes active
for j from 0 to $DB-1$, in time-step $\mu_{\mathcal{B}}(i)+j$ **do** v transmits source message iff $\mathcal{S}^v_j = 1$
end for

3.2 Wake-up

Let S be an (n, g)-universal radio synchronizer with $g(q) = \frac{cq\log q \log n}{\log\log q}$. We will say that a node v becomes active in a time-step i if it either spontaneous wakes up at i, or received its first transmission at time-step $i-1$. Our wake-up algorithm is the following (Algorithm 2):

Algorithm 2 Wake-up at a node v

Let i be the time-step in which v becomes active
for j from 0 to $g(n)-1$, in time-step $i+j$ **do** v transmits source message iff $S^v_j = 1$
end for

4 Analysis of broadcasting and wake-up algorithms

In this section we show that our algorithms for broadcasting and wake-up have the claimed running times. We begin with the analysis of the broadcasting algorithm.

▶ **Theorem 9.** *Algorithm 1 performs broadcast in $O(n\log D\log\log\frac{D\Delta}{n})$ time-steps.*

To begin the analysis, fix some arbitrary node v and let P be a shortest path from the source (or first informed node) x to v. Number the nodes in this path consecutively, e.g., $P_0 = x$ and $P_{dist(x,v)} = v$. Classify all other nodes into *layers* dependent upon the furthest node along the path P to which they are an in-neighbor (some nodes may not be an in-neighbor to any node in P; these can be discounted from the analysis). That is, layer $L_\ell = \{u \in V : \max_{u \text{ in-neighbour to } P_i} i = \ell\}$ for $\ell \leq dist(x,v)$. We separately define layer $L_{dist(x,v)+1}$ to be $\{v\}$.

At any time step, we call a layer *leading* if it is the foremost layer containing an active node, and our goal is to progress through the network until the final layer is leading, i.e., v is active. The use of layers allows us to restrict to the set of nodes of our main interest: if we focus on the path node whose in-neighborhood contains the leading layer, we cannot have interference from earlier layers since they contain no in-neighbors of this path node, and we cannot have interference from later layers since they are not yet active.

▶ **Lemma 10.** *Let $h : [\Delta] \to \mathbb{N}$ be a non-decreasing function, and define $T(n, D, \Delta, h)$ to be the supremum of the function $\sum_{i=1}^{D} h(q_i)$, where integers $1 \leq q_i \leq \Delta$ satisfy the additional constraint $\sum_{i=1}^{D} q_i \leq n$. If a broadcast or wake-up protocol ensures that any layer (under any choice of v) of size q remains leading for no more than $h(q)$ time-steps, then all nodes become active within $T(n, D, \Delta, h)$ time-steps.*

Proof. Let $q_i = |L_i|$. Layer $L_{dist(x,v)+1}$ must be leading (and thus node v active) once no other layers are leading, and so this occurs $\sum_{i=1}^{dist(x,v)} h(q_i)$ time-steps after layer L_1

becomes leading. Since $\sum_{i=1}^{dist(x,v)} h(q_i) \leq \sum_{i=1}^{D} h(q_i)$ and $\sum_{i=1}^{D} q_i \leq n$, this is no more than $T(n, D, \Delta, h)$ time-steps. Since v was chosen arbitrarily, all nodes must be active within $T(n, D, \Delta, h)$ time-steps of x becoming active. ◀

We make use of Lemma 10 to give bounds on the running times of our algorithms:

▶ **Lemma 11.** *Algorithm 1 ensures that any layer of size q remains leading for fewer than $\mathcal{B}\lceil\frac{q+r}{r}\rceil$ time-steps.*

Proof. For all nodes w, let $\omega(w)$ be the time-step that w becomes active during the course of the algorithm. By definition of a block selector, for any layer L_i of size q_i there is a time-step $j < s(L_i) + B\lceil\frac{q_i}{r}\rceil$ in which exactly one element of L_i transmits. Then, either path node P_i hears the transmission (and so layer L_i is no longer leading in time-step $j+1$), or P_i has active in-neighbors not in L_i, in which case these must be in a later layer so L_i is not leading. Thus, L_i can remain leading for no more than $s(L_i) + \mathcal{B}\lceil\frac{q_i}{r}\rceil - \omega(L_i) < \mathcal{B}\lceil\frac{q_i+r}{r}\rceil$ time-steps. ◀

With these tools, we are now ready to complete the proof of Theorem 9.

Proof. Proof of Theorem 9 By Lemma 10, Algorithm 1 ensures that all nodes are active (and have therefore heard the source message) within $T(n, D, \Delta, h)$ time-steps, where $h(q) = \mathcal{B}\lceil\frac{q+r}{r}\rceil$. We will use an upper bound $T(n, D, \Delta, h')$, where $h'(q) = \mathcal{B}\frac{q+2r}{r}$. Since h' is linear and increasing, $\sum_{i=1}^{D} h'(q_i)$ subject to $\sum_{i=1}^{D} q_i \leq n$ is maximized whenever $\sum_{i=1}^{D} q_i = n$, for example at $q_i = \frac{n}{D}$ for all $i \in [D]$. So, the algorithm completes broadcast within

$$\sum_{i=1}^{D} h'(\frac{n}{D}) = \sum_{i=1}^{D} \mathcal{B}\frac{\frac{n}{D}+2r}{r} = 3\mathcal{B}D = 3c'n\log D\log\log\frac{D\Delta}{n}$$

time-steps. ◀

In a similar way, we can analyze Algorithm 2.

▶ **Theorem 12.** *Algorithm 2 performs wake-up in $O(\frac{\min(n,D\Delta)\log n\log\Delta}{\log\log\Delta})$ time-steps.*

5 Conclusions

The task of broadcasting in radio networks is a longstanding, fundamental problem in communication networks. Our result for deterministic broadcasting in directed networks combines elements from several of the previous works with some new techniques, and, in doing so, makes a significant improvement to the fastest known running time. Our algorithm for wake-up also improves over the previous best running time, and relies on a proof of smaller universal synchronizers, a combinatorial object first defined in [4].

Neither of these algorithms are known to be optimal. The best known lower bound for both broadcasting and wake-up is $\Omega(\min(n\log D, D\Delta\log\frac{n}{\Delta}))$ [10]; our broadcasting algorithm therefore comes within a log-logarithmic factor, but our wake-up algorithm remains a logarithmic factor away.

As well as the obvious problems of closing these gaps, there are several other open questions regarding deterministic broadcasting in radio networks. Firstly, the lower bound for undirected networks is weaker than that for directed networks [18], and so one avenue of research would be to find an $O(n\log D)$ lower bound in undirected networks, matching the broadcasting time of [16]. Secondly, the algorithms given here, along with almost all

previous work, are non-explicit, and therefore it remains an important challenge to develop explicit algorithms that can come close to the existential upper bound. The best constructive algorithm known to date is by Indyk [14], but it is a long way from optimality.

Some variants of the model also merit interest, in particular the model with collision detection. It is unknown whether the capacity for collision detection improves deterministic broadcast time, as it does for randomized algorithms [12]. Collision detection does remove the requirement of spontaneous transmissions for the use of the $O(n)$ algorithm of [5], but a synchronized global clock would still be required. It should be noted that collision detection renders the wake-up problem trivial, since if every active node transmits in every time-step, collisions will wake up the entire network within D time-steps.

References

1. N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, 1991.
2. R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.
3. B. Chlebus, L. Gąsieniec, D. R. Kowalski, and T. Radzik. On the wake-up problem in radio networks. In *Proceedings of the 32nd Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pages 347–359, 2005.
4. B. Chlebus and D. R. Kowalski. A better wake-up in radio networks. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 266–274, 2004.
5. B. S. Chlebus, L. Gąsieniec, A. Gibbons, A. Pelc, and W. Rytter. Deterministic broadcasting in unknown radio networks. *Distributed Computing*, 15(1):27–38, 2002.
6. B. S. Chlebus, L. Gąsieniec, A. Östlin, and J. M. Robson. Deterministic radio broadcasting. In *Proceedings of the 27th Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pages 717–728, 2000.
7. B. S. Chlebus, D. R. Kowalski, and A. Pelc. Electing a leader in multi-hop radio networks. In *Proceedings of the 16th International Conference on Principles of Distributed Systems (OPODIS)*, pages 106–120, 2012.
8. M. Chrobak, L. Gąsieniec, and D. R. Kowalski. The wake-up problem in multihop radio networks. *SIAM Journal on Computing*, 36(5):1453–1471, 2007.
9. M. Chrobak, L. Gąsieniec, and W. Rytter. Fast broadcasting and gossiping in radio networks. *Journal of Algorithms*, 43(2):177–189, 2002.
10. A. E. F. Clementi, A. Monti, and R. Silvestri. Distributed broadcasting in radio networks of unknown topology. *Theoretical Computer Science*, 302(1-3):337–364, 2003.
11. A. Czumaj and W. Rytter. Broadcasting algorithms in radio networks with unknown topology. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 492–501, 2003.
12. M. Ghaffari, B. Haeupler, , and M. Khabbazian. Randomized broadcast in radio networks with collision detection. In *Proceedings of the 32nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 325–334, 2013.
13. M. Ghaffari and B. Haeupler. Near optimal leader election in multi-hop radio networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 748–766, 2013.
14. P. Indyk. Explicit constructions of selectors and related combinatorial structures, with applications. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 697–704, 2002.

15 T. Jurdziński and D. R. Kowalski. The wake-up problem in multi-hop radio networks. In M. Y. Kao, editor, *Encyclopedia of Algorithms*. Springer-Verlag, Berlin, 2015.

16 D. Kowalski. On selection problem in radio networks. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 158–166, 2005.

17 D. Kowalski and A. Pelc. Faster deterministic broadcasting in ad hoc radio networks. *SIAM Journal on Discrete Mathematics*, 18:332–346, 2004.

18 D. Kowalski and A. Pelc. Broadcasting in undirected ad hoc radio networks. *Distributed Computing*, 18(1):43–57, 2005.

19 E. Kushilevitz and Y. Mansour. An $\omega(d \log(n/d))$ lower bound for broadcast in radio networks. *SIAM Journal on Computing*, 27(3):702–712, 1998.

20 G. De Marco. Distributed broadcast in unknown radio networks. *SIAM Journal on Computing*, 39(6):2162–2175, 2010.

21 G. De Marco and A. Pelc. Faster broadcasting in unknown radio networks. *Information Processing Letters*, 79:53–56, 2001.

22 D. Peleg. Time-efficient broadcasting in radio networks: A review. In *Proceedings of the 4th International Conference on Distributed Computing and Internet Technology (ICDCIT)*, pages 1–18, 2007.

Networks of Complements*†

Moshe Babaioff[1], Liad Blumrosen[2], and Noam Nisan[3]

1 **Microsoft Research, Herzelia, Israel**
`moshe@microsoft.com`
2 **Hebrew University, Jerusalem, Israel**
`blumrosen@gmail.com`
3 **Hebrew University, Jerusalem, Israel, and Microsoft Research, Herzelia, Israel**
`noam@cs.huji.ac.il`

Abstract

We consider a network of sellers, each selling a single product, where the graph structure represents pair-wise complementarities between products. We study how the network structure affects revenue and social welfare of equilibria of the pricing game between the sellers. We prove positive and negative results, both of "Price of Anarchy" and of "Price of Stability" type, for special families of graphs (paths, cycles) as well as more general ones (trees, graphs). We describe best-reply dynamics that converge to non-trivial equilibrium in several families of graphs, and we use these dynamics to prove the existence of approximately-efficient equilibria.

1998 ACM Subject Classification J.4. Economics, G.2.2 Network Problems

Keywords and phrases Complements, Pricing, Networks, Game Theory, Price of Stability

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.140

1 Introduction

Sellers typically do not operate in isolated markets but in conjunction with other sellers and buyers. In particular, sellers need to take into account the fact that buyers may value the goods they sell as substitutes or complements to goods sold by others. This has a tremendous impact on how sellers compete with each other. Indeed, in Cournot's [8] famous paper from 1838 about sellers who compete through quantities, Cournot also describes a model of a duopoly selling perfect complements, zinc and copper. In Cournot's example, a manufacturer of zinc may observe that some of her major customers produce brass (made of zinc and copper); Therefore, zinc manufacturers compete not only with other zinc sellers, but they also indirectly compete with manufacturers of copper, as both target the money of brass producers. We are interested in a more complicated competition structure, where as zinc can be also used for Galvanization of iron, zinc sellers compete at the same time with sellers of iron. In a similar way, iron is also demanded by car manufacturers that need to purchase glass from other sellers, and so on.

Another classic example is by Ellet [10], who studied how owners of two consecutive segments of a canal determine the tolls for shippers; Clearly, every shipper must purchase a permit from both owners for being granted the right to cross the canal. Another, more contemporary, example might be a high-tech or pharmaceutical firm that must use two

* The full version of this extended abstract is available at `https://arxiv.org/abs/1605.00136`.
† Liad Blumrosen and Noam Nisan were supported by by the Israeli Science Foundation grant number 230/10.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 140; pp. 140:1–140:14


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

registered patents to manufacture its product; The owners of the two patents quote prices for the usage rights, and these patents can be viewed as perfect complements for the firm. As a final example, consider an international trader who wishes to export goods from country X to country Y, and needs to pay license fees to both countries. In the last two examples, one can see how licenses have a network structure, as each patent may be needed for the production of several different products, and trade may occur between country X and country Y, but also between country X and another country Z, etc.

In this paper, we study markets where goods are complementary to several other goods, but not substitutes. Price discrimination is impossible, and sellers need to offer the same price in all markets. This situation creates a global price competition between sellers, which raises interesting questions we aim to explore: What kinds of equilibria exist in these games? How efficient are the equilibria in this game? Will natural dynamics reach highly efficient equilibria?

The structure of the market plays a central role in our analysis. We model the market using a weighted undirected graph, where each vertex represents a seller of a certain good. An edge with weight v between vertices i and j indicates that there is a buyer that is willing to pay an amount v for the bundle of goods $\{i, j\}$.[1] In the above example, a market with sellers of copper, zinc, iron and glass can be represented as a path-graph with 4 vertices (See Figure 1), where edges connect copper and zinc, zinc and iron, and iron and glass.

We study the following simultaneous full-information pricing game between sellers on a graph: the sellers observe the values of the buyers, which are common knowledge; each seller then posts a single price to all buyers; a buyer on an edge buys the two goods on this edge if the total price (the sum of the prices of the two goods) is no larger than her value. Sellers have zero manufacturing costs and unlimited supply, and the profit of each seller is the price she posted times the number of buyers that accepted this price. In a (pure) Nash equilibrium, no seller would benefit from changing its price given the prices offered by the other sellers.

There are two natural benchmarks for measuring the quality of equilibria in such games. The first is the *maximum welfare*, which is the sum of values of all buyers.[2] This is the welfare that would be achieved with zero prices for all goods. The second benchmark is the *optimal monopolist revenue*, which is the optimal total revenue achievable by a monopolist that owns all the goods in the network. (Clearly, the optimal monopolist revenue is always at most the maximum welfare.) Several papers studied the problem of computing the optimal pricing for a monopolist in our setting. The problem was proposed by [12] which showed that it is APX-hard and presented an approximation algorithm that is logarithmic in the number of buyers. A 1/4-approximation algorithm was later presented by [3] when buyers are interested in bundles of size 2, and recently [19] showed that this bound is tight under some computational assumptions. A similar algorithm for this problem was suggested by [20] in the context of setting up peering connections in networks. [18] provided an improved bound for monopolist revenue maximization with buyers having the same value but interested in one or two items. Unlike these papers which focus on the monopolist's algorithmic problem, we focus on analyzing equilibria in the game between competing sellers and on how the welfare and revenue of these equilibria approximate the above benchmarks.

---

[1] Throughout this paper, for the simplicity of presentation, we consider buyers that demand bundles of size 2 which we can model using graphs; Buyers that demand bundles of arbitrary sizes can be modeled as hyper-edges on a hyper-graph. We show how some of our results extend to hyper-graphs in the full paper.

[2] Assuming 0 production costs for sellers, and payments cancel out.

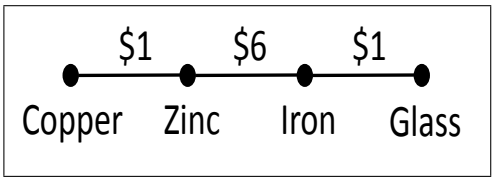


Figure 1 A market with 4 sellers and 3 buyers. The weight on each edges is the amount this buyer is willing to pay for the bundle of the two adjacent goods. For example, the middle edge represents a buyer that is willing to pay \$6 for a bundle of zinc and iron, but has value 0 for each good alone.

It is straightforward to see that some equilibria in these games demonstrate a complete market failure. For example, a vector of prices of ∞ posted by all sellers forms a Nash equilibrium that yields zero welfare and revenue. Yet, there might be multiple equilibria in these games, and one may hope that other equilibria perform better. Indeed, we prove that for some families of graphs the best equilibria ("price of stability") have high revenue and welfare. Observe that we cannot hope for full efficiency as even for path graphs, the most efficient equilibria are sometimes not fully efficient: For example, in Figure 1, if either the seller of zinc or of iron offer a price above 1, then one of its edges will not be sold. However, there is no equilibrium in which these two sellers offer prices at most 1: if this occurs, each seller gains at most 2 (from selling to its two edges) but they can get revenue of at least 5 by offering 5 and selling to the middle edge only.

Another way to alleviate the problem of low welfare and revenue of some equilibria is by restricting attention to equilibria with natural properties. For instance, it may be unreasonable for a seller that does not sell any good to insist on a very high price despite having zero costs; Such behavior might be considered as a malicious behavior towards his neighboring sellers. We therefore sometimes restrict attention to *non-malicious* equilibria, where all sellers that are not selling at all offer a price of 0. Non-malicious equilibria serve as a main tool for proving the existence of approximately-optimal equilibria: we show that the revenue of every such equilibrium approximates the maximum welfare, and we show cases where such equilibria exist.

Our Results

Our main goal is to understand how the network structure affects various properties of equilibria in markets. We aim to understand how well can the welfare and revenue in equilibrium approximate the maximum welfare and the optimal revenue. We explore whether best-response dynamics can lead to approximately-optimal equilibria, and we prove the existence of such equilibria in several settings. We study graphs of different complexities, like paths, cycles, trees and general graphs.

We start by considering some special families of graphs (paths and cycles) and show that there is a natural best response dynamics that reaches equilibria with a constant fraction of the optimal welfare as revenue. For path graphs, we consider a dynamics starting with all sellers pricing at infinity. Then, we change the price of a seller on one side of the path to zero, and let each seller in turn best reply to current prices. We show that when this best-reply process reaches the other end of the path, it ends up in an equilibrium. Moreover, for at least one of the end points, the revenue when starting from that point is at least half the maximum welfare. An extension of this dynamics for cycles converges to an equilibrium with revenue of at least one quarter of the optimal welfare.

After we prove that for some families of graphs there exist Nash equilibria with good revenue guarantees, the immediate question is whether this generalizes to more complex graphs. It turns out that non-maliciousness of equilibria can be used to prove bounds on the revenue. In particular, we show that for any tree, every non malicious equilibrium has revenue that is at most $O(\log d)$ factor smaller than the optimal welfare, where d is the maximum degree. One might hope that such a result can be achieved for general graphs as well, but unfortunately this is not the case. For a clique of degree d, the loss can be linear in d,[3] and therefore we cannot hope for a logarithmic fraction for general graphs. Our main positive result presents a refined bound that depends not only on the maximum degree, but also on the *arboricity* of the graph (see, e.g., [22]). This additional parameter nicely captures the difference between trees and cliques (and other graphs). The arboricity of an undirected graph is the minimum number of forests into which its edges can be partitioned. (For example, a tree has arboricity 1, a cycle has arboricity 2, and a clique of n nodes has arboricity $n/2$.) We prove the following bound on the revenue of any non-malicious equilibrium:[4]

▶ **Theorem.** *In every graph with maximum degree d and arboricity w and every non-malicious equilibrium in it, the total revenue of all sellers is at least $\Omega(\frac{1}{w+\log d})$ of the maximum welfare.*

The above theorem does not claim anything about the existence of non-malicious equilibria, but only bounds the revenue obtained by such equilibria if they exist. For this result to imply a bound on the "price of stability" in such games,[5] one needs to show that non-malicious equilibria actually exist. We prove the existence of non-malicious equilibria via the natural heuristic of repeated best-reply dynamics. Such a dynamics starts with arbitrary prices; At each step, a seller who is not best replying is chosen, and he updates his price to a best reply (breaking ties non-maliciously - towards 0 price). Our main result in this context shows that in tree graphs, a specific sequence of best replies does stop at a non-malicious equilibrium.

▶ **Theorem.** *In every tree, for every initial profile of prices, there exists a sequence of best replies that terminates in a non-malicious Nash equilibrium.*

In particular, this result implies the existence of non-malicious equilibria in trees, and together with the approximation theorem above, we conclude that the price of stability in trees (i.e., the approximation achieved by the best equilibrium in every tree) is at least $\Omega(\frac{1}{\log d})$. Of course, it would be interesting to strengthen this result and show that for any graph such dynamics terminate in a (non-malicious) equilibrium. Based on simulations we have executed, we conjecture that this is indeed the case.

▶ **Conjecture.** *In every graph any non-malicious best response dynamics starting from any price vector converges to an equilibrium in polynomial number of steps.*

This is the strongest version of our conjecture. One can also try to prove weaker versions of the conjecture by restricting the graph structure, order of plays, and relaxing the required time to convergence.

---

[3] Consider a clique of degree d with all edges with the same value of 1. Such a clique has a non-malicious equilibrium in which one seller prices at 0 while all others price at 1. The welfare in this non-malicious equilibrium is only linear in d, while the optimal welfare is quadratic in d, and thus there is a linear loss in d.

[4] We note that the theorem holds also for non-simple graphs that have parallel edges.

[5] Our result above also has a "price-of-anarchy" flavor, as we prove that all non-malicious equilibria (when exist) exhibit the approximation guarantee.

We also prove several impossibility results. We first show that the $\Omega(\frac{1}{\log d})$ approximation for trees is tight, by presenting trees in which the welfare of every non-malicious equilibrium is at most an $\Omega(\frac{1}{\log d})$ fraction of the monopolist revenue (and thus of the optimal welfare). However, this bound is proved using constructions for which there are other ("malicious") equilibria that achieve a constant approximation. We therefore strengthen this impossibility result and show instances where *all* equilibria, malicious or not, achieve bad results. In particular, we would like to find out whether the revenue of the best equilibrium is always a constant fraction of the monopolist's revenue. Our main lower bounds show that the answer is negative, even for trees.

▶ **Theorem.** *For graphs with maximum degree d:*

- *There are* graphs *for which the welfare in every Nash equilibrium is at most $O(\frac{1}{\sqrt{\log d}})$ fraction of the monopolist's revenue.*
- *There are* trees *for which the welfare in every Nash equilibrium is at most $O(\frac{1}{\log \log d})$ fraction of the monopolist's revenue.*

Note that the fact that the revenue in every equilibrium is low compared to the monopolist revenue, implies that it is also low compared to the maximum welfare. Moreover, our bounds are actually stronger, showing that not only the revenue in equilibrium is low, but also the welfare.

Related Literature

The analysis of price competition goes back to Cournot [8] and Bertrand [4]. The famous competition model of [8] describes producers who compete through quantities, but in the same work he also described a model of a duopoly selling perfect complements.

[21] consider buyers interested in perfect complements, where each bidder is interested in one specific bundle (single-minded bidders); They study mechanism design for a single seller and buyers with private information, where our focus is on competition between multiple sellers with complete information. Auctions for networks of buyers and sellers with private information were studied in [17].

There is a line of literature studying interactions of sellers and buyers over networks. Among them, [16] considered bargaining in networks, where agents can choose whom they want to negotiate with and the solution implies matching of buyers and sellers. In our model, neighbouring sellers also "negotiate" on dividing the value of the buyer on the edge between them, but unlike [16], a seller can serve several neighbouring buyers simultaneously (with the constraint of non-discriminatory pricing). Sub-game perfect equilibria in bargaining games were studied in [7]. [13] studied general equilibrium models on networks with linear utilities of buyers.

[2] considered a similar model of sellers that compete in prices, and a trade network that is modeled as graph, with one main difference from this paper: items are substitutes for the buyers. Namely, a buyer is interested in buying *either* from one seller or from the other seller on the edge, while here we consider buyers that are interested in *both*. This difference implies significant differences in results. For example, there, unlike our paper, pure Nash equilibria hardly ever exist. For trees, equilibrium utilities of the sellers are uniquely defined, while here they are not.[6]

[6] Our framework can be also viewed as a variant of graphical games [15]. There are several known algorithms for computing equilibria in graphical games (see survey [14]). Another related family of

2 Model

We study trade networks which are modeled as weighted undirected graphs, where sellers are represented by nodes and buyers by weighted edges. The set of edges is denoted by E, let N be the set of nodes and let $i, j \in N$ denote generic sellers. Sellers have no costs and can supply any number of items. The sellers are the players in the game, interested in maximizing revenue. Each buyer is single minded and is interested in the bundle of items sold by both sellers that lie on that edge. The weight $v_{i,j}$ on the edge represents the value of this buyer for buying the bundle $\{i, j\}$. In particular, buyers gain zero value from buying each item alone. A seller i posts a single price $p_i \geq 0$ that will be available to all incident edges (cannot price discriminate between buyers). A buyer of bundle $\{i, j\}$ buys if and only if $v_{i,j} \geq p_i + p_j$.

For a given price vector p we denote the set of sold edges by $S(p)$. The edge (i, j) is *tight* if $p_i + p_j = v_{i,j}$ (the sum of prices of the sellers on the edge equals the value of the edge). We say that *an edge* (i, j) *has slack* if $p_i + p_j < v_{i,j}$. For a given seller i and an edge (i, j), *the slack of seller* i *on that edge* is $v_{i,j} - p_j$ (that is, it equals to the difference between the value of the edge and the price of the *other* seller on that edge).

For a given price vector p, the *revenue* r_i of seller i with price p_i is $p_i \cdot |\{j|(i, j) \in S(p)\}|$, that is, p_i for every adjacent edge that is sold. The *total revenue* is $\sum_i r_i$ and the *welfare* is $\sum_{(i,j)\in S(p)} v_{i,j}$.

The sellers compete in a game in which they simultaneously post prices. These prices form a *(pure) Nash equilibrium (NE)* if each seller maximizes his revenue given the prices of all other sellers. In this paper we only consider pure Nash equilibria.

For a given network, the *revenue of the monopolist* is the supremum of the total revenue over all price vectors for the sellers. The *maximum welfare* is simply the sum of values of all edges (this can be obtained when every seller prices at 0). We study the ratio between the revenue in equilibrium and either the revenue of the monopolist or the maximum welfare (whichever bound is harder to prove): For our positive results we show that the equilibrium revenue is some fraction of the maximum welfare (which clearly implies approximation to the monopolist revenue), while for our negative results we prove inapproximability of the monopolist revenue (which implies the same result with respect to the maximum welfare).

Non Malicious Equilibria. We have already observed that the welfare of the worst equilibrium is arbitrarily low. Thus, we sometimes study an equilibrium refinement in which sellers with 0 utility price at 0. We say that a losing seller (seller with 0 utility) is *non malicious* if she prices at 0. A price vector for the sellers is called *non malicious* if every losing seller is non malicious. This, in particular, implies that every seller's revenue is at least as high as the price he sets. We use the concept of non-malicious NE when proving our main result: we first show that such equilibria always approximate the optimal welfare, and we then show that they always exist in trees. This implies a positive result on the price-of-stability in trees.

games is Polymatrix games (see [5, 9, 6]), where an action of a player is played in several simultaneous bimatrix games (the games in our paper have continuous action spaces); This class of games played an important role in showing hardness of equilibrium computation results. Our goal in this paper is to find equilibria with good economic properties, and compare their properties to the optimal non-strategic solution. Our paper shows that allowing losing sellers to price "maliciously" might be beneficial to society as a whole, this phenomenon was termed the "Windfall of Malice" by [1].

3 Special Families of Graphs: Paths and Cycles

In this section, we present positive results for special cases of graphs and algorithms that compute Nash equilibria with high revenue for those cases. These algorithms consist of a particular sequence of best responses of the sellers. We present a linear time algorithm for path graphs that finds an equilibrium that obtains revenue that is at least half of the maximum welfare. We use it to present a linear time algorithm for cycles that computes an equilibrium with revenue at least a quarter of the maximum welfare.

3.1 Path Graphs

We first consider the simplest non-trivial graphs: path graphs. Our algorithm (Algorithm 1) assumes that sellers are indexed from left to right, with the leftmost seller indexed by 1. The algorithm starts with all sellers pricing at ∞ and then changes the price of the leftmost seller to 0. Then, it goes over the sellers from left to right, letting each seller to choose a best response. If the seller cannot get positive utility, he prices at the value of the previous edge (not at 0 - and thus the equilibrium might be malicious). The way we break ties for sellers with 0 utility ensures that after seller i best responds, all previous sellers are still best responding (as seller i leaves no slack to the previous seller). Our algorithm may achieve terrible revenue when executed in a certain direction on the path (for example, when the weights of the edges are monotonically decreasing), but in such cases we show that running it from for the opposite direction will perform well. This is essentially since every time the value of the edge is larger than the previous edge, it is sold tightly. The algorithm thus picks the better equilibrium out of executing the above procedure starting from one end, or from the other.[7]

Algorithm 1 A $\frac{1}{2}$-Approximation for path graphs.

1. Initialize all prices to infinity, and the price of the first (leftmost) seller to 0.
2. Starting from the second seller, go over the sellers from left to right and let each seller best respond to the current prices. If every price of the current seller i gets him 0 utility, set his price to the value of edge $(i-1, i)$ (the value of the previous edge).
3. Mirror the path left to right (mapping node i to become node $n+1-i$) and repeat the above algorithm. Output one of these two price vectors with the higher total revenue.

This algorithm terminates in a NE with revenue of at least half the maximum welfare (proof appears in the full paper):

▶ **Proposition 1.** *For any path graph, Algorithm 1 terminates in a NE after a linear number of steps. The total revenue in this equilibrium is at least half of the maximum welfare.*

3.2 Cycles

Our next result shows that in any cycle graph there is always a NE with high revenue. Moreover, we show how to compute such a NE in linear time; The algorithm runs the above

[7] We illustrate the algorithm for the path graph of Figure 1: The algorithm chooses a price of 0 for the copper seller, then a price of 1 for the zinc seller, a price of 5 for iron and a price of 1 for glass. This is a NE with revenue 7 (out of maximum welfare of 8). By symmetry, the revenue starting from the other end would be the same.

path-graph algorithm for one full round on all sellers, and then initiates a best-response dynamic that will end up in such an equilibrium.

Fix an arbitrary seller and mark him as seller 1. Going clockwise along the cycle starting from seller 1, number the other sellers 2 to n. We present the algorithm as Algorithm 2.

Algorithm 2 A $\frac{1}{4}$-Approximation for Cycles

1. Initialize all prices to be infinity, initialize the price of seller 1 to 0.
2. Going clockwise: Starting from the second seller till the first seller (inclusive), go over the sellers clockwise and let each seller best response to the current prices. If every price of the considered seller i gets him 0 utility, set his price to the value of edge $(i-1, i)$ (the value of the previous edge in the cycle, with edge $(n, 1)$ being the edge prior to 1).
3. Run a best response dynamics (by iteratively letting a seller that is not best responding to update his price) till it stops.
4. Mirror the cycle and repeat the above algorithm (i.e. the new order after seller 1 is $n, n-1, \ldots, 2, 1$). Output one of these two price vectors with the higher total revenue.

▶ **Theorem 2.** *For any cycle, Algorithm 2 terminates in a Nash equilibrium after linear number of steps. The total revenue in this equilibrium is at least one quarter of the maximum welfare.*

The details of the proof can be found in the full paper. We next present an informal description of the algorithm together with an outline of the proof. First, the algorithm essentially runs Algorithm 1 in one direction, starting with a price of 0 for seller 1 and going clockwise, letting each seller best respond (pricing at the value of the prior edge if his utility is 0) including the first seller as the last seller to change his price. We note that the first seller is now best responding (as the edge to the second seller had no slack, so he could not get any utility out of it). After that cycle completes, we argue that there is at most one seller (the second seller) that is not best responding. From this point on, the algorithm runs a best response dynamics till it stops. We argue that at each point, the only seller that might not be best responding is the next seller, and once he best responds, either the sellers are in a NE or the edge with the previous seller is tight. As we show, it follows that this dynamics will stop after a linear number of steps. We also show that the algorithm running either clockwise or counter-clockwise gets revenue that is at least one quarter of the maximum welfare: the path-graph algorithm gets at least half, and from then edges are only added (and sold tightly), except the edge following the last seller to update his price that might be dropped. We show that the revenue of the dropped edge is no larger than the remaining revenue, thus we lose at most half the revenue obtained by the path algorithm.

4 A Positive Result

We start by presenting our main positive result, showing that the revenue in some NE is at least some fraction of the maximum welfare (and thus also some fraction of the monopolist revenue). We prove this claim by showing that this is true for any non-malicious NE. For any network for which a non-malicious equilibrium exists (like in trees, see Theorem 9), this implies that at least the same fraction can be obtained in the highest revenue equilibrium.[8]

[8] This result actually holds for a more general graph model, where some buyers are interested in a single good, not a pair, and also for graphs that contain parallel edges (that is, there might be multiple buyers

The bound we present is in terms of the maximum degree and the arboricity of the graph. The *maximum degree* is the largest degree of any node: the maximal number of buyers that demand an item from the same seller. The *arboricity* of an undirected graph is the minimum number of forests into which its edges can be partitioned. Equivalently, it is the minimum number of spanning forests needed to cover all the edges of the graph.

To gain some intuition for the notion of arboricity, here are some simple examples. A forest (and a tree) has arboricity 1. A cycle has arboricity 2 as it clearly cannot be spanned by a single forest, but it can be partitioned into two trees, a spanning tree and the missing edge. A clique of size n has arboricity $\lceil n/2 \rceil$: the arboricity is at least $n/2$ as any tree has at most $n-1$ edges and the clique has $n(n-1)/2$ edges, and it is at most $\lceil n/2 \rceil$ as it can be covered by $\lfloor n/2 \rfloor$ Hamiltonian paths, plus a star for odd n.

▶ **Theorem 3.** *In every graph with maximum degree d and arboricity w and every non-malicious NE in it, the total revenue of all sellers is at least $\frac{1}{2(w+1+\ln d)}$ fraction of the maximum welfare.*

Proof. Fix a graph and a non malicious Nash equilibrium in it. We will say that a vertex u is *low for the edge* (v,u) if u's price in the equilibrium is at most half of the value of the edge (v,u). We will say that an edge is *all-high* if both of the vertices on it are not low for the edge (in this case clearly the edge does not buy). Let E^H denote the set of all-high edges. For a vertex v let E^v denote the set of edges (v,u) (edges adjacent to v) such that u is low for (v,u). Observe that since every edge is either all-high or low for some node, the set of all edges E is covered as follows: $E = E^H \cup (\cup_v E^v)$. Thus

$$\sum_{(v,u)\in E} v_{(v,u)} \leq \sum_{(v,u)\in E^H} v_{(v,u)} + \sum_v \sum_{(v,u)\in E^v} v_{(v,u)} \,.$$

Denote the total revenue by r. To complete the proof we bound each of the two terms separately, the first by $2w \cdot r$ and the second by $2(\ln d + 1) \cdot r$, together proving the theorem.

We start by bounding the edges that are not all-high. Let r_v denote the revenue of v. This claim is well known (e.g., [11]) and we present the proof for completeness in the full paper.[9]

▶ **Claim 4.** *For every node v it holds that $\sum_{(v,u)\in E^v} v_{(v,u)} \leq r_v \cdot 2(\ln d + 1)$*

By the claim we derive that

$$\sum_v \sum_{(v,u)\in E^v} v_{(v,u)} \leq \sum_v r_v \cdot 2(\ln d + 1) = 2(\ln d + 1) \cdot r \,.$$

We next bound the total value of the all-high edges. To take care of the total weight of the all-high edges we will use the fact that in a graph of arboricity w there exists a mapping from edges to vertices such that every edge is mapped to one of its two vertices and no vertex has more than w edges mapped to it (this is true as we can just root each tree and map every edge to its child node). Since the edge is all-high and the equilibrium is non malicious, the price – and thus also the revenue – of each of the two vertices on the edge is at least half the value of the edge. Summing, again, over all vertices, we get that the total weight of all

interested in each pair of goods).

[9] The claim essentially says that a single price can gain revenue of at least a logarithmic fraction of the total demand (in our case, it is the residual demand given the prices of the neighbours- which is at least half the value of each edge as these edges are not all-high.).

all-high edges is at most $2w$ the total revenue of all vertices. Stated formally, consider the mapping M from E^H to the set of nodes N that maps each edge to an adjacent node and never maps more than w edges to the same node. For every node v that incident at least one high edge, define $u^*(v)$ to be a vertex such that for all u such that $(v,u) \in E^H$ we have $v_{(v,u^*(v))} \geq v_{(v,u)}$. Then, it holds that

$$\sum_{(v,u)\in E^H} v_{(v,u)} \leq \sum_{v} \sum_{u|M(v,u)=v} v_{(v,u)} \leq \sum_{v|\exists u} \sum_{M(v,u)=v} w \cdot v_{(v,u^*(v))} \tag{1}$$

$$\leq w \cdot \sum_{v} 2 \cdot r_v \leq 2w \cdot r \tag{2}$$

◀

For any network for which a non-malicious equilibrium exists, the theorem ensures that some Nash equilibrium has high revenue. Thus, it can be viewed as a "Price of Stability" result for such networks. In particular, it bounds the price-of-stability for trees, as for any tree a non-malicious equilibrium exists by Theorem 9. It also ensures that every non-malicious equilibrium has high revenue, thus can be viewed as a "Price of Anarchy" result for non-malicious equilibria.

The theorem implies a bound on trees that is only logarithmic in the maximum degree.

▶ **Corollary 5.** *In any tree with maximum degree d and every non-malicious equilibrium in it, the total revenue of all sellers is $\Omega(1/\ln d)$ fraction of the maximum welfare.*

5 Impossibility Results

Theorem 3 gives a positive result, ensuring that the revenue in equilibrium is some fraction of the welfare. Yet, it might be possible that an improved bound can be shown. Specifically, we would like to answer the following question: Is the revenue of best equilibrium always a constant fraction of the revenue of the monopolist? Unfortunately, the answer is no, even for trees. We first present a lower bound for general graphs. This bound is slightly weaker than the logarithmic bound (in the maximum degree) of Theorem 3. We then present a lower bound for trees, showing that the best equilibrium revenue is not necessarily a constant fraction of the monopolist revenue. For missing proofs see the full paper.

5.1 General Graphs

We start with the lower bound for general graphs. For proving the theorem, we construct graphs with arboricity that is much smaller than their maximum degree.

▶ **Theorem 6.** *There exists a family of graphs with maximum degree d and arboricity w for which $w^2 = O(\ln d)$ and the revenue that a monopolist seller can get is factor $\Omega(w)$ larger than the revenue (and welfare) in any Nash equilibrium. It terms of d, the factor can be as large as $\Omega(\sqrt{\ln d})$ when $w^2 = \Theta(\ln d)$.*

The bound is proven using the following construction (see Figure 2). There is a clique of size $w+1$ and any node in the clique is connected to an "harmonic gadget": d edges with values $1, 1/2, 1/3, \ldots, 1/d$. The value of an edge connecting two nodes in the clique is 4. We first claim that for these parameters, a monopolist would price all the clique nodes at 0 and get full revenue from all the harmonic gadgets. In any NE, however, at most one seller prices below $1/w$, thus not much revenue is gained from the harmonic gadgets.

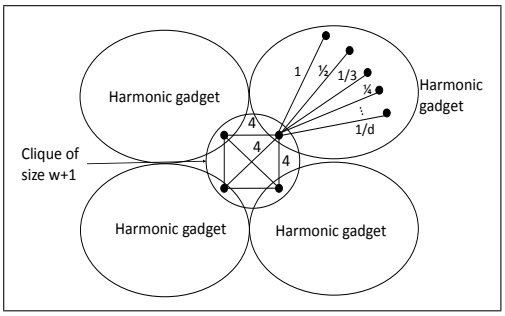


Figure 2 The construction of Theorem 6. A clique of $w+1$ sellers, each edge in it is of value 4. Each node in the clique is connected to an "harmonic gadget".

5.2 Trees

For trees, the above bound (when $w = 1$) is a constant. We next present a different construction that shows that for some trees, the revenue of a monopolist can be factor $\Omega(\ln \ln d)$ larger than the revenue of the best equilibrium. In particular, it show that a constant upper bound is impossible.

▶ **Theorem 7.** *There exists a family of trees with maximum degree d for which the revenue that a monopolist seller gets is factor $\Omega(\ln \ln d)$ larger than the revenue (and welfare) in any NE.*

For proving this impossibility result we construct the following graph, fixing an integer m (to be determined later). Consider a path with $2m+1$ edges. The first edge has value 5. For any $j = 1, 2, \ldots m$, given that edge $2j-1$ has value v, set the value of the edge $2j$ to be $2v+2$ and the value of edge $2j+1$ to be $2v+6$. Any node of even index is additionally connected to an "harmonic gadget" with $d-2$ spikes: $d-2$ edges with values $1, 1/2, 1/3, 1/4, \ldots, 1/(d-2)$.

We first argue that in any Nash equilibrium there is at most a single seller of even index on the path with price that is at most 1. We then argue that a monopolist seller can get revenue of at least $m \ln(d-2)$ by pricing every seller on the path at 0 and gaining all the revenue from the m harmonic gadgets. We conclude that the ratio of the monopolist revenue to equilibrium revenue (and welfare) is at least $\frac{\max\{m \ln(d-2), 2^m\}}{3+\ln d+2^{m+4}}$. For $m = \Theta(\ln \ln d)$ this ratio tends to $\Omega(m)$ as we aimed to prove. Full details appear in the full paper.

For trees, there is still a gap between this $\Omega(\ln \ln d)$ bound and the upper bound of $O(\ln d)$ for non-malicious NE as implied by Corollary 5. Closing this gap is left as an open problem.

5.3 Non malicious Nash equilibrium

We have observed that the worst Nash equilibrium might have zero revenue, even for a single edge. In this section, we consider non-malicious NE and present two simple examples that show that the upper bound of Theorem 3 is tight in both parameters for the non-malicious NE with the worst revenue ("Price of Anarchy").

▶ **Proposition 8.** *The following holds:*

- *For any $w \geq 2$ there exists a graph (symmetric clique) with $2w$ nodes and arboricity w for which the revenue in some non-malicious equilibrium is smaller than the monopolist revenue by a factor of at least w. Moreover, for that graph, every best-response dynamics starting at zero prices converges to such an equilibrium.*

- *For any $d \geq 2$ there exists a tree (star) with maximum degree d for which the revenue in some non-malicious NE is smaller than the monopolist revenue by a factor of at least* $\ln d$.

The two claims directly follow from the following simple examples. Consider first the clique graph of $n = 2w$ nodes with all weights being 1. Note that such a clique has arboricity w. Here is a non-malicious Nash equilibrium: one of the sellers prices at 0 and all others price at 1. The total revenue (and total welfare in this NE) is exactly $(n-1)$. Note that there exists a fully-efficient fully-revenue-extracting non-malicious equilibrium where each seller prices at $1/2$. In this NE the revenue (and welfare) is exactly $n(n-1)/2$. Thus, the ratio between the best non-malicious equilibrium revenue and the worst non-malicious equilibrium revenue is $n/2 = w$. Finally, observe that in any best response dynamics starting at 0, sellers keep increasing prices from 0 to 1, except when there is only one seller pricing at 0, and the others at 1. Thus, any best response dynamics starting at 0 prices will end at a non-malicious NE in which one seller prices at 0 and all others price at 1.

Consider now a star with d spikes, with edge i of value $1/i$. There is a non-malicious NE with revenue of 1: the center prices at 1 and all other price at 0. Revenue (and welfare) of $\sum_{i=1}^{d} 1/i \geq \ln d$ can be achieved by pricing the center at 0 and any other seller at the price of its edge. The second claim follows.

6 Best Reply Dynamics

Our main positive result (Theorem 3) ensures that for any graph for which a non-malicious equilibrium exists, there is an equilibrium with high revenue. Thus, one naturally wonders if all graphs admit a non-malicious equilibrium. If so, can one find such an equilibrium in polynomial time? Is there a natural dynamics that ends in such an equilibrium?

A natural procedure for converging to an equilibrium is repeated best-reply dynamics, and one might hope that such dynamics will indeed always converge to a non-malicious equilibrium in polynomial time. Such a dynamics starts with arbitrary prices, and at each step a seller that is not best replying is chosen, and he updates his price to a best reply. If the goal is finding a non malicious equilibrium, then one needs to consider a seller with 0 utility that is not pricing at 0 as a seller that is not best replying. We conjecture that for any graph, such a process terminates in an equilibrium (which is clearly non malicious). A stronger conjecture is that such an equilibrium will be reached in polynomial time.[10]

In this section, we prove the existence of non-malicious equilibria in *trees* by presenting some best-reply dynamic that converges to equilibrium in a finite number of iterations.[11]

▶ **Theorem 9.** *In every tree, for every initial profile of prices, there exists a sequence of player best replies that terminates in a non-malicious Nash equilibrium.*

The full proof appears in the full paper. We now present a sketch of the proof, and recursively define the sequence of best responses. We pick a leaf u of the tree that is connected to the rest of the tree via vertex v. Let x_0 be the initial price of u and y_0 be the initial price of v. Our best reply dynamics will proceed by repeatedly, for $i = 1 \ldots$, let x_i be u's reply to y_{i-1}, and then recursively use a best reply sequence that updates the rest of the tree, assuming that u's price is set to x_i. Note that this recursive best-reply sequence starts with

[10] We remark that we have simulated best response dynamics on general graphs with arbitrary order, and they always terminated in equilibrium very fast.

[11] Our proof only ensures that the dynamics will stop, but does not ensure terminating in polynomial time.

v updating his price (since all the other vertices are already best-replying from the previous recursive call, but then other vertices may update their price and v may update again, and so on). The v's price at the end of the recursive call is called y_i. The recursive call on the sub-tree terminates due to an inductive use of the theorem (i.e., the theorem is proved by induction on the number of vertices in the tree). To ensure that the induction hypothesis applies to the recursive call which is applied not just to a subtree, but rather to a subtree to which an extra leaf u with a *fixed* value x_i is attached, we prove the theorem (inductively) also for trees in which each vertex may have an arbitrary number of leaves with a fixed value attached to them.

References

1. Moshe Babaioff, Robert Kleinberg, and Christos H. Papadimitriou. Congestion games with malicious players. *Games and Economic Behavior*, 67(1):22–35, 2009.
2. Moshe Babaioff, Brendan Lucier, and Noam Nisan. Bertrand networks. In *ACM Conference on Electronic Commerce (ACM-EC)*, 2013.
3. Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, EC'06, pages 29–35, 2006.
4. Joseph Louis François Bertrand. theorie mathematique de la richesse sociale. *Journal de Savants*, 67:499–508, 1883.
5. Yang Cai and Constantinos Daskalakis. On minmax theorems for multiplayer games. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'11, pages 217–234, 2011.
6. Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *J. ACM*, 56(3):14:1–14:57, 2009.
7. Margarida Corominas-Bosch. Bargaining in a network of buyers and sellers. *Journal of Economic Theory*, 115(1):35–77, 2004.
8. Antoine Augustin Cournot. *Recherches sur les principes mathematiques de la theori des Richesses*. Trans. N.T. Bacon, New York: Macmillan, 1929, 1838.
9. Argyrios Deligkas, John Fearnley, Rahul Savani, and Paul Spirakis. Computing approximate nash equilibria in polymatrix games. In *Web and Internet Economics: 10th International Conference, WINE 2014*, pages 58–71, 2014.
10. C. Ellet. *An essay on the laws of trade in reference to the works of internal improvement in the United States*. Reprints of economic classics. A.M. Kelley, 1966.
11. Andrew V. Goldberg, Jason D. Hartline, Anna R. Karlin, Michael Saks, and Andrew Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242–269, 2006.
12. Venkatesan Guruswami, Jason D. Hartline, Anna R. Karlin, David Kempe, Claire Kenyon, and Frank McSherry. On profit-maximizing envy-free pricing. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'05, pages 1164–1173, 2005.
13. Sham M. Kakade, Michael Kearns, Luis E. Ortiz, Robin Pemantle, and Siddharth Suri. Economic properties of social networks. In *Advances in Neural Information Processing Systems 17*, pages 633–640. MIT Press, 2005.
14. Michael Kearns. *In Noam Nisan, Tim Roughgarden, Eva Tardos and Vijay Vazirani (Editors), Algorithmic Game Theory. Chapter 7. Graphical Games*. Cambridge University Press, 2007.
15. Michael J. Kearns, Michael L. Littman, and Satinder P. Singh. Graphical models for game theory. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI'01, pages 253–260, 2001.

16 Jon Kleinberg and Éva Tardos. Balanced outcomes in social exchange networks. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC'08, pages 295–304, 2008.

17 E. Kranton and Deborah F. Minehart. A theory of buyer-seller networks. *American Economic Review*, 91:485–508, 2001.

18 Robert Krauthgamer, Aranyak Mehta, and Atri Rudra. Pricing commodities. *Theor. Comput. Sci.*, 412(7):602–613, 2011.

19 Euiwoong Lee. Hardness of graph pricing through generalized max-dicut. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, STOC'15, pages 391–399, 2015.

20 E.w Lee, D. Buchfuhrer, L. Andrew, A. Tang, and S. Low. Progress on pricing with peering. In *Proceedings of the 45th Annual Allerton Conference on Computing, Communications and Control*, Allerton'07, pages 286–291, 2007.

21 Daniel Lehmann, Liadan Ita O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. In *JACM 49(5)*, pages 577–602, Sept. 2002.

22 C. St.J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal of the London Mathematical Society*, s1-36(1):445–450, 1961.

House Markets with Matroid and Knapsack Constraints*

Piotr Krysta[1] and Jinshan Zhang[2]

1 **Department of Computer Science, University of Liverpool, Liverpool, UK**
pkrysta@liverpool.ac.uk
2 **Department of Computer Science, University of Liverpool, Liverpool, UK**
jinshan.zhang@liverpool.ac.uk

Abstract

Classical online bipartite matching problem and its generalizations are central algorithmic optimization problems. The second related line of research is in the area of algorithmic mechanism design, referring to the broad class of house allocation or assignment problems. We introduce a single framework that unifies and generalizes these two streams of models. Our generalizations allow for arbitrary matroid constraints or knapsack constraints at every object in the allocation problem. We design and analyze approximation algorithms and truthful mechanisms for this framework. Our algorithms have best possible approximation guarantees for most of the special instantiations of this framework, and are strong generalizations of the previous known results.

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

Keywords and phrases Algorithmic mechanism design, Approximation algorithms, Matching under preferences, Matroid and knapsack constraints

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.141

1 Introduction

Classic online bipartite matching is one of the central algorithmic optimization problems. Since the seminal paper of Karp, Vazirani and Vazirani [21], there have been new developments and generalizations of this model [4, 14, 20]. A related line of research is within algorithmic mechanism design for a broad class of house allocation/assignment problems [1, 7, 18, 19, 27]. House Allocation (HA) problem is a model of assigning indivisible objects to agents, where each agent with a preference order over a subset of objects requires at most one object and payments are not allowed. HA and its generalizations have wide real life applications such as Campus Housing Allocation [12], Student-project Allocation [2], Machine-job Assignment [9]. Our goal is to unify these two streams of models into a single algorithmic framework.

In classic HA, agents' preferences over objects are strict. In many situations, it is more suitable to allow the agent to express indifferences or ties among objects [25]. For example, in Campus Housing Allocation, each student may provide some features (e.g., separate bathroom) of dormitories, and he is indifferent between dormitories with same features. Besides, other optimality criteria for the allocation are also desired, like Pareto optimality. An allocation μ is Pareto optimal if there is no other allocation μ' such that no agent is worse off in μ' and at least one agent is strictly better in μ' compared to μ. Moreover, the agents

* This work was supported by EPSRC under grant EP/K01000X/1.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 141; pp. 141:1–141:14

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

are strategic (selfish) players who may not reveal their preference orders truthfully unless they can not get better by misreports. Thus, designing truthful mechanisms is a central issue in mechanism design for HA. A mechanism is truthful (in dominant strategies) if no agent can get better off by misreporting his preference, independent from other agents' reports.

In real world applications, not all the agents can be guaranteed to be matched since the set of their acceptable objects is a subset of all the objects and there may not be enough objects. Thus, designing algorithms to match as many agents as possible is also a much desired objective. This can be quantified by the approximation ratio between the sizes of the optimal matching and the matching obtained by the algorithm, preserving truthfulness and Pareto optimality. In addition, different agents may have different weights or priorities, and in this case, the ratio is between the weights of the optimal weighted matching and the output matching. Krysta et al. [22] initiated the study of designing truthful and Pareto optimal mechanisms for HA with good approximation ratios. They present a tight deterministic 2-approximation, truthful and Pareto optimal mechanism, and a randomized $\frac{e}{e-1}$-approximate, universally truthful and Pareto optimal mechanism. In their setting, each object can be allocated to at most one agent. Can we generalize their results to more general settings?

A simple generalization, called generalized HA, allows each object a to be allocated to at most a given number, $b_a \in \mathbb{N}$, of agents. We observe that all the results in [22] can easily be applied to this generalization. Let us create b_a copies of each object a, and replace a in each agent's preference order by these b_a copies in the same indifference class as a. By this modification, called modified HA, each copy of each object is allowed to be allocated to at most one agent. Each feasible allocation of the generalized HA can be modified to a feasible solution to the modified HA, and vice versa. The mechanism in [22], which is Pareto optimal and truthful for the modified HA, is also Pareto optimal and truthful for the generalized HA.

We study here as our framework, two further natural generalizations of the HA problem: Matroid House Allocation Problem (MHA) and Knapsack House Allocation Problem (KHA). Consider the following more realistic application of HA to the Campus Housing Allocation problem. A university has many dormitories, and each student submits a preference order over dormitories to the university. Besides this, some students may have further requirements about their dormitories which should be fulfilled, e.g., one may require that the room should have an independent bathroom or be far from the kitchen. Thus, for each dormitory a, viewed as an object, and each student i, the rooms in a satisfying the requirements of i are a subset of all the rooms in a. Hence, all the feasible sets of students allocated to a form a transversal matroid (see definition in Section 2) of dormitory a. Our first generalization of HA is based on this application, where each object in HA is associated with a (possibly different) matroid and can be allocated to any number of agents in the ground set (which is the set of agents) of the matroid. The set of agents allocated to each object forms an independent set of its matroid. We call this generalization the Matroid House Allocation Problem (MHA). We can show that MHA includes as special cases all the online bipartite matching models [4, 14, 20, 21] ([20] if fixed vertices are used at most once) mentioned earlier.

The second generalization is as follows: each agent (or job) i is associated with a capacity c_{ia} for each object (machine) a, and each object a is associated with a capacity C_a. As before, each object can be allocated to multiple agents, but the total capacity of the agents allocated to a does not exceed its capacity (i.e., $\sum_i c_{ia} \leq C_a$). We call this generalization the Knapsack House Allocation Problem (KHA). An interesting application of KHA is Job Scheduling Markets [9]. Our main results are the following:

1. We present a tight, deterministic, truthful and Pareto optimal mechanism (TMHA) for MHA with the approximation ratio of 2, where agents have weights and ties.

2. Based on TMHA, a randomized, universally truthful and Pareto optimal mechanism (RTMHA) is presented. We prove that RTMHA is an $\frac{e}{e-1}$-approximation mechanism for MHA where agents have weights and ties. Note, truthful $O(1)$-approximation is impossible if each agent has different weights over objects [17, 3]. 1. and 2. are strong generalizations of the results in [22], and are obtained with completely new techniques.
3. A universally truthful, Pareto optimal mechanism for KHA with a $(4, 2)$-approximation (4-approximation ratio, violating each knapsack constraint by a factor 2) for parallel machines (objects) ($c_{ia} = c_i$, for any object a) when weights and ties are allowed ($(4, 1)$-approximation without ties). For general KHA with weights and ties, we observe that [11] implies a universally truthful $O(1)$-approximate mechanism (but not Pareto optimal).
4. Our mechanism RTMHA applied to the online weighted matroid bipartite matching problem and the online matroid job recruitment problem, implies tight $\frac{e}{e-1}$-approximations.

Technical Contributions. We present a generic approach to design deterministic truthful and Pareto optimal mechanisms for MHA and KHA when agents' preferences include ties, by reducing mechanism design to algorithm design of the underlying graph (i.e., calling the algorithm of finding maximum cardinality matching on an auxiliary graph as a black box). The mechanism RTMHA for MHA shares in spirit the same idea as Random SDMT mechanism in [22]. The main difficulty lies in the analysis of its $\frac{e}{e-1}$-approximation. Our new setting is much more general than [22], thus a completely different analysis is needed. We split the analysis of RTMHA into three parts, and its starting point is based on the charging map method (extended to matroid constraints), widely used for matching problems [4, 10].

First, for unweighted agents without ties, we show that the symmetric difference between a matching μ^τ output by TMHA under an order τ of all agents and the matching $\mu^{\tau_{-i}}$ output by TMHA under order τ_{-i} with agent i absent, is an alternating path starting from i. This characterization implies that, for each object $a \in A$, the agents $S_a(\tau)$ matched to a under μ^τ and agents $S_a(\tau_{-i})$ matched to a under $\mu^{\tau_{-i}}$ satisfy a nice exchange property: if some agent can be added to $S_a(\tau)$ then he can also be added to $S_a(\tau_{-i})$ obeying the matroid constraint.

When preferences have ties, we say that two matchings are equivalent if the matched agents are the same and the matched objects of the same agent are in the same indifference class of that agent. The second part of our analysis, for unweighted agents with ties, shows a similar characterization between the equivalence class $\mathrm{CL}(\mu^\tau)$ (all matchings equivalent to matching μ^τ) and the class $\mathrm{CL}(\mu^{\tau_{-i}})$. That is, there exists an injective map from $\mathrm{CL}(\mu^\tau)$ to $\mathrm{CL}(\mu^{\tau_{-i}})$ such that the symmetric difference of each matching in $\mathrm{CL}(\mu^\tau)$ with its image in $\mathrm{CL}(\mu^{\tau_{-i}})$ is an alternating path. This lets us reduce the problem for agents with ties to that with agents without ties. We believe this technique will find further applications to analyze mechanisms for agents with ties in ordinal settings, which may simplify the proofs. For example, with this technique, we greatly simplify the proof for weighted agents with ties and objects with uniform matroid constraints in [22], where they use a different technique based on trading graphs. Based on this characterization we prove the important Injectivity Lemma; it can be used to show the $\frac{e}{e-1}$-approximation for unweighted agents with ties directly.

Thirdly, for the most general setting of weighted agents with ties, we carefully utilize the previous injectivity lemma, proving a strengthened version of such a lemma (see Lemma 13). Based on this lemma, we show that there exists an injective map from marginal 'bad' events to 'good' events, which suffices for the analysis of the final approximation ratio.

Related Work. Random Serial Dictator (RSD) [1] and Probabilistic Serial (PS) [7] are two paradigmatic randomized mechanisms for HA. Krysta et al. [22] generalize RSD mechanism

to HA with ties and obtain a tight 2-approximate deterministic truthful and Pareto optimal mechanism for weighted agents and an $\frac{e}{e-1}$-approximate universally truthful, Pareto optimal mechanism. They show that no universally truthful, Pareto optimal mechanism can have approximation better than $\frac{18}{13}$, and if the mechanism is additionally non-bossy, the lower bound on the approximation ratio is $\frac{e}{e-1}$. Our random mechanism is non-bossy for agents with strict preferences and thus has best possible approximation ratio in this sense. Bogomolnaia and Moulin [8] show the same ratio $\frac{e}{e-1}$ also for PS. Recently, [17, 3] establish an $O(\sqrt{n})$ bound of RSD (w.r.t. the optimal social welfare) when agents have cardinal values (with arbitrary weights on the objects) over all the objects and show that no truthful in expectation mechanism can have approximation better than $\Omega(\sqrt{n})$. We obtain constant approximation ratios when the agents are weighted. This lower bound $\Omega(\sqrt{n})$ implies that we cannot obtain similar results when agents have different weights over objects. Tight deterministic truthful mechanisms for weighted matching markets were proposed by Dughmi and Ghosh [15].

(M)HA is related to online bipartite matching problem (OBM) as follows. If each agent in HA ranks his desired objects in the order that precisely follows the arrival order of objects in the OBM, the two problems are equivalent, as emphasized in [6, 22]. Karp et al. [21] initialize the study of OBM and provide a RANKING algorithm with a tight approximation ratio $\frac{e}{e-1}$. Aggarwal et al. [4] are the first to study the weighted version of OBM (WOBM) when the fixed vertices have weights (or priorities). They use the charging map approach and prove that the ratio $\frac{e}{e-1}$ also holds. Recently, the analysis of WOBM has been unified into the primal-dual framework by Devanur et al. [14]. There are further generalizations of the online bipartite matching model, e.g., [10, 23, 24]. Matroid HA has an online interpretation, which is loosely related to the Matroid Secretary Problem introduced by Babaioff et al. [5].

Organization. Section 2 contains preliminaries, and Section 3 presents our deterministic mechanism for MHA. We develop a randomized mechanism for MHA in Section 4. Its analysis is divided into three subsections: 4.1 for unweighted agents without ties, 4.2 for unweighted agents with ties, 4.3 for weighted agents with ties. Applications of our mechanisms for MHA and our results for KHA will be published in the the full version of the paper.

2 Preliminaries

Let $N = \{1, 2, \cdots, n_1\}$ be a set of n_1 *agents* and A a set of n_2 *objects*; $n = n_1 + n_2$. Let $[i] = \{1, 2, \cdots, i\}$. Each agent $i \in N$ finds a subset of objects acceptable and has a preference ordering, not necessarily strict, over these objects.

We write $a_t \succ_i a_s$ if agent i *strictly prefers* object a_t to object a_s, and $a_t \simeq_i a_s$ if i is *indifferent between* a_t and a_s. We use $a_t \succeq_i a_s$ to denote that i *weakly prefers* a_t to a_s, i.e., either $a_t \succ_i a_s$ or $a_t \simeq_i a_s$. In some cases a weight w_i is associated with each agent i, representing i's priority or importance; let $W = (w_1, w_2, \ldots, w_{n_1})$. If we are in an unweighted setting then $w_i = 1$ for each i.

Each agent's acceptable objects are divided into *indifference classes*: he is indifferent between the objects in the same class and has a strict preference ordering over these classes. For each agent i, let C_k^i, $1 \le k \le n_2$, be the kth indifference class (*tie*), of i; assume that $C_q^i = \emptyset$, $\forall q, l \le q \le n_2$, if $C_l^i = \emptyset$ for some $l \in [n_2]$. Let $L(i) = (C_1^i \succ_i C_2^i \succ_i \cdots \succ_i C_{n_2}^i)$ be the preference list of i. We write $a \in L(i)$ if a appears in $L(i)$ (i finds a acceptable). Let $L = (L(1), L(2), \cdots, L(n_1))$ and $L(-i) = (L(1), \ldots, L(i-1), L(i+1), \ldots, L(n_1))$.

A matroid is a pair $\mathcal{M} = (X, \mathcal{I})$, where X is a ground set and $\mathcal{I} \subseteq 2^X$ (each $Y \in \mathcal{I}$ is called independent set) satisfying properties: (1) [**non-emptiness**] $\emptyset \in \mathcal{I}$; (2) [**heredity**]

If $Y \in \mathcal{I}$ and $Z \subseteq Y$, then $Z \in \mathcal{I}$; (3) [**exchange**] If $Z, Y \in \mathcal{I}$ and $|Z| > |Y|$, then there is $z \in Z \backslash Y$ such that $Y \cup \{z\} \in \mathcal{I}$. A base of $\mathcal{M}$ is an independent set with maximum size. By (3), any maximal independent set is a base. For any $S \subseteq X$, let $r_{\mathcal{M}}(S)$ be the maximum size of an independent set contained in S; $r_{\mathcal{M}}(X)$ is called the rank of $\mathcal{M}$. A matroid $\mathcal{M} = (X, \mathcal{I})$ is called an ℓ-uniform matroid if $Y \subseteq X$ is independent if and only if $|Y| \leq \ell$. A direct sum of uniform matroids (not necessarily with the same rank) is called a partition matroid. Given a set X and a family $\mathcal{F} = \{X_i\}_{i=1}^{s}$ of subsets of X, a set $T \subseteq X$ is called a transversal set of X if there is an injective map f: $T \to \mathcal{F}$ s.t. $x \in f(x)$, for any $x \in T$. The matroid $(X, \mathcal{I})$, with $\mathcal{I} = \{T \,|\, T \text{ is a transversal set of } X\}$, is called a transversal matroid.

In Matroid HA (MHA), we have a matroid $\mathcal{M}_a = (N, \mathcal{I}_a)$ on agent set N, for each object $a \in [n_2]$. A (feasible) *matching* (or *b-matching*) μ is an allocation (subset of $N \times A$) assigning at most one object to each agent i[1] such that, for each object a, the set of agents who are assigned a is an independent set of $\mathcal{M}_a$ (multiple agents may be allocated the same object). If $(i, a) \in \mu$, agent i and object a are *matched together*. If $(i, a) \in \mu$ for some a, we say that i is *matched*, and *unmatched* otherwise. The definitions of *matched* (*unmatched*) for an object are analogous. If $i \in N$ is matched, $\mu(i)$ denotes the object matched to i. If object a is matched, $\mu^{-1}(a)$ denotes the set of agents matched to a. Two matchings μ, μ' are *equivalent* (denoted $\mu \simeq \mu'$) if in these two matchings, the matched agents are the same and the matched objects for each agent are in the same indifference class of this agent, i.e., $\mu(i) \simeq_i \mu'(i)$, allowing $\emptyset \simeq_i \emptyset$, for any $i \in N$. Let $\mathrm{CL}(\mu)$ be the class of all matchings equivalent to μ. For two sets B and C, we sometimes use $B + C$ ($B - C$, resp.) to denote $B \cup C$ ($B \backslash C$, resp.), and sometimes write a set $\{x\}$ simply as x. Let Π denote the set of all permutations of agents.

In what follows, we will consider the undirected graph $G = (V, E)$ where $V = (N \cup A)$ and $E = \{(i, a), i \in N, a \in L(i)\}$. We also use μ to denote a matching (a feasible matching for MHA) in G. The *size* of a matching μ is equal to the number of agents matched under μ. In the presence of weights, the *weight* of a matching is equal to the sum of the weights of the matched agents. For any subset $E' \subseteq E$, define $E'_v = \{u \,|\, (u, v) \in E'\}$, for any $v \in V$. An instance of MHA is denoted by $I = (N, A, L, (\mathcal{M}_a)_a, W)$. We drop W and $(\mathcal{M}_a)_a$ and write $I = (N, A, L, \mathcal{M})$ if agents are unweighted. Let $\mathcal{I}^{MHA}$ denote the set of all possible instances of MHA. For two given matchings μ_1, μ_2, we will use $\mu_1 \oplus \mu_2$ to denote the symmetric difference with respect to their sets of edges. An *alternating path* in G, w.r.t. a matching μ_1, is a path that consists of edges that alternately belong to μ_1 and do not belong to μ_1.

A matching μ is *Pareto optimal* if there is no other matching under which some agent is strictly better off while none is worse off, w.r.t. their preferences. Formally, μ is *Pareto optimal* if there is no other matching μ' such that (i) $\mu'(i) \succeq_i \mu(i)$ for all $i \in N$, and (ii) $\mu'(i') \succ_{i'} \mu(i')$ for some $i' \in N$. Given an order $\sigma \in \Pi$, a matching μ is strictly lexicographically σ-better than a matching μ' if there exists a $k \in [n_1]$ such that $\mu(\sigma(i)) \simeq_{\sigma(i)} \mu'(\sigma(i))$, $i \in [k-1]$ and $\mu(\sigma(k)) \succ_{\sigma(k)} \mu'(\sigma(k))$. A matching μ is called a lexicographically σ-maximal matching if there does not exist a matching μ' such that μ' is strictly lexicographically σ-better than matching μ. Note, if μ is lexicographically σ-maximal, then μ must be Pareto optimal.

Let $\mathcal{M}^{MHA}$ denote the set of all possible matchings. A *deterministic mechanism* ϕ maps an instance of MHA to a matching, i.e., $\phi : \mathcal{I}^{MHA} \to \mathcal{M}^{MHA}$. Let $R : \mathcal{M}^{MHA} \to [0, 1]$ be a probability distribution over possible matchings (a *random matching*), i.e., $\sum_{\mu \in \mathcal{M}^{MHA}} R(\mu) = 1$. A *randomized mechanism* ϕ is a mapping from $\mathcal{I}^{MHA}$ to a distribution over possible matchings, i.e., $\phi : \mathcal{I}^{MHA} \to Rand(\mathcal{M}^{MHA})$, where $Rand(\mathcal{M}^{MHA})$ is the set of all random

[1] Note that if agents are allowed to receive more than one object, the model becomes a many-to-many matching and the preference spaces of agents are much more complicated, which is not studied here.

matchings. A deterministic (randomized, resp.) mechanism is Pareto optimal if it returns a Pareto optimal matching (a distribution over Pareto optimal matchings, resp.).

Agents' preferences are private knowledge and they may prefer not to reveal their preferences truthfully if it is not in their best interests, for a given mechanism. A *deterministic* mechanism is *truthful* (*in dominant strategies*) if agents always find it in their best interests to declare their preferences truthfully, no matter what other agents declare, i.e., for every i and every possible declared list $L'(i)$ for i, $\phi(L(i), L(-i)) \succeq_i \phi(L'(i), L(-i))$, $\forall L(i), L(-i)$. A randomized mechanism ϕ is *universally truthful* if it is a probability distribution over deterministic truthful mechanisms. Let $w(\phi(I))$ be the (expected) weight of the (random) matching output by ϕ on instance $I \in \mathcal{I}^{MHA}$, and $w(I)$ be the weight of a maximum weight Pareto optimal matching in I. The *approximation ratio* of ϕ is defined as $\max_{I \in \mathcal{I}^{MHA}} \frac{w(I)}{w(\phi(I))}$.

For any agent i, define a matroid on agent i as $\mathcal{M}_i = (A, \mathcal{I}_i)$, with $\mathcal{I}_i = \{S \,|\, S \subseteq A \text{ and } |S| \leq 1\}$. We define two useful matroids on edge set E. Let $\mathcal{M} = (E, \mathcal{I})$, where, for any $E' \subseteq E$, $E' \in \mathcal{I}$ if and only if $E'_a \in \mathcal{I}_a$, for any $a \in A$. Let $\mathcal{M}' = (E, \mathcal{I}')$, where, for any $E' \subseteq E$, $E' \in \mathcal{I}'$ if and only if $E'_i \in \mathcal{I}_i$, for any $i \in N$. Note, $\mathcal{M}'$ is a partition matroid. Also, an allocation $E' \subseteq E$ is feasible for MHA if and only if $E' \in \mathcal{I} \cap \mathcal{I}'$, meaning that E' is an independent set in both $\mathcal{M}'$ and $\mathcal{M}$. Hence, our objective is to design (universally) truthful, Pareto optimal mechanisms to find an edge set that is independent in both $\mathcal{M}'$ and $\mathcal{M}$, such that its size (or weight for weighted agents) is maximized. Edmonds [16] provides an algorithm to compute a maximum independent set that is independent for two matroids.

▶ **Proposition 1** ([16, 13]). *Given two matroids $\mathcal{M}^1 = (X, \mathcal{I}^1)$ and $\mathcal{M}^2 = (X, \mathcal{I}^2)$, there is a matroid intersection algorithm $\mathcal{A}^{MI}$ for computing a maximum cardinality common independent set $S \in \mathcal{I}^1 \cap \mathcal{I}^2$, terminating in polynomial time $\Gamma(|X|) \leq O(|S|^{\frac{3}{2}}|X|Q^{test})$, where Q^{test} is the time needed to test if a given set is independent.*

We assume for simplicity that the test time $Q^{test} = O(1)$, and we omit it in the rest of the paper. We also need a fact from exchange theory between two bases of a matroid.

▶ **Proposition 2** (Corollary 39.12a [26]). *For a matroid $\mathcal{M} = (X, \mathcal{I})$ and $B_1, B_2 \in \mathcal{I}$ with $|B_1| = |B_2|$, there exists a bijection f: $B_1 - B_2 \to B_2 - B_1$ s.t. $\forall x \in B_1 - B_2$, $B_1 - x + f(x) \in \mathcal{I}$.*

3 Deterministic Mechanism for MHA

We introduce TMHA, a truthful and Pareto optimal mechanism, that generalizes SDMT-1 [22] to the case where sets allocated to the objects are independent sets of matroids. Let $I = (N, A, L, \mathcal{M})$ be an instance of MHA, and $\sigma \in \Pi$ (w.l.o.g., $\sigma(i) = i$ for $i \in N$). TMHA, given in Algorithm 1, proceeds in n_1 phases, each phase corresponds to one iteration of the for loop. Notice, at any stage of TMHA, $(i, a) \in E$ if and only if either agent i is matched in μ and $a \simeq_i \mu(i)$ or TMHA is at phase i, examining the indifference class that contains a.

▶ **Observation 3.** *At the end of phase i of TMHA, if agent i is assigned no object then he will be assigned no object when TMHA terminates. If i is provisionally assigned an object a, then he will be allocated an object in the same indifference class as a in the final matching.*

By Observation 3, different tie breaking rules lead to equivalent produced matchings.

▶ **Theorem 4.** *Given an order $\sigma \in \Pi$ of agents, the matching generated by TMHA is a lexicographically σ-maximal matching, thus, also Pareto optimal.*

TMHA is truthful, no matter which matching is selected in each phase of the mechanism:

Algorithm 1: Truthful Mechanism for MHA (TMHA)

Input: Agents N; Objects A; Preference list profile L; Matroids $(\mathcal{M}_a)_a$; Order σ
Output: Matching μ

Let $G = (N \cup A, E)$, $E \leftarrow \emptyset$, $\mu \leftarrow \emptyset$.
for *each agent* $i \in N$ *in the order of* σ **do**
  Let $\ell \leftarrow 1$
  Step (*): **if** $C_\ell^i \neq \emptyset$ **then**
    $E \leftarrow E \cup \{(i,a) : a \in C_\ell^i\}$; // all new edges are non-matching edges
    Run $\mathcal{A}^{MI}$ on G and obtain a maximum cardinality matching μ'
    if $|\mu'| = |\mu| + 1$ **then**
      modify μ to μ'; // i must be provisionally allocated some $a \in C_\ell^i$ and (i,a) is now a matching edge
    else
      $E \leftarrow E \setminus \{(i,a) : a \in C_\ell^i\}$; $\ell \leftarrow \ell + 1$; Go to Step (*)
Return μ; //each matched agent is allocated his matched object

▶ **Theorem 5.** *The mechanism TMHA is truthful.*

We now show a bound on the time complexity of TMHA. Let γ denote the size of the largest indifference class for a given instance I.

▶ **Theorem 6.** *TMHA terminates in time* $O(n_1^3\gamma)$.

Any Pareto optimal matching is at least half the size of a maximum size such matching [9]. Thus, TMHA achieves approximation ratio 2 with respect to the maximum cardinality matching. We show that, when agents are assigned arbitrary weights, TMHA is 2-approximate (w.r.t. maximum weight Pareto optimal matching) if the order σ is by non-increasing agents' weights, breaking ties arbitrarily. The bound is tight since no deterministic truthful mechanism can achieve an approximation ratio better than 2 even if $r_{\mathcal{M}_a} = 1$, for any $a \in A$, see [22].

▶ **Theorem 7.** *TMHA achieves a 2-approximation w.r.t. the size of a maximum weight Pareto optimal matching, if agents are ordered in* σ *by non-increasing order of their weights.*

4 Randomized Mechanism for MHA

We now present a universally truthful, Pareto optimal mechanism for Matroid House Allocation Problem (see Algorithm 2, where $g(y) = e^{y-1}$). When the matroid for each object is a uniform matroid with rank one, Algorithm 2 reduces to SDMT-1 from [22]. The analysis of RTMHA will be gradually developed in the following three subsections for various settings. We will start with the simplest setting of unweighted agents without ties (Subsection 4.1), then proceed to the setting of unweighted agents with ties (Subsection 4.2), and, finally, weighted agents with ties (Subsection 4.3). The next subsection builds on the previous one.

If the weights are agents' private data with no over-bidding assumption, Algorithm 2 is universally truthful (w.r.t. preferences and weights) and Pareto optimal as well [22].

4.1 Unweighted Agents without Ties

We will show now that RTMHA achieves $\frac{e}{e-1}$-approximation for unweighted agents without ties. Note that the order of RTMHA for unweighted agents is just the uniform random order

Algorithm 2: Random Truthful Mechanism for MHA (RTMHA)

Input: Agents N; Objects A; Preference list profile L; Matroids $(\mathcal{M}_a)_a$; Weights W
Output: Matching

```
for each agent i ∈ N do
    Pick Y_i ∈ [0,1] uniformly at random;
Sort agents in decreasing order of w_i(1 − g(Y_i)) (break ties in favor of smaller index);
Run TMHA according to above order;
Return the matching;
```

of agents. Our technique is based on a charging map method (extended by us to matroids), which is widely used in analyzing the approximation ratio for matching problems [4, 10].

We assume here that agents' preferences are strict: $|C_j^i| \leq 1$, for $i \in [n_1]$, $j \in [n_2]$. Recall, each object a is associated with a matroid $\mathcal{M}_a = (N, \mathcal{I}_a)$. We present a characterization of $\mu^\tau \oplus \mu^{\tau_{-i}}$, where μ^τ and $\mu^{\tau_{-i}}$ are the matchings obtained by TMHA under the order τ of agents and τ_{-i} (τ with i absent), respectively. We first prove a useful lemma.

Lemma 8 shows exchange properties between two independent sets of a matroid and their switching sets. Let $T_0 = \{i'_1, i'_2, \cdots, i'_k\}$ and $T_k = \{i_1, i_2, \cdots, i_k\}$ be two sets of size k with different elements (orders of elements in T_0 and T_k are fixed). We define the *switching sets* T_ℓ, $\ell \in [k-1]$ between T_0 and T_k, as $T_\ell = \{i_1, i_2, \cdots, i_\ell, i'_{\ell+1}, \cdots, i'_k\}$, $\ell \in [k-1]$. (Lemma 8 is used in the proof of Lemma 10 with elements being agents.)

▶ **Lemma 8.** *Let $\mathcal{M} = (X, \mathcal{I})$ be a matroid and $S \in \mathcal{I}$. Let $T_0 = \{i'_1, i'_2, , \cdots, i'_k\} \subseteq X$ and $T_k = \{i_1, i_2, \cdots, i_k\} \subseteq X$ and their switching sets $T_\ell = \{i_1, i_2, \cdots, i_\ell, i'_{\ell+1}, \cdots, i'_k\} \in \mathcal{I}$, $\ell \in [k-1]$. Suppose $S + T_\ell \in \mathcal{I}$, for any $\ell = 0, 1, \cdots, k$. We also have $S + \{i_1, \cdots, i_\ell, i'_\ell\} \notin \mathcal{I}$, for any $\ell \in [k]$. Then, for any element j (which is not in $S + T_0 + T_k$),*

(i) *if $S + T_k + j \in \mathcal{I}$ then $S + T_0 + j \in \mathcal{I}$;*

(ii) *if $S + T_0 + j \notin \mathcal{I}$ then $S + T_k + j \notin \mathcal{I}$ (contrapositive proposition of* (i)*).*

Proof. For (i), by the exchange property of matroids, there exists a $y \in T_k + j$ such that $S + T_0 + y \in \mathcal{I}$. If $y \neq j$, suppose $y = i_\ell$ for some $\ell \in [k]$. Then we have $T = S + \{i_1, \cdots, i_\ell\} \in \mathcal{I}$ (from $S + T_k \in \mathcal{I}$) and $T' = S + \{i'_1, \cdots, i'_\ell, i_\ell\} \in \mathcal{I}$ (from $S + T_0 + y \in \mathcal{I}$). Then by the exchange property of matroids, there exists $y' \in T' - T$ such that $T + y' \in \mathcal{I}$. Let $y' = i'_{\ell'}$ for some $\ell' \leq \ell$. Consequently, $S + \{i_1, i_2, \cdots, i_{\ell'}, i'_{\ell'}\} \in \mathcal{I}$, which leads to a contradiction. ◀

▶ Remark. Although in Lemma 8 we require two initial sets with the same cardinality, we can relax the requirement to allow them to have different cardinalities. Namely, let $T_0 = \{i'_1, i'_2, \cdots, i'_{k'}\}$ and $T_k = \{i_1, i_2, \cdots, i_k\}$ with $k' \leq k$ (the orders of elements in two sets are fixed). The switching sets T_ℓ, $\ell \in [k']$ between T_0 and T_k are defined as: $T_\ell = \{i_1, i_2, \cdots, i_\ell, i'_{\ell+1}, \cdots, i'_{k'}\}$, $\ell \in [k'-1]$, and $T_{k'} = T_k$. Lemma 8 still holds.

Given $\tau \in \Pi$, let μ^τ be the matching obtained by TMHA under τ, and let $S_a(\tau)$ be the set of agents matched to $a \in A$ by TMHA. Let, for any $t \in [n_1]$, $S_a^t(\tau) \subseteq S_a(\tau)$ be the top t agents in $S_a(\tau)$, i.e., $S_a^t(\tau) = \{i \in S_a(\tau) \,|\, \tau^{-1}(i) \leq t\}$. Then $S_a(\tau) = S_a^{n_1}(\tau)$. Let τ_{-i} be order τ after removing agent i from τ. Similarly, define $S_a^t(\tau_{-i})$ and $S_a(\tau_{-i}) = S_a^{n_1-1}(\tau_{-i})$. When we say "*under* τ", we mean the process of running TMHA under order τ. For any $a \in A$, let $F_a^t = S_a^t(\tau_{-i}) \cap S_a^t(\tau)$. $T_0^t = S_a^t(\tau_{-i}) - S_a^t(\tau) = \{i'_1, i'_2, \cdots, i'_\ell, i'_{\ell+1}, \cdots, i'_{k'}\}$, $k' = |T_0^t|$. Let $T_k^t = S_a^t(\tau) - S_a^t(\tau_{-i}) = \{i_1, i_2, \cdots, i_\ell, i_{\ell+1}, \cdots, i_k\}$, $k = |T_k^t|$. We will show in the next lemma that $k \in \{k', k'+1\}$, for any $t \in [n_1]$ and $a \in [n_2]$. Note that this includes the case $k' = 0$ (in a sense that Property $\mathcal{P}$ (i)-(iv) below trivially holds). The orders in T_0^t and T_k^t

both follow order τ. We have the switching sets $T^t_\ell = \{i_1, i_2, \cdots, i_\ell, i'_{\ell+1}, \cdots, i'_{k'}\}$ between T^t_0 and T^t_k, $\ell \in [k'-1]$, and $T^t_{k'} = T^t_k$. We finally define Property $\mathcal{P}$ as follows.

▶ **Definition 9.** Let $E' = \{(i', \mu^\tau(i')), (i', \mu^{\tau_{-i}}(i')) : i' \in N, i'$ matched by μ^τ and $\mu^{\tau_{-i}}\}$. We say that Property $\mathcal{P}$ holds for μ^τ and $\mu^{\tau_{-i}}$ if: $C = ((\mu^\tau \oplus \mu^{\tau_{-i}}) \cap E') \cup \{(i, \mu^\tau(i))\}$ is an alternating path starting from agent i if it is non-empty, and for any $a \in A$, and $t \in [n_1]$:

- **(i)** $F^{n_1}_a + T^t_\ell \in \mathcal{I}_a$, for any $\ell \in [k']$;
- **(ii)** $F^t_a + \{i_1, \cdots, i_\ell, i'_\ell\} \notin \mathcal{I}_a$, $\ell \in [k']$;
- **(iii)** For any agent s, if $S^t_a(\tau) + s \in \mathcal{I}_a$, then $S^t_a(\tau_{-i}) + s \in \mathcal{I}_a$ (or equivalently, $S^t_a(\tau_{-i}) + s \notin \mathcal{I}_a$ implies that $S^t_a(\tau) + s \notin \mathcal{I}_a$).
- **(iv)** $\tau^{-1}(i_1) < \tau^{-1}(i'_1) < \tau^{-1}(i_2) < \tau^{-1}(i'_2) < \cdots < \tau^{-1}(i_{k'}) < \tau^{-1}(i'_{k'})$

We are ready to give our characterization lemma in the following.

▶ **Lemma 10.** *Property $\mathcal{P}$ holds for unweighted agents with strict preference orders.*

Lemma 10 can be used to directly prove that RTMHA is $\frac{e}{e-1} + o(1)$-approximate for unweighted agents without ties. Analysis in Subsection 4.2 builds on developments in this section.

4.2 Unweighted Agents with Ties

When agents have ties without weights, we will present a similar characterization in Lemma 11 (by alternating path) of symmetric differences between the matching obtained by TMHA under some order τ of all agents and the matching obtained by TMHA under τ_{-i} when an agent i is absent. We will prove that in any step of the two processes of TMHA under τ and under τ_{-i}, there is an injective map from the equivalence class of matchings generated by TMHA under τ to the class generated by TMHA under τ_{-i}. This injective map is such that each pair of the corresponding matchings from these two classes satisfies Property $\mathcal{P}$.

For any $t \in [n_1]$, let $\mathrm{CL}^t(\mu^\tau)$ denote the equivalence class of matchings equivalent to the matching found by TMHA under τ until tth iteration. Precisely, if $\mu^\tau_{\leq t} = \{(i, \mu^\tau(i)) : \tau^{-1}(i) \leq t\}$ is the matching μ^τ restricted to the first t agents of τ, then: $\mathrm{CL}^t(\mu^\tau) = \{\mu \in G \,|\, \mu \simeq \mu^\tau_{\leq t}\}$. Let $\mathrm{CL}(\mu^\tau) = \mathrm{CL}^{n_1}(\mu^\tau)$. When we consider the process of running TMHA under τ_{-i}, to simplify the notation, we consider an equivalent process that runs TMHA under τ while imposing the condition that agent i is unmatched. Hence, in the following we suppose TMHA running on τ_{-i} is such an equivalent process. Thus, we have $\mathrm{CL}^t(\mu^{\tau_{-i}})$, for any $t \in [n_1]$.

▶ **Lemma 11.** *For any t, there exists an injective map f from $\mathrm{CL}^t(\mu^\tau)$ to $\mathrm{CL}^t(\mu^{\tau_{-i}})$ such that for any $\mu \in \mathrm{CL}^t(\mu^\tau)$, μ and $f(\mu)$ satisfy Property $\mathcal{P}$.*

Proof. The proof is by induction on the iterations of the two processes of TMHA run under τ and under τ_{-i}. First, note that $\mathrm{CL}^s(\mu^\tau) = \mathrm{CL}^s(\mu^{\tau_{-i}})$, for any $s < \tau^{-1}(i)$. Thus each matching from $\mathrm{CL}^s(\mu^\tau)$ can be mapped to itself. This shows that if i is unmatched under τ, the lemma trivially holds. Hence, w.l.o.g., suppose agent i is matched under τ.

Second, for any matching $\mu^{s+1} \in \mathrm{CL}^{s+1}(\mu^\tau)$, there exists a unique matching $\mu^s \in \mathrm{CL}^s(\mu^\tau)$ such that $\mu^{s+1} = \mu^s \cup \{(\tau(s+1), \mu^{s+1}(\tau(s+1)))\}$, for any $s \leq n_1 - 1$.

We now prove the induction base case for $s = \tau^{-1}(i)$. For any matching $\mu^s \in \mathrm{CL}^s(\mu^\tau)$, we define f as $f(\mu^s) = \mu^s_{<s} \in \mathrm{CL}^s(\mu^{\tau_{-i}})$. Now we can see that the symmetric difference of μ^s and $f(\mu^s)$ is a single edge and all parts of Property $\mathcal{P}$ hold. Hence, the base case is true.

Now suppose the lemma holds for $s = k-1$, where $k - 1 \geq \tau^{-1}(i)$, i.e., we have an injective function f: $\mathrm{CL}^{k-1}(\mu^\tau) \to \mathrm{CL}^{k-1}(\mu^{\tau_{-i}})$ such that for any $\mu \in \mathrm{CL}^{k-1}(\mu^\tau)$, μ and

$f(\mu)$ satisfy Property $\mathcal{P}$. Let's see the case $s = k \leq t$. Let $j = \tau(k)$ be the kth agent of τ. Consider this new agent j, and any $\mu \in \mathrm{CL}^k(\mu^{\tau_{-i}})$. Since for any a which is strictly better than $\mu(j)$, i.e., $a \succ_j \mu(j)$, agent j will not be matched to a under μ, which means $j + S_a^{k-1}(\mu) \notin \mathcal{I}_a$, by part (iii) of Property $\mathcal{P}$, $j + S_a^{k-1}(f^{-1}(\mu_{<k})) \notin \mathcal{I}_a$ if $f^{-1}(\mu_{<k})$ exists. Therefore, for any $\mu^{k,\tau} \in \mathrm{CL}^k(\mu^\tau)$ and $\mu^{k,\tau_{-i}} \in \mathrm{CL}^k(\mu^{\tau_{-i}})$, we will have $\mu^{k,\tau}(j) \preceq_j \mu^{k,\tau_{-i}}(j)$. Let $b = \mu^{k,\tau}(j)$ and $c = \mu^{k,\tau_{-i}}(j)$. Note that for each matching $\mu^{k,\tau} \in \mathrm{CL}^k(\mu^\tau)$, we have $\mu^{k,\tau}_{\leq k-1} \in \mathrm{CL}^{k-1}(\mu^\tau)$; similar property holds for $\mathrm{CL}^k(\mu^{\tau_{-i}})$. Hence, we consider two cases:
Case(i). If $b \simeq_j c$, we define $f(\mu^{k,\tau}) = f(\mu^{k,\tau}_{<k}) \cup (j, b)$. We can see that f is well defined since j can be added into $S_b(\mu^{k,\tau}_{<k})$, it also can be added into $S_b(f(\mu^{k,\tau}_{<k}))$ by Property $\mathcal{P}$ (iii). Second, there is no change for alternating path after j is considered. Third, j will be added into the set $S_b(\mu^{k,\tau}) \cap f(S_b(\mu^{k,\tau}))$ and the matched object of the other agents will not change when compared their matchings under τ and under τ_{-i}. Thus, Property $\mathcal{P}$ holds.
Case (ii). If $b \prec_j c$, by induction hypothesis, Property $\mathcal{P}$ (ii), there exists an object $b' \simeq_j c$ such that $j + S_{b'}(\mu^{k,\tau}_{<k}) \notin \mathcal{I}_{b'}$ and $j + S_{b'}(f(\mu^{k,\tau}_{<k})) \in \mathcal{I}_{b'}$. We know the current alternating path (up-to iteration $k-1$) must be with b' as its another end point. Hence, b' is unique. Otherwise, let $d \neq b'$ be the end point of the alternating path. Then the switching agent, i.e., i'_ℓ in Property $\mathcal{P}$ (ii) for object b' is not j, and by Property $\mathcal{P}$ (i), $j + S_{b'}(\mu^{k,\tau}_{<k}) \in \mathcal{I}_{b'}$, a contradiction. Define $f(\mu^{k,\tau}) = f(\mu^{k,\tau}_{<k}) \cup (j, b')$. Now we see that two edges $(j, b') \in \mu^{k,\tau_{-i}}$ and $(j, b) \in \mu^{k,\tau}$ will be added to the alternating path; thus, Property $\mathcal{P}$ holds. The end point of the alternating path is now object b. The last part of the proof of the induction step (omitted here) is to show that sets of agents matched to objects b and b' satisfy Property $\mathcal{P}$ (note that sets of agents matched to other objects do not change). ◀

By Lemma 11, we can suppose w.l.o.g. that there exists a matching in the graph G such that all the agents are matched. (Otherwise, we can find a maximum matching in graph G and remove the unmatched agents in this matching and their adjacent edges. After this, Lemma 11 implies that the expected matching size of RTMHA can not increase (because $|\mu| \geq |f(\mu)|$) and the graph has a matching with all agents matched.) Let μ^* denote this matching in G. Given τ, let τ_i^t denote the order obtained from τ by first removing agent i from τ and then inserting him into the tth position of τ_{-i}, i.e., $\tau_i^t(i) = t$. Now fix $t \in [n_1]$. For any $\tau \in \Pi$, $a \in A$, let $U_a^t(\tau)$ be defined as $U_a^t(\tau) = \{i \in S_a(\mu^*) \mid i \text{ is not matched by TMHA under } \tau_i^t\}$.

▶ **Lemma 12** (INJECTIVITY LEMMA). $|U_a^t(\tau)| \leq |S_a^t(\tau)|$.

Proof. Suppose that $|U_a^t(\tau)| > |S_a^t(\tau)|$. Then by matroid exchange property, there is an agent $i \in U_a^t(\tau)$ such that $S_a^t(\tau) + i \in \mathcal{I}_a$. Suppose i is unmatched by TMHA under τ. Since i is unmatched under τ_i^t, $S_a^t(\tau) = S^t(\tau_i^t)$. However $S_a^t(\tau_i^t) + i = S_a^t(\tau) + i \in \mathcal{I}_a$, implying agent i will be matched to an object at least as good as a by TMHA under τ_i^t, a contradiction.
Suppose now that agent i is matched by TMHA under τ. Removing agent i from τ, we get τ_{-i}. Then inserting i into the tth position of τ_{-i} we get τ_i^t. Since agent i is unmatched by TMHA under τ_i^t, the processes of TMHA under τ_i^t and under τ_{-i} is the same. Let f be the injective function from $\mathrm{CL}^t(\mu^\tau)$ to $\mathrm{CL}^t(\mu^{\tau_{-i}})$ in Lemma 11. By Property $\mathcal{P}$ (iii), $S_a^t(\tau) + i \in \mathcal{I}_a$, then $S_a^t(f(\tau)) + i \in \mathcal{I}_a$. That is $S_a^t(\tau_i^t) + i \in \mathcal{I}_a$, implying agent i will be matched to an object at least as good as a by TMHA under τ_i^t. This leads to a contradiction. ◀

Lemma 12 can be used to directly prove that RTMHA is $\frac{e}{e-1} + o(1)$-approximate for unweighted agents with ties. Analysis in Subsection 4.3 builds on developments in this section.

4.3 Weighted Agents with Ties

Building on Subsection 4.1 and 4.2 we give the proof of the $\frac{e}{e-1}$-approximation for weighted agents with ties. A careful utilization of Lemmas 11 and 12 is needed to obtain a strengthened version of injectivity lemma (Lemma 13). This lemma helps define an injective function from marginal 'bad' events to 'good' events, allowing to prove the ratio. Interestingly, Lemma 11 and primal-dual analysis [22], leads to a significantly simpler proof (compared to [22]) of the ratio $\frac{e}{e-1}$ when each associated matroid is uniform. The main technical ingredient of the proof is a strong version of the injectivity lemma similar to Lemma 12 to define an injective map from marginal set $\bigcup_{t\geq 1} M_t$ (defined later) to the set $2^{\bigcup_{t\geq 1} Q_t}$.

Because of weights, we consider the discrete version of the function $1-g(y)=1-e^{y-1}$ used in the sampling. For every $i\in N$, we will choose an integer $t=\tau(i)$ uniformly at random from $\{1,\cdots,\kappa\}$, where $\kappa\in\mathbb{N}_+$ is a parameter. Let $\psi(t)=1-(1-\frac{1}{\kappa})^{\kappa-t+1}$. The random order of agents follows the decreasing order of $w_i\psi(\tau(i))$, for any $i\in[n_1]$. Note, $\tau\in[\kappa]^{n_1}$ is different from previous subsections (where $\tau\in\Pi$), but we still call it a permutation and the discrete process is the same as RTMHA when $\kappa\to\infty$. For each $\tau\in[\kappa]^{n_1}$, there is a corresponding order of agents defined above. Hence, when we say 'under τ', we mean to run the above discrete RTMHA (called DRTMHA) under decreasing order of $w_i\psi(\tau(i))$, $i\in[n_1]$.

Let $\tau_i^t\in[\kappa]^{n_1}$ denote the same order of agents except we set i's τ value to t, i.e., $\tau_i^t(i)=t$ and $\tau_i^t(k)=\tau(k)$, for any agent $k\neq i$. The other notions have the same meaning as before unless explicitly redefined. Let μ^* be the maximum weighted matching on G, i.e., the optimal matching to our problem. Recall, $S_a(\mu^*)$ is the set of agents matched to a under μ^*. Let Q_t be the set of all the triples among permutations, τ values and agents such that the agent with τ value t at the current order is matched by TMHA. Precisely,

$$Q_t=\{(\tau,t,i)\,|\,i \text{ is matched by TMHA under } \tau \text{ and } \tau(i)=t,\ i\in N,\tau\in[\kappa]^{n_1}\}.$$

Let R_t be the set of all the triples among permutations, t values and agents such that the agent matched in μ^* with τ value at the current order is unmatched. That is:

$$R_t=\{(\tau,t,i)\,|\,i \text{ is unmatched under } \tau \text{ and } \tau(i)=t, \text{ and } i \text{ is matched in } \mu^*,\tau\in[\kappa]^{n_1}\}.$$

Our goal now is to define an injective map from R_t to $2^{\bigcup_{s\leq t} Q_t}$. Towards this aim, we define the marginal (loss) set M_t, which is a subset of R_t with marginal property, i.e., an agent in M_t will be matched after decreasing his τ value by one: $M_t=\{(\tau,t,i)\in R_t\,|\,i \text{ is matched under } \tau_i^{t-1}\}$. Let $M_0=\emptyset$. Let $OPT=w(\mu^*)$ be the optimal weight and $B=\frac{OPT}{\kappa}$. Let $w(Q_t)$ denote the total weights of agents for triples in Q_t: $w(Q_t)=\sum_{(\tau,t,i)\in Q_t} w_i$. Similarly, define $w(R_t)$ and $w(M_t)$. Let $x_t=\frac{w(Q_t)}{\kappa^{n_1}}$ be the expected weights gained by DRTMHA on all agents with τ value t, and $y_t=\frac{w(M_t)}{\kappa^{n_1}}$ the expected marginal loss of all agents with τ value t. Since each agent appears with τ equal to $t\in[\kappa]$ in DRTMHA with equal probability $\frac{1}{\kappa}$, if $Q_t'\subseteq Q^t$ is the set of triples with agents matched in μ^*, then $\frac{w(Q_t')}{\kappa^{n_1}}+\frac{w(R_t)}{\kappa^{n_1}}=B$, for any $t\in[\kappa]$. Therefore,

$$x_t+\frac{w(R_t)}{\kappa^{n_1}}\geq B. \tag{1}$$

So the expected weight of matching obtained by DRTMHA is $\sum_{t\in[n_1]} x_t$. There is a bijection $h: R_t\to\bigcup_{s\leq t} M_s$. Note, for any $(\tau,t,i)\in R_t$, if there is $s\in[t-1]$ s.t. $(\tau_i^s,s,i)\in Q_s$, then there is such s that is a unique maximal one. Note, in this case $(\tau_i^{s+1},s+1,i)\in M_{s+1}$. Define $h(\tau,t,i)=(\tau_i^{s+1},s+1,i)$. Otherwise, define $h(\tau,t,i)=(\tau_i^1,1,i)\in M_1$; note that

$M_0 = \emptyset$. (h is well defined.) For any $(\tau, s, i) \in M_s$, and some $s \leq t$, agent i will not be matched under τ_i^t since it is unmatched under τ, by Lemma 11 (Property $\mathcal{P}$ (iii)). Then $(\tau_i^t, t, i) \in R_t$, which means h is surjective, i.e., $h(\tau_i^t, t, i) = (\tau, s, i)$. If $h(\tau, t, i) = h(\tau', t, i') = (\sigma, s, i'') \in M_s$, by definition $i = i' = i''$ and $\tau = \tau' = \sigma_{i''}^t$. Then h is also injective. Hence,

$$w(R_t) = \sum_{s \leq t} w(M_s) \tag{2}$$

For any $\tau \in [\kappa]^{n_1}$, $a \in A$, let $S_a(\tau)$ be the set of agents matched to a by DRTMHA under τ. Let $U_a(\tau)$ be the set of agents in $S_a(\mu^*)$ such that changing the τ of any one of them, e.g., agent $i \in S_a(\mu^*)$ to some t_i, the new triple of τ together with his new value t_i and himself belongs to M_{t_i} (Note that for any agent $i \in S_a(\mu^*)$, if there exists a t_i such that $(\tau_i^{t_i}, t_i, i) \in M_{t_i}$, then t_i is unique for agent i). Precisely, $U_a(\tau) = \{i \in S_a(\mu^*) \,|\, (\tau_i^{t_i}, t_i, i) \in M_{t_i}\}$.

▶ **Lemma 13** (INJECTIVITY LEMMA). *There exists an injective function $g_{a\tau}$: $U_a(\tau) \to S_a(\tau)$ such that $w_i\psi(t_i) \leq w_{i'}\psi(\tau(i'))$, for any $i \in U_a(\tau)$ and $i' = g_{a\tau}(i)$.*

Proof (Sketch). From map h: $R_t \to \bigcup_{s \leq t} M_s$, we know $U_a^t(\tau)$ is well defined. Now, for any $i \in U_a^t(\tau)$, by definition, i is unmatched under $\tau_i^{t_i}$. Let ℓ_i denote the position where agent i sits in the order of agents of DRTMHA under $\tau_i^{t_i}$ (we call it a position value). We can partition $U_a^t(\tau)$ into different groups such that different groups have different position values ℓ_i's, and agents in the same group have the same position value. That is, partition set $U_a^t(\tau)$ into n_3 such groups $\{T_s\}_{s \in [n_3]}$, where T_s has position value ℓ_s.

Note that if i is unmatched when his position value is ℓ_i, then he will remain unmatched when he is in any position $\ell > \ell_i$, by Lemma 11 (part (iii) of Property $\mathcal{P}$). We now use Lemma 12 iteratively on set $\bigcup_{j \leq s} T_j$, $s \in [n_3]$ to prove Lemma 13 as follows: suppose we have constructed a map $g_{a\tau}$ on set $\bigcup_{j \leq s} T_j$ satisfying the property of Lemma 13. Then by Lemma 12, there exists an injective map $f_{a\tau}^{\ell_{s+1}}$: $\bigcup_{j \leq s+1} T_j \to S_a^{\ell_{s+1}}(\tau)$, satisfying the property of the lemma. Combining the two maps $g_{a\tau}$ and $f_{a\tau}^{\ell_{s+1}}$, we obtain the map on $\bigcup_{j \leq s+1} T_j$. ◀

Next we define a map H from $\bigcup_{t \in [\kappa]} M_t$ to $2^{\bigcup_{t \in [\kappa]} Q_t}$ to bound the marginal 'bad' events $\bigcup_{t \in [\kappa]} M_t$ by 'good' events $\bigcup_{t \in [\kappa]} Q_t$ as follows: For any $t \in [\kappa]$, $(\tau, t, i) \in M_t$,

$$H(\tau, t, i) = \{(\tau_i^s, p_i^s, g_{a\tau_i^s}(i)) \,|\, a = \mu^*(i) \text{ and } p_i^s = \tau_i^s(g_{a\tau_i^s}(i)) \text{ and } s \in [\kappa]\}.$$

Note that for any $(\tau, t, i) \in M_t$, $|H(\tau, t, i)| = \kappa$. Due to the injectivity of $g_{a\tau}$ (Lemma 13), we will show that the image sets of H are disjoint.

▶ **Lemma 14.** *For any $(\tau, t, i) \in M_t$ and $(\tau', t', i') \in M_{t'}$, if $(\tau, t, i) \neq (\tau', t', i')$, then $H(\tau, t, i) \cap H(\tau', t', i') = \emptyset$.*

Proof. Suppose there is a triple $(\sigma, s, \ell) \in H(\tau, t, i) \cap H(\tau', t', i')$. Then there exists $a \in A$ s.t. $\mu^*(i) = \mu^*(i') = a$, by the definition of $g_{a\sigma}$ (since $S_a(\sigma) \cap S_b(\sigma) = \emptyset$, for any $a \neq b$). By the definition of H, we know that $\tau = \sigma_i^t$ and $\tau' = \sigma_{i'}^{t'}$. Hence, $i, i' \in U_a(\sigma)$ since $(\tau, t, i) \in M_t$, $(\tau', t', i') \in M_{t'}$. As $g_{a\sigma}$ is an injective function from $U_a(\sigma)$ to $S_a(\sigma)$, $g_{a\sigma}(i) = g_{a\sigma}(i')$, implies $i = i'$. As $\tau = \sigma_i^t$, $\tau' = \sigma_i^{t'}$, there exists a unique t such that $(\sigma_i^t, t, i) \in M_t$, which gives that $t = t'$ and $\tau = \sigma_i^t = \sigma_i^{t'} = \tau'$. This shows $(\tau, t, i) = (\tau', t', i')$, and thus a contradiction. ◀

By Lemmas 13 and 14, we have for any $(\tau, t, i) \in M_t$, $t \in [k]$, and any $(\sigma, s, i') \in H(\tau, t, i)$, $w_i\psi(t) \leq w_{i'}\psi(s)$ and $(\sigma, s, i') \in \bigcup_{\ell \leq k} Q_\ell$. Summing over M_t: $\psi(t)w(M_t) =$

$\sum_{(\tau,t,i)\in M_t} w_i\psi(t) \leq \frac{\sum_{(\sigma,s,i')\in H(M_t)} w_{i'}\psi(s)}{\kappa}$. Therefore:

$$\begin{aligned}
\sum_{t\leq\kappa} \psi(t)w(M_t) &= \sum_{t\leq\kappa} \sum_{(\tau,t,i)\in M_t} w_i\psi(t) \\
&\leq \frac{\sum_{t\leq\kappa}\sum_{(\sigma,s,i')\in H(M_t)} w_{i'}\psi(s)}{\kappa} \\
&= \frac{\sum_{(\sigma,s,i')\in\bigcup_{t\leq\kappa} H(M_t)} w_{i'}\psi(s)}{\kappa} \\
&\leq \frac{\sum_{(\sigma,s,i')\in\bigcup_{t\leq\kappa} Q_t} w_{i'}\psi(s)}{\kappa} \\
&= \frac{\sum_{t\leq\kappa}\sum_{(\tau,t,i)\in Q_t} w_i\psi(t)}{\kappa} \\
&= \frac{\sum_{t\leq\kappa}\psi(t)w(Q_t)}{\kappa}.
\end{aligned}$$

Dividing both sides of this inequality by κ^{n_1} implies $\sum_{t\leq\kappa}\psi(t)y_t \leq \frac{\sum_{t\leq\kappa} x_t\psi(t)}{\kappa}$. Combining this inequality with (1) and (2), and following a step in the analysis from [4] implies.

▶ **Theorem 15.** *DRTMHA is universally truthful and has an approximation ratio* $\frac{e}{e-1}+O(\frac{1}{\kappa})$ *for weighted agents with ties and terminates in* $O(n_1^4\gamma\log\kappa)$ *time, for any* $\kappa\in\mathbb{N}_+$.

References

1. A. Abdulkadiroğlu and T. Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3):689–701, 1998.
2. D.J. Abraham, R.W. Irving, and D.F. Manlove. Two algorithms for the student-project allocation problem. *Journal of Discrete Algorithms*, 5(1):73–90, 2007.
3. M. Adamczyk, P. Sankowski, and Q. Zhang. Efficiency of truthful and symmetric mechanisms in one-sided matching. In *Proc. 7th International Symposium on Algorithmic Game Theory (SAGT)*, pages 13–24. Springer, 2014.
4. G. Aggarwal, G. Goel, C. Karande, and A. Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proc. 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1253–1264. ACM-SIAM, 2011. URL: `http://dl.acm.org/citation.cfm?id=2133036.2133131`.
5. M. Babaioff, N. Immorlica, and R. Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proc. 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 434–443. ACM-SIAM, 2007.
6. A. Bhalgat, D. Chakrabarty, and S. Khanna. Social welfare in one-sided matching markets without money. In *Proc. of Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, (APPROX), and 15th International Workshop, (RANDOM)*, pages 87–98, 2011.
7. A. Bogomolnaia and H. Moulin. A new solution to the random assignment problem. *Journal of Economic Theory*, 100(2):295 – 328, 2001.
8. A. Bogomolnaia and H. Moulin. Size versus fairness in the assignment problem. *Games and Economic Behavior*, 90:119–127, 2015.
9. D. Chakrabarty and C. Swamy. Welfare maximization and truthfulness in mechanism design with ordinal preferences. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, pages 105–120, 2014.

10 H. T-H Chan, F. Chen, X. Wu, and Z. Zhao. Ranking on arbitrary graphs: Rematch via continuous LP with monotone and boundary condition constraints. In *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1112–1122. ACM-SIAM, 2014.

11 N. Chen, N. Gravin, and P. Lu. Truthful generalized assignments via stable matching. *Mathematics of Operations Research*, 39(3):722–736, 2014.

12 Y. Chen and T. Sönmez. Improving efficiency of on-campus housing: An experimental study. *American Economic Review*, 92(5):1669–1686, 2002.

13 W.H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM Journal on Computing*, 15(4):948–957, 1986.

14 N.R. Devanur, K. Jain, and R. Kleinberg. Randomized primal-dual analysis of RANKING for online bipartite matching. In *Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 101–107. ACM-SIAM, 2013.

15 S. Dughmi and A. Ghosh. Truthful assignment without money. In *Proc. 11th ACM Conference on Electronic Commerce (EC)*, pages 325–334. ACM, 2010.

16 J. Edmonds. Submodular functions, matroids, and certain polyhedra. *Combinatorial structures and their applications*, pages 69–87, 1970.

17 A. Filos-Ratsikas, S. Kristoffer Stiil Frederiksen, and J. Zhang. Social welfare in one-sided matchings: Random priority and beyond. In *Proc. 7th International Symposium on Algorithmic Game Theory (SAGT)*, pages 1–12, 2014.

18 P. Gärdenfors. Assignment problem based on ordinal preferences. *Management Science*, 20(3):331–340, 1973. doi:10.1287/mnsc.20.3.331.

19 A. Hylland and R. Zeckhauser. The efficient allocation of individuals to positions. *Journal of Political Economy*, 87(2):293–314, 1979.

20 B. Kalyanasundaram and K. Pruhs. An optimal deterministic algorithm for online b-matching. *Theoretical Computer Science*, 233(1):319–325, 2000.

21 R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 352–358. ACM, 1990.

22 P. Krysta, D. Manlove, B. Rastegari, and J. Zhang. Size versus truthfulness in the house allocation problem. In *Proc. 15th ACM Conference on Economics and Computation (EC)*, pages 453–470. ACM, 2014.

23 M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 597–606. ACM, 2011.

24 A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *Journal of the ACM*, 54(5):22, 2007.

25 D. Saban and J. Sethuraman. House allocation with indifferences: a generalization and a unified view. In *Proc. 14th ACM Conference on Electronic Commerce (EC)*, pages 803–820. ACM, 2013.

26 A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer, 2003.

27 L. Zhou. On a conjecture by Gale about one-sided matching problems. *Journal of Economic Theory*, 52(1):123–135, 1990.

Reservation Exchange Markets for Internet Advertising*

Gagan Goel[1], Stefano Leonardi[†2], Vahab Mirrokni[3], Afshin Nikzad[‡4], and Renato Paes-Leme[5]

1 **Google Research, New York, USA**
2 **Sapienza University of Rome, Rome, Italy**
3 **Google Research, New York, USA**
4 **Stanford University, Standford, USA**
5 **Google Research, New York, USA**

Abstract

Internet display advertising industry follows two main business models. One model is based on *direct deals* between publishers and advertisers where they sign legal contracts containing terms of fulfillment for a future inventory. The second model is a spot market based on auctioning page views in real-time on advertising exchange (AdX) platforms such as DoubleClick's Ad Exchange, RightMedia, or AppNexus. These exchanges play the role of intermediaries who sell items (e.g. page-views) on behalf of a seller (e.g. a publisher) to buyers (e.g., advertisers) on the opposite side of the market. The computational and economics issues arising in this second model have been extensively investigated in recent times.

In this work, we consider a third emerging model called *reservation exchange market*. A reservation exchange is a two-sided market between buyer orders for blocks of advertisers' impressions and seller orders for blocks of publishers' page views. The goal is to match seller orders to buyer orders while providing the right incentives to both sides. In this work we first describe the important features of mechanisms for efficient reservation exchange markets. We then address the algorithmic problems of designing revenue sharing schemes to provide a fair division between sellers of the revenue collected from buyers.

A major conceptual contribution of this work is in showing that even though both clinching ascending auctions and VCG mechanisms achieve the same outcome from a buyer perspective, however, from the perspective of revenue sharing among sellers, clinching ascending auctions are much more informative than VCG auctions.

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

Keywords and phrases Reservation Markets, Internet Advertising, Two-sided Markets, Clinching Auction, Envy-free allocations.

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.142

1 Introduction

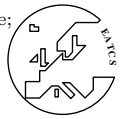
The universe of internet advertisement is divided in two big worlds: *search ads* and *display ads*. At first glance, they look very similar: both sell impressions using variants of the second

* A preliminary version of this work was presented at the 11th Ad Auctions Workshop, Portland, June 2015.

† This work is supported by the Google Focused Award "Algorithms for Large-scale Data Analysis" and by EU FET project MULTIPLEX no. 317532. Work partially done while the author was a visiting scientist at Google Research, New York.

‡ Work partially done while the author was on internship at Google Research, New York.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 142; pp. 142:1–142:13



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

price auction with reserves. A closer look, however, reveals that they are very different: in *search ads*, the platform (Google or Bing, for example) is both the *auctioneer* and the *seller*, i.e., it sells inventory in their own properties. This makes it a one-sided market design problem, or in other words, the designer needs to reason only about the incentives of the buyers. In *display ads*, the platform is auctioning inventory not owned by them, turning it into a two-sided market design problem, where incentives for buyers (advertisers) and sellers (publishers, such as websites, blogs and news portals) need to be balanced.

Designing practical markets for display ads is challenging, since the theory of market design is much more developed for one-sided markets (there are tools like VCG, Myerson's optimal auction, ...) while for two-sided markets, what classic auction theory offers are mainly impossibility results, such as the Myerson-Satterthwaite Impossibility Theorem [23].

One complicating factor in display ads is that while we are used to think of internet advertisement in the form of auctions, auctions are only the tip of the iceberg. The most premium inventory is sold via the *reservations market* (also called *direct deals* or *guaranteed contracts*.) In this method, a publisher and an advertiser make a deal to allocate a certain number of impressions over a certain period, for a pre-specified price per impression. This deal is made offline in advance for a future inventory. Direct deals are known to suffer from inefficiencies for two reasons: first is the manual nature of formation of these contracts which allows a publisher to sign deals with a small number of advertisers, thus creating allocation inefficiencies. The second reason is the manual negotiation between buyers and publishers which incurs a huge cost and lowers the overall efficiency.

Auctions are fully automated and don't suffer from any of these inefficiencies. On the other hand, they can't guarantee to buyers and sellers the certainty that reservations can. For example, a brand launching a large campaign to advertise a new product certainly benefits from the certainty (both in terms of cost and volume of impressions) provided by reservation contracts.

The idea of *automated reservation market* is to overcome the shortcomings of both auctions (real-time spot market) and traditional reservations (offline negotiation). Such markets would allow sellers and buyers to transact for a bulk inventory in advance. This is inspired by a number of recent two-sided markets for online advertising, e.g., an exchange for future contracts[1]. In particular, we study a two-sided market, which we call a *reservation market*, where publishers can post offers for blocks of ad slots characterized by parameters like supply level, reserve price, and their preference for a set of advertisers. Advertisers post requests for ad slots defined by parameters like valuation, demand, and targeting constraints specifying where and when they want to show their ads. The role of the reservation mechanism is to match the seller and buyer orders that attain some economic objectives.

Note that, unlike ad exchange markets that offer impressions available on the spot, the reservation market offers guaranteed deals for an inventory available in the future. Moreover, the reservation market brings together multiple publishers and advertisers with the goal of reducing the intermediation costs and the underlying inefficiencies of one-to-one deals, also by selling bundles of inventories from different publishers.

In this work, we start the investigation of the economics and algorithmic principles that are central to these reservation markets. The major questions we address in this work are: What features and incentive properties form the basis of a successful reservation market? What are the economic objectives of a reservation market? What are the revenue sharing policies that we can employ? What are the algorithmic problems we need to address in the design of reservation mechanisms?

[1] `http://www.massexchange.com/`

1.1 Our contribution

Our main contribution is to propose a formal model of reservation exchange market and discuss what are desirable properties (referred to as *axioms*) for this market. Secondly, we propose two specific mechanisms that help us understand the extent to which some of the aforementioned properties can be simultaneously achieved. We also provide several algorithmic results for the two mechanisms that we study.

Axioms for a reservation exchange market. In Section 2, we provide a simple and clean abstraction of reservation exchange markets for display ads as a two-sided matching market with buyer orders on one side and seller orders on the other side and in Section 3 we identify a list of axioms that we wish any mechanism for these markets to approximately satisfy.

The first axiom for the reservation mechanism that we discuss is the *efficiency of the market,* i.e., the social welfare of all agents of the market. The agents of the market are sellers and buyers, both with quasi-linear utilities. Buyers aim at maximizing their utility, i.e., the total value of the inventory received minus price. Sellers aim at maximizing revenue minus reservation price. The mechanism decides on the allocation to buyer orders of the inventory supplied from each seller, a payment to be charged to each buyer and a distribution of the revenue among sellers.

Individual rationality (IR) requires that participating in the mechanism is beneficial to all agents. *Incentive compatibility* (IC) requires that truthfully reporting one's preferences to the mechanism is the best strategy for each agent, independently from what the other agents report.

Budget Balance (BB) states the payments of the advertisers must entirely and exclusively be transferred to the publishers, i.e., the buyers and the sellers are allowed to trade without leaving to the mechanism any share of the payments, and without the mechanism adding money into the market. This axiom might appear strange at first glance, but it reflects a business practice common to most exchanges, which is of the exchange to get a fixed cut (typically called *revenue sharing margin*) of the seller's revenue. The reasoning behind this rule is that sellers have the option to send their inventory to different exchanges and keeping the revenue sharing margin fixed helps the exchange to be perceived as fair and hence attract more seller's inventory. Since fixed margins are an industrial standard in the ads world, any practical mechanism must implement some of that. Fixed margins are equivalent to budget-balance applied to bids rescaled by the revenue sharing margin up to rescaling bids.

An ideal goal in a reservation exchange is to design IR, IC, BB mechanisms that maximize the social welfare of all agents in the market. Unfortunately, Myerson and Satterthwaite [23] proved impossibility for an IR, IC, and BB mechanism that maximizes social welfare in such a market. The direction we pursue in this work is to ensure full efficiency and incentive compatibility for buyers as the most desirable goal for advertiser. This can be achieved for one-side markets by the family of Vickrey-Clarke-Groves (VCG) mechanisms [6, 29]. As extensively discussed in this work, the main problem with the vanilla VCG allocation is that it does not offer any good incentives to sellers, e.g., VCG can match fungible inventories from different sellers to buyers that offer very different payments thus producing the feeling that revenue is unfairly distributed among sellers.

An alternative to enforcing incentive compatibility for sellers is to design a mechanism that leads to a fair distribution of mechanism's revenue among sellers. Envy-free allocations [28] and other notions of market equilibria have often been considered in markets that cannot achieve full efficiency with truthful allocations. This leads to our third axiom as follows: sellers should not envy each other with respect to the revenue that is received from the mechanism.

Most of the first part of the paper will be devoted to the discussion of which definition of *envy-freeness among sellers* is meaningful for reservation exchange markets. Our conclusion is that such a definition should crucially rely on the buyer-seller transactions that can arise in an efficient allocation, while it should disregard revenue that can only be obtained from allocations with suboptimal social welfare. With this aim, when introducing our notion of envy-freeness between sellers, we define the concept of *clinching graph* as the collection of buyer-seller transactions that can arise in a VCG allocation. We conveniently define and compute the clinching graph by resorting to the implementation of VCG through an ascending clinching auction.

A major insight of this paper is that while the usual description of VCG payments as externalities imposed by agents on others offers little clue on how to split the proceeds of the auction among the sellers, the alternative description of VCG as an ascending auction (Ausubel's clinching framework [1]) provides additional structure obtained from the execution of the auction that can be exploited to design revenue sharing schemes. The clinching auction returns not only bundle prices, but the order in which each item was sold and the price at which the sale occurred. In an ideal case, whenever a *clinch* happens, if the clinching auction points to a unique item to be clinched, then there is a unique way to split the revenue among the sellers (and in this case clinching auctions capture the full information in how to split the revenue). However, sometimes, when a clinch happens, there are multiple items that can be used for that clinch. This is precisely the case when the clinching auction, even though it provides more information than VCG, it doesn't lead to a unique revenue sharing scheme, and we rely on a notion of envy-freeness for the revenue sharing scheme.

Finally, we discuss further desirable properties of reservation markets as additional axioms, and study our proposed mechanisms for their satisfaction of these axioms. These axioms are stability properties that prevent the market to be manipulated from buyers or sellers. We define the concept of *buyer monotonicity (BM)* as the property that the revenue of all sellers does not decrease when new buyer orders are presented. A second property called *seller monotonicity (SM)* states that the increase of the reservation price of a seller is not responsible of the decrease of the revenue of another seller.

Algorithmic results. We restrict our attention to buyer incentive-compatible efficient mechanisms based on VCG allocations. For all these mechanisms truth-telling is a dominant strategy for buyers. The major issue we face is to complement the VCG mechanism with a suitable envy-free revenue sharing scheme between sellers. Our first result is actually a negative result:

- There exists no efficient revenue sharing scheme that is both envy-free and budget balance. We actually demonstrate that any envy-free revenue sharing scheme cannot distribute more than a $\sqrt{3}-1+\delta$ share of the total revenue, for any $\delta > 0$.

Given the impossibility result above, we investigate the possibility of finding good trade-offs between budget balance or envy freeness. Relaxing one of these two constraints imply that either the mechanism is able to distribute a guaranteed share of the total revenue or that any seller has only limited envy of any other seller. With this goal in mind, we propose two revenue sharing schemes: (i) a *revenue sharing by the clinching auction (CA)*, and (ii) a *revenue sharing by the eating mechanism (EM)*.

For the CA revenue sharing scheme, we prove the following desirable properties:

- CA is budget balance.
- CA is 1/2-envy free and this bound is tight.
- CA is budget monotone and seller monotone.

Finally, for the EM revenue sharing scheme, we prove that following three results:

- EM is envy-free.
- EM is at most $\frac{11+\epsilon}{12+\epsilon}$ budget balance, for any $\epsilon > 0$.
- EM is at least $\frac{e-1}{e}$-budget balance.

Most of the technical proofs are given in the full version of the paper.

1.2 Related work

Double Auctions. Double auctions are special cases of two-sided markets with unit-supply buyers and sellers. Myerson and Satterthwaite [23] proved that it is impossible to obtain an IR, Bayesian IC[2], and weak BB [3] mechanism to maximize social welfare in double auctions. Since then, much of the literature has focused on trading off social welfare for buyers and sellers, incentive compatibility and budget balance for double auctions [21, 25, 26, 11, 7]. The seminal work on double auctions [21] shows that efficiency for both buyers and sellers can actually be achieved asymptotically in large markets. In the context of one-shot auctions, optimal auctions for two-sided settings has been studied by [23], following the Nobel-prize winning work of [22]. The problem of finding the right trade-offs between social welfare, IC and BB is largely open for two-sided markets that model reservation exchanges. In this work we investigate two-sided markets that achieve IC for buyers and envy-freeness for sellers. This follows a line of work that looks at envy-freeness and other market equilibria if social welfare cannot be optimized truthfully [14, 19, 28]. Recently, this literature has also been adopted to design the optimal revenue sharing double auctions in the context of advertising exchanges [18]. In this paper, we focus on two-sided markets where multiple buyers are allocated to multiple sellers and the allocation and pricing are done differently. Other than online advertising systems, optimal two-sided markets can be applied to online and offline retailers and e-commerce websites like Amazon and eBay. A very recent paper by [20] studies EBay's double auction problem, but their setting is different from this paper as they consider one buyer and multiple sellers, and explore approximately optimal pricing schemes for this setting.

Clinching Ascending Auctions. One fundamental component of this work is the use of the structure that can be obtained from the execution of Ausubel's clinching auction [1] in designing revenue sharing schemes for the sellers. The clinching auction has been very successful in a variety of scenarios: designing auctions with budget constraints [10, 13, 8, 16], designing online auctions [17], extracting revenue in settings with budgets [3, 9]. The current paper adds to this line of work by showcasing another application of the clinching framework.

Cooperative games. Cooperative game theory may provide insights for modeling the fair sharing of revenue in the ad reservation exchange market. Shapley value [27] is a widely adopted notion of fair division between agents of the value of a game. It is however hard to extend this concept to our model since a crucial axiom of Shapley value (summability) does not hold in our case. Similar difficulties can also be found while trying to design a revenue sharing scheme that results in an attribution that lies in the core of a game [15]. On the positive side, we mention that the revenue sharing scheme by the clinching auction we

[2] Bayesian incentive compatibility is a less restrictive form of incentive compatibility.
[3] Weak budget balance allows the mechanism to retain a share of the payments while not subsidizing the market.

present resembles Shapley values since it is defined as the expected revenue obtained over all possible seller permutations.

Market Equilibria. Several notions of equilibrium in markets have been studied. In a Walrasian equilibrium, we have item prices such that every agent receives a bundle of items that maximizes her utility, the market clears, and the corresponding outcome is efficient. However, except from very special cases (e.g. unit demand buyers), it can't be converted to a mechanism that is incentive-compatible for the buyer [19, 12]. Envy-freeness for buyers is also a concept widely used to characterize the stability of allocations. We do not survey here the extensive literature on this topic. However, we notice that we instead adopt the notion of envy-freeness to characterize fair revenue sharing schemes between sellers.

Reservation-based Internet advertising has also previously considered with more optimization-related questions than mechanism design questions. Examples of this line of work that is quite unrelated to the scope of this paper can be found in [30, 4, 24]. Markets that combine characteristics of the spot market and of direct deals between publishers and sellers have been also considered in [5] with the goal of maximizing the revenue of one single publisher.

2 Preliminaries

We consider a two-sided market, referred to as a *reservation exchange market*, consisting of a set B of buyers and a set S of sellers. Each seller $s_i \in S$ holds a supply of ℓ_i units of an *indivisible* good and has a reserve price ρ_i for each unit of those goods. Each buyer is interested in purchasing at most d_i units and has a value v_i per unit. The structure of the matching market is captured by a bipartite graph $G = (B \cup S, F)$ which indicates which buyer is interested in buying goods from which seller.

For example, in the case of internet advertisement, each buyer $b_j \in B$ corresponds to an advertiser and a seller $s_i \in S$ corresponds to a publisher. An edge $(b_j, s_i) \in F$ indicates that buyer b_j is interested in purchasing inventory from the publisher s_i's website. We define $B_i = \{b_j \in B; (b_j, s_i) \in F\}$ as the set of buyers who target seller s_i inventory. Similarly, we define $S_j = \{s_i \in S; (b_j, s_i) \in F\}$ as the set of sellers that are targeted by buyer b_j.

We are interested in designing *reservation exchange mechanisms* (or simply reservation mechanisms) which associate for any given matching market described by (B, S, v, d, ℓ, ρ) an outcome composed of:

- an allocation $x_i[j] \in \mathbb{Z}$, indicating how many goods from seller s_i are sold to buyer b_j, respecting demands $a_j := \sum_i x_i[j] \leq d_j$ and supply $c_i := \sum_j x_i[j] \leq \ell_i$.
- a total amount P_j paid by each buyer b_j, such that the payment per unit doesn't exceed buyer j's value per unit: $P_j \leq a_j \cdot v_j$
- a revenue sharing scheme which allocates for each buyer b_j and seller s_i, a portion $R_i[j]$ of the buyer's payment P_j to seller s_i, such that $\sum_i R_i[j] \leq P_j$. We define $R_i := \sum_j R_i[j]$ to be the total revenue obtained by seller i.

An outcome satisfying the properties above is said to be a *feasible outcome*. Given a feasible outcome, the utility of involved agents are as follows:

- buyers have quasi-linear utilities, i.e, $u_j = v_j \cdot a_j - P_j$.
- sellers have the revenue minus the reservation price $R_i - \rho_i \cdot c_i$ as their utility.

In the next section, we discuss a set of desirable properties for a reservation mechanism and discuss which subsets of those properties can be simultaneously satisfied.

3 Axioms for Reservation Exchange Markets

In this section, we develop an axiomatic approach to reservation markets. First, we define a set of desirable properties, referred to axioms, for a well-designed market. As it is the case with axiomatic approaches, some seemingly innocuous axioms might generate impossibility results and some seemingly powerful axioms might not prevent the pitfalls we intended. Here, we define a family of axioms (many with different variations) and discuss, using examples, their strengths and weaknesses.

3.1 Fundamental Axioms: efficiency and budget-balance

We establish as our first and most important goal the maximization of market efficiency, which is the sum of the utilities of all agents involved:

- **Efficiency [Eff]:** The implemented outcome maximizes $SW(B \cup S) = \sum_{j \in B} v_j \cdot a_j + \sum_{i \in S} \rho_i \cdot (\ell_i - c_i)$ among all feasible outcomes, assuming the seller derives utility ρ_i for unsold items.

In order to simplify notation, for each seller that has a reserve price $\rho_i > 0$ we add a *proxy buyer* $j(i)$ with demand $d_{j(i)} = \ell_i$ and value $v_{j(i)} = \rho_i$. Let also $j(i)$ be the endpoint of a single edge connecting it to seller i. Notice that there is a social-welfare-preserving one-to-one map between outcomes for sellers with reserve prices and sellers without reserves but with proxy buyers. This reduction allows us to ignore from this point on the reserve prices ρ_i and focus on maximizing $\sum_i v_j \cdot a_j$ as the [Eff] goal.

A second goal of the mechanism is to distribute the revenue between sellers. A budget balance mechanism should distribute the entire revenue collected from the buyers to the sellers. A β-budget balance mechanism should distribute at least a β-fraction of the revenue.

- **β-Budget Balance [β-BB]:** The implemented outcome is β-budget balance for a constant $\beta \in [0, 1]$ if $\sum_{i \in S} R_i \geq \beta \sum_{j \in B} P_j$. We say that the reservation mechanism is exact budget balance if $\beta = 1$.

3.2 Stability properties

A second set of properties describes the stability of the allocation and resilience to manipulation via adding or removing buyers or sellers:

- **Buyer Monotonicity [BM]**: If a new buyer order b_j is added, the revenue of all sellers in S_j does not decrease.
- **Seller monotonicity [SM]**: If a seller increases his reserve price, the revenue of all other sellers does not decrease.

3.3 Incentive compatibility

We next define a set of desirable incentives properties for buyers and sellers.

- **[B-IC] Buyer incentive compatibility:** Buyers maximize their utility by reporting their true valuations to the mechanism.
- **[S-IC] Seller incentive compatibility:** Sellers maximize their utility by declaring true reserve prices and supply levels.

Unfortunately, [B-IC] and [S-IC] cannot be simultaneously achieved in a two-sided market [23, 21] if not at the expense of efficiency. Here, we choose to relax seller incentive

compatibility and instead, enforce a fairness constraint among sellers while keeping buyer incentive compatibility.

If we enforce [Eff] and [B-IC], the only mechanism available to decide on the allocation and buyer payments is the VCG mechanism. VCG, however, treats all the sellers as one and therefore doesn't prescribe how the revenue of the auctions should be distributed among the sellers. The central issue in the design of reservation exchange mechanisms is how to distribute the revenue from the VCG auction in a manner that is fair to the sellers. As we will see next, defining a precise notion of fairness that matches our intuition is a quite non-trivial task. First, we show how the most natural definitions fail to capture important situations.

3.4 Seller Fairness and Envy-Freeness

We start by identifying a set of properties that we believe a fair revenue sharing scheme should satisfy. The challenge here is in identifying, when a buyer gets some item at price p, if a seller can stake a claim on this revenue or not. Firstly, a seller s_i can claim revenue only from buyers that are interested in the inventory owned by seller s_i, i.e., $R_i[j] = 0$ for $j \notin B_i$. Also, if a certain buyer b_j never receives goods from seller s_i under any efficient allocation, seller s_i shouldn't be able to claim stake on the revenue of buyer b_j. This is so because even if this seller drops this connection to the buyer, it won't change the set of efficient outcomes. Moreover, the seller may end up getting a lower revenue because of the reduced competition after dropping such a connection.

In order to capture the above notions, we define for each seller s_i, the set $A_i \subseteq B_i$ as the set of buyers that are allocated at least one good from s_i in *some* efficient allocation. We are now able to define the concept of envy-free revenue sharing: roughly speaking, we say that a revenue sharing scheme is envy-free if each seller s_i extracts from A_i more revenue then any other seller with at most the same supply and proportionally more revenue than any seller with higher supply. More specifically, this concept is defined as follows:

- **Envy-free Revenue Sharing [ERS]**: $\forall s_i, s_{i'}$, $\sum_{j \in A_i} R_i[j] \geq \min(1, \frac{\ell_{s_i}}{\ell_{s_{i'}}}) \cdot \sum_{j \in A_i} R_{i'}[j]$.

If all sellers have the same supply, this boils down to $\sum_{j \in A_i} R_i[j] \geq \sum_{j \in A_i} R_{i'}[j]$. We note that if one is able to design an envy-free mechanism for sellers with unit-supply, this automatically extends to sellers with non-unit supply by the following reduction: transform each seller of supply ℓ_i in ℓ_i unit supply sellers. An envy free allocation in the transformed setting naturally translates to an envy-free allocation in the original setting. For this reason we will assume for the remainder of the paper that sellers are unit supply.

However, as shown in the following example, the above notion of envy-freeness doesn't fully capture the notion of a fair allocation among the sellers:

▶ **Example 1.** Consider two buyers with valuation $v_1 = 2$, $v_2 = 1$, and demands $d_1 = 2$, $d_2 = 1$. There are two unit supply sellers s_1, s_2 with preference constraints shown in Figure 1. For any mechanism satisfying [Eff], [B-IC] and [ERS], the allocation and payments charged to the buyers must be the one of the VCG mechanism. So, the mechanism sells both items to buyer b_1 at total price 1.

Since $A_1 = A_2 = \{b_1\}$, ERS imposes to share the revenue equally between s_1 and s_2. This way to partition of the revenue can be hardly called fair, since the 1 dollar in revenue is caused by the competition with buyer b_2 that is brought to the market by seller s_2. So a natural intuition is that a 'fair' scheme should attribute the 1 dollar in revenue to seller 2.

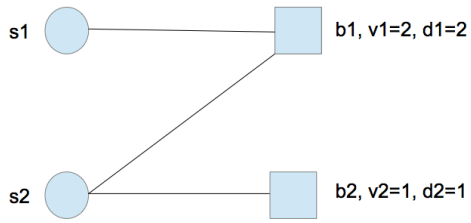


Figure 1 The two sellers receive different revenue.

The above example implies the need to refine the envy-freeness property to incorporate some notion of *which buyers are responsible for putting the price pressure*. To get a handle on such a notion, we consider ascending auctions to refine our envy-freeness property.

3.5 Fairness via ascending auction

The traditional definition of the VCG mechanism is that it allocates according to an efficient allocation and charges each agent according to the externality it imposes on other agents. One problem with this way of defining VCG is that it returns a bundle of items to each agent and a total price but does not specify how much of the payment is attributed to each item. An alternative way to define VCG is via an ascending auction [2, 1], in which there is a price clock p that gradually ascends, and as the price increases items are allocated to buyers. The total payment of the buyer in such a case is the sum over the prices of all individual items, where the price of each item is the value of the price clock when the buyer acquired the item.

The ascending auction description of VCG returns for each buyer b_j his allocation, say $x_j \in \mathbb{Z}_+$ together with x_j prices $p_1 \leq p_2 \leq \ldots \leq p_{x_j}$ corresponding to the value of the price clock when he acquires each of those items. In other words, we can describe the outcome of VCG as an ascending auction as a set of n *buying events*, where $n = \sum_i \ell_i$ and each buying event is a pair (b_j, p) indicating that one item was sold to buyer b_j at price p. Note that we assume all items are sold by VCG, which is always the case when we consider proxy buyers as discussed in Section 3.1.

Let E be the set of buying events that represents the outcome of the auction. During its execution, the ascending auction maintains in each step a tentative assignment of buying events to sellers. This allows us to define a bipartite graph between sellers and buying events called the *clinching graph*. We say that a seller s_i is connected to buying event e_j if this seller is tentatively allocated to that buying event in some point of the auction execution. If the auction execution is not unique (because of ties, caused for example by two identical sellers) we consider an edge to be in the clinching graph if for some execution of the auction its corresponding buying event is connected to the corresponding seller.

We refer to [1, 10, 13, 16] and to the full version of the paper for a formal definition of the ascending auction, but we illustrate its execution for the instance in Example 1. The price clock starts at zero, and at that price the auction is already able to allocate the item in s_1 to buyer b_1, since he faces no competition on that item. This generates a buying event $(b_1, 0)$ that is associated with seller s_1. For prices between 0 and 1, both buyers compete for the remaining item. When the price clock reaches 1, buyer b_2 is no longer interested in the remaining item and buyer b_1 can acquire it at price 1, generating a buying event $(b_1, 1)$, which is associated with seller s_2. There are no ties, so this is the unique execution of the auction, which generates the clinching graph represented in Figure 2.

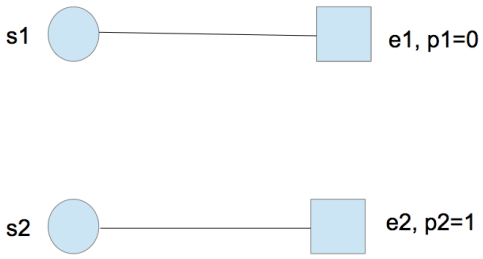


Figure 2 Each seller is linked in the clinching graph to only one buying event.

Given the clinching graph, we are now able to define a stronger notion of envy-freeness based on it. We denote by E_i the set of buying events associated to buyer b_i and by $r_i[j]$ the revenue obtained by seller s_i from buying event $e_j \in E_i$. Let us denote the revenue of unit supply seller s_i by $r_i = \sum_{j \in E_i} r_i[j]$. We now state a new version of Envy-free Revenue Sharing, that we denote by ERSCG, as follows:

- **Envy-free Revenue Sharing from Clinching Graph [ERSCG]**: $\forall s_i, s_{i'}$, $\sum_{j \in E_i} r_i[j] \geq \sum_{j \in E_i} r_{i'}[j]$.

According to the definition of clinching graph, we obtain for the example of Figure 2 that $E_1 = \{e_1\}$ and $E_2 = \{e_2\}$. The revenue sharing scheme that attributes $r_1[1] = 0$, $r_1[2] = 0$, $r_2[1] = 0$ and $r_2[2] = 1$ is therefore ERSCG. We conclude that the new definition of envy-freeness is able to characterize a fair sharing of the revenue between sellers.

We also define an approximate version of the previous property:

- **α-Envy-free Revenue Sharing from Clinching Graph[α-ERSCG]:** $\forall s_i, s_{i'}$, $\sum_{j \in E_i} r_i[j] \geq \alpha \sum_{j \in E_i} r_{i'}[j]$.

We conclude by observing that the notion of envy-freeness we introduce can easily be adapted to the original non-unit supply sellers.

4 Impossibility of Envy-freeness and Budget Balance

Before presenting two revenue sharing schemes based on the definition of clinching graph in Sections 5 and 6, we show that envy-freeness and budget balance are indeed contradicting objectives for any revenue sharing scheme based on an efficient allocation.

▶ **Theorem 2.** *There does not exist any revenue sharing efficient mechanism for the reservation exchange market which is BB and α-ERSCG for $\alpha \geq \sqrt{3} - 1 + \delta \approx 0.732$, for an arbitrary small value $\delta > 0$.*

5 Revenue sharing by the Clinching Ascending Auction.

The first revenue sharing scheme is based on the allocations computed by the clinching ascending auction (CA). We denote by CA this revenue sharing scheme.

A detailed description of the use of the clinching ascending auction [2, 1] to compute efficient VCG allocations is given in the full version of the paper. We specifically use a version for matching markets given in [13].

Crucial to the definition of revenue sharing scheme is the notion of *priority order* among sellers that is used in the execution of the CA. Whenever the CA is indifferent about buying between a set of sellers, it decides in "favor" of the seller with lower priority in the precedence

order. Intuitively, the seller of higher priority will enjoy a payment that is at least as good as the lower priority seller since the price in the ascending auction is non-decreasing. A priority order between sellers is simply represented by a permutation $\pi \in \Pi(S)$ where $\Pi(S)$ defines the set of all permutations of set S.

We set $r_i^\pi[j] = p_j$ if the execution of CA on permutation π matches e_j to s_i.

Revenue share of seller s_i from buying event e_j is defined as

$$r_i[j] = \mathbb{E}_{\pi \in \Pi(S)}[r_i^\pi[j]]. \tag{1}$$

The revenue of seller s_i is defined as $r_i = \sum_{e_j \in E_i} r_i[j]$. Since the total revenue of the mechanism $REV = \sum_{j \in E} p_j$ is shared between sellers, we state a first property of the revenue sharing scheme CA:

▶ **Claim 3.** *The revenue sharing scheme CA is BB.*

We next prove that the revenue sharing scheme CA is not exact ERSCG.

▶ **Theorem 4.** *The revenue sharing scheme CA is at most* $1/2$*-ERSCG.*

We next prove the approximate envy-freeness of revenue sharing scheme CA.

▶ **Theorem 5.** *The revenue sharing scheme CA is* $1/2$*-ERSCG.*

We conclude with the properties of buyer monotonicity and seller monotonicity for revenue sharing scheme CA.

▶ **Theorem 6.** *BM and SM hold for revenue sharing scheme CA.*

6 Revenue sharing by the Eating mechanism.

The eating mechanism is defined as a fractional process in time on the clinching graph $CG = (E \cup S, H), H = \{(e_j, s_i) : e_j \in E_i\}$. At each time $x \in [0, 1]$, the unit supply seller s_i "eats" from the the non-exhausted buying event $e_j \in E_i$ of highest payment p_j. Each seller will eat at most up to a fraction of 1. A buying event is exhausted when it has been eaten for 1 unit. The result of the eating mechanism is a fractional assignment $x_i[j] \in [0, 1]$ such that $\sum_{j \in E_i} x_i[j] \leq 1$ for each seller $s_i \in S$ and $\sum_{s_i : e_j \in E_i} x_i[j] \leq 1$ for each buying event $e_j \in E$.

Revenue share of seller s_i from buying event e_j is defined as

$$r_i[j] = x_i[j] \times p_j. \tag{2}$$

The total revenue of seller s_i is also equal to $r_i = \sum_{j \in E_i} r_i[j]$.

It is easy to observe that the revenue shares by the eating mechanism can be computed in polynomial time. We first show that revenue share mechanism EM is envy-free.

▶ **Theorem 7.** *The revenue sharing scheme EM is ERSCG.*

Proof. Consider any two sellers s_i and $s_{i'}$. At any time $x \in [0, 1]$ of execution of the eating mechanism, seller s_i eats from the non-exhausted buying event $e_j \in E_i$ of highest payment p_j. At the same time x, $s_{i'}$ eats from a buying event $e_{j'}$. Buying event $e_{j'}$ is either outside E_i or it offers payment $p_{j'} \leq p_j$. When s_i stops eating, all $e_j \in E_i$ are exhausted. We conclude that the revenue of s_i on E_i is higher than the revenue of $s_{i'}$ on E_i. ◀

It is not clear that in the EM sharing scheme all sellers eat up to 1. We show in the following that EM is not exact budget balance.

▶ **Theorem 8.** *For any* $\epsilon > 0$*, the revenue sharing scheme EM is at most* $\frac{11+\epsilon}{12+\epsilon}$*-BB.*

▶ **Theorem 9.** *The revenue sharing scheme EM is* $1 - 1/e$*-BB.*

7 Conclusions

The reservation exchange market is an emerging model for internet advertising that brings together multiple publishers and advertisers interested in trading inventories of impressions available in the future. In this work, we present the axioms and the design principles at the basis of mechanisms for reservation exchange markets. The goal we define for these markets is the design of mechanisms that are incentive compatible for buyers, envy-free for sellers, efficient and budget balance. We show that this is possible if one of the requirements of budget balance or envy-freeness is slightly relaxed. Our efficient revenue sharing mechanisms are based on the notion of clinching graph that is a convenient representation of the trades of efficient VCG allocations.

We leave several open problems in the context of reservation exchange markets. First of all, it would be interesting to close some of the gaps on approximate envy-freeness and budget balance of the revenue sharing mechanisms we propose. It is also unknown whether the eating mechanism holds some of the monotonicity properties we define in this paper. Moreover, since the clinching graph seems to provide fundamental insights for designing fair revenue sharing mechanisms, it would be helpful to derive its structure from basic properties of VCG mechanisms.

Acknowledgements. We thank Marek Adamczyk for suggesting the analysis of budget balance of the eating mechanism.

References

1 Lawrence M. Ausubel. An efficient ascending-bid auction for multiple objects. *American Economic Review*, 94(5):1452–1475, December 2004.

2 Lawrence M. Ausubel and Paul R. Milgrom. Ascending auctions with package bidding. *Frontiers of Theoretical Economics*, 1:1019–1019, 2002.

3 Sayan Bhattacharya, Vincent Conitzer, Kamesh Munagala, and Lirong Xia. Incentive compatible budget elicitation in multi-unit auctions. In *SODA*, pages 554–572, 2010.

4 Craig Boutilier, David C. Parkes, Tuomas Sandholm, and William E. Walsh. Expressive banner ad auctions and model-based online optimization for clearing. In *Proceedings of the 23rd National Conference on Artificial Intelligence – Volume 1*, AAAI'08, pages 30–37. AAAI Press, 2008.

5 Bowei Chen, Shuai Yuan, and Jun Wang. A dynamic pricing model for unifying programmatic guarantee and real-time bidding in display advertising. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*, ADKDD'14, pages 1:1–1:9, New York, NY, USA, 2014. ACM.

6 E.H. Clarke. Multipart pricing of public goods. Public Choice. *Public Choice*, pages 17–33, 1971.

7 Riccardo Colini-Baldeschi, Bart de Keijzer, Stefano Leonardi, and Stefano Turchetta. Approximately efficient double auctions with strong budget balance. In *Proceedings of the Fifteenth ACM-SIAM Symposium on Discrete Algorithms*, SODA'16. ACM, 2016. To appear.

8 Riccardo Colini-Baldeschi, Monika Henzinger, Stefano Leonardi, and Martin Starnberger. On multiple keyword sponsored search auctions with budgets. In *ICALP (2)*, pages 1–12, 2012.

9 Nikhil R. Devanur, Bach Q. Ha, and Jason D. Hartline. Prior-free auctions for budgeted agents. *CoRR*, abs/1212.5766, 2012.

10 Shahar Dobzinski, Ron Lavi, and Noam Nisan. Multi-unit auctions with budget limits. In *FOCS*, pages 260–269, 2008.

11 Paul Dütting, Inbal Talgam-Cohen, and Tim Roughgarden. Modularity and greed in double auctions. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC'14, pages 241–258. ACM, 2014.

12 Michal Feldman and John Lai. Mechanisms and impossibilities for truthful, envy-free allocations. In *Proceedings of the 5th International Conference on Algorithmic Game Theory*, SAGT'12, pages 120–131, Berlin, Heidelberg, 2012. Springer-Verlag.

13 Amos Fiat, Stefano Leonardi, Jared Saia, and Piotr Sankowski. Single valued combinatorial auctions with budgets. In *ACM Conference on Electronic Commerce*, pages 223–232, 2011.

14 D. Foley. Resource allocation and the public sector. *Yale Economic Essays*, 7:45–98, 1967.

15 D. B. Gillies. Solutions to general non-zero-sum games. In *In A. W. Tucker, R. D. Luce, Contributions to the Theory of Games IV*, Annals of Mathematics Studies 40, pages 47–85, 1953.

16 Gagan Goel, Vahab S. Mirrokni, and Renato Paes Leme. Polyhedral clinching auctions and the adwords polytope. In *STOC*, pages 107–122, 2012.

17 Gagan Goel, Vahab S. Mirrokni, and Renato Paes Leme. Clinching auctions with online supply. In *SODA*, 2013.

18 Renato Gomes and Vahab S. Mirrokni. Optimal revenue-sharing double auctions with applications to ad exchanges. In *23rd International World Wide Web Conference, WWW'14, Seoul, Republic of Korea, April 7-11, 2014*, pages 19–28, 2014.

19 F. Gul and E Stacchetti. Walrasian equilibrium with gross substitutes. *Journal of Economic Theory*, 56:95–124, 1999.

20 Kamal Jain and Christopher A. Wilkens. ebay's market intermediation problem. *CoRR*, abs/1209.5348, 2012.

21 R. Preston McAfee. A dominant strategy double auction. *Journal of Economic Theory*, 56(2):434–450, April 1992.

22 R. Myerson. Optimal auction design. *Mathematics of operations research*, 6:58–73, 1981.

23 R. Myerson and M. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economics Theory (JET)*, 29:265–281, 1983.

24 David C. Parkes and Tuomas Sandholm. Optimize-and-dispatch architecture for expressive ad auctions. In *In Proceedings of First Workshop on Sponsored Search Auctions*, 2005.

25 Mark A. Satterthwaite and Steven R. Williams. The rate of convergence to efficiency in the buyer's bid double auction as the market becomes large. *The Review of Economic Studies*, 56(4):477–498, 1989.

26 Mark A. Satterwhite and Steven R. Williams. The optimality of a simple market mechanism. *Econometrica*, 70(5):1841–1863, 2002.

27 L.S. Shapley. A value for n-person games. In *H. Kuhn and A. W. Tucker, editors. Proceedings of the 5th International Conference on Algorithmic Game Theory.*, Contributions to the theory of games, pages 120–131, 1953.

28 Hal R. Varian. Equity, envy, and efficiency. *Journal of Economic Theory*, 9:63–91, 1974.

29 W Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, pages 8–37, 1961.

30 William E. Walsh, Craig Boutilier, Tuomas Sandholm, Rob Shields, George L. Nemhauser, and David C. Parkes. Automated channel abstraction for advertising auctions. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.

Competitive Analysis of Constrained Queueing Systems

Sungjin Im[*1], **Janardhan Kulkarni**[†2], **and Kamesh Munagala**[‡3]

1 **Electrical Engineering and Computer Science, University of California, Merced, CA, USA**
sim3@ucmerced.edu

2 **Microsoft Research, Redmond, WA, USA**
jakul@microsoft.com

3 **Department of Computer Science, Duke University, Durham, NC, USA**
kamesh@cs.duke.edu

Abstract

We consider the classical problem of constrained queueing (or switched networks): There is a set of N queues to which unit sized packets arrive. The queues are interdependent, so that at any time step, only a subset of the queues can be activated. One packet from each activated queue can be transmitted, and leaves the system. The set of feasible subsets that can be activated, denoted $\mathcal{S}$, is downward closed and is known in advance. The goal is to find a scheduling policy that minimizes average delay (or flow time) of the packets. The constrained queueing problem models several practical settings including packet transmission in wireless networks and scheduling cross-bar switches.

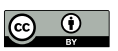
In this paper, we study this problem using the the competitive analysis: The packet arrivals can be adversarial and the scheduling policy only uses information about packets currently queued in the system. We present an online algorithm, that for any $\epsilon > 0$, has average flow time at most $O\left(\frac{R^2}{\epsilon^3}OPT + NR\right)$ when given $(1+\epsilon)$ speed, *i.e.*, the ability to schedule $(1+\epsilon)$ packets on average per time step. Here, R is the maximum number of queues that can be simultaneously scheduled, and OPT is the average flow time of the optimal policy. This asymptotic competitive ratio $O(\frac{R^2}{\epsilon^3})$ improves upon the previous $O(\frac{N}{\epsilon^2})$ which was obtained in the context of multi-dimensional scheduling [6]. In the full general model where N can be exponentially larger than R, this is an exponential improvement. The algorithm presented in this paper is based on Makespan estimates which is very different from that in [6], a variation of the Max-Weight algorithm. Further, our policy is myopic, meaning that scheduling decisions at any step are based only on the current composition of the queues. We finally show that speed augmentation is necessary to achieve any bounded competitive ratio.

1998 ACM Subject Classification C.2.1 Network Architecture and Design: Packet-switching networks, F.2.2 Nonnumerical Algorithms and Problems: Sequencing and scheduling

Keywords and phrases Online scheduling, Average flow time, Switch network, Adversarial

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.143

* S. Im was supported in part by NSF CCF-1409130.
† A part of this work was done while J. Kulkarni was at Duke University.
‡ K. Munagala was supported by NSF grants CCF-1408784, IIS-1447554, and CCF-1348696.



43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 143; pp. 143:1–143:13

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

1 Introduction

Stochastic processing networks [16] capture a wide range of resource allocation scenarios. In the general setting, there are a set of N queues, $Q_1, Q_2, \ldots, Q_N$. Packets of unit size arrive at these queues according to some arrival process. Time is discrete, and at each time step, at most one packet from each queue can be served (or scheduled), each packet requiring unit amount of service. The queues use shared resources for scheduling, leading to constraints on the subsets of queues that can be simultaneously scheduled. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ denote the set of feasible subsets of queues that can be simultaneously scheduled at any time slot; we assume the system $\mathcal{S}$ is downward closed so that if $S \in \mathcal{S}$, then any subset of S also belongs to $\mathcal{S}$. [1] For such a system let $R = \max_{S \in \mathcal{S}} |S|$ denote the maximum number of packets that can be simultaneously be scheduled.

Such a general model was first formulated in the seminal work of Tassiulas and Ephremides [16]. We first discuss some applications. First consider a $n \times n$ packet switch with crossbar architecture [11, 12, 4]. Packets arrive at each of the n input ports, with each packet specifying an output port to which it must be scheduled. The crossbar architecture enforces the constraint that at most one packet can be scheduled per input or output port per time slot. There are therefore $N = n^2$ queues $\{Q_{ij}\}$, one per input-output pair (i, j). Queue Q_{ij} queues packets arriving at input i for output j. A feasible subset of queues in $\mathcal{S}$ is a matching of input to output ports, so that $\mathcal{S}$ is the set of all matchings between inputs and outputs. A generalization of this setting arises in wireless networking, where there is a queue Q_{ij} between every pair of locations i and j that are within communication radius; a feasible subset of queues is any set of pairs (i, j) that can simultaneously communicate without interference.

A similar problem arises in *multicast* switch scheduling [14], where packets arriving at an input port of a crossbar switch need to be simultaneously transmitted to multiple output ports.

We now present the formal model. Packets arrive into the queues according to an adversarially chosen process during a *finite* time interval $[0, T]$. We do not constrain the number of packets that can arrive into any queue at any time step. We assume each packet suffers a delay of at least one time slot. The main objective considered in this setting is the average flow time. Let n_t denote the number of packets awaiting service at time t; this is the queue size at time t. Let n denote the total (finite) number of packets arriving in the system. Let T' be a sufficiently large time, for example, $T' = T + n$, by which any 'non-idle' algorithm can complete all packets.

$$\text{Average Flow Time } = \frac{1}{n} \sum_{t=0}^{T'} n_t \,. \tag{1}$$

A scheduling policy is an algorithm that decides (at every time slot) the set of queues to schedule. In an *online* policy, this decision is based on the number of packets in each queue, but not on the knowledge of future arrivals. Policies whose decisions are based only on queue sizes (or current system state) are termed *Markovian* in queueing theory. In this paper, we will call such policies *myopic*. Our focus will be on designing myopic policies.

Integral vs. Fractional Schedules. We make a distinction between *integral* and *fractional* schedules. The definition above assumes time is slotted into unit length slots, and we execute

[1] Such a set system is often called an independence system, and a set $S \in \mathcal{S}$ is said to be independent.

one schedule $\sigma \in \mathcal{S}$ per time slot. Using the notation in [15], let

$$\langle \mathcal{S} \rangle = \left\{ \sum_{\sigma \in \mathcal{S}} \alpha_\sigma \sigma \;\; | \;\; \sum_{\sigma \in \mathcal{S}} \alpha_\sigma \leq 1, \;\; \alpha_\sigma \geq 0 \;\; \forall \sigma \in \mathcal{S} \right\} \tag{2}$$

For each $S \in \mathcal{S}$, we let $\sigma(S)$ denote a binary vector in $2^{[N]}$ that encodes the subset of queues activated in S, and for notational convenience, we let $\sigma \in \mathcal{S}$, omitting S. The set $\langle \mathcal{S} \rangle$ therefore represents collections of schedules, each executed to a fraction, so that the total fraction is one unit. A fractional schedule executes one schedule from $\langle \mathcal{S} \rangle$ per time slot. In other words, a fractional schedule assumes time is continuous, and that packets are divisible. When α amount of schedule σ is executed, α amount of the corresponding packets leave the system. Note that an integral schedule is a special case of a fractional schedule where $\alpha_\sigma \in \{0, 1\}$ for all $\sigma \in \mathcal{S}$.

In fractional schedule, an alternative but equivalent definition of average flow time will be useful: we assume that packets are processed in first-in-first-out manner in each queue, and a packet j's completion time C_j is the earliest time when the whole packet leaves the queue. An individual packet's flow time is defined as its completion time C_j minus its arrival time at the queue, and the average flow time is the total flow time of all packets divided by the total number of packets. It is easy to see this definition of average flow time coincides with the above definition (1).

Let OPT denote the average flow time of the fractional policy that makes optimal scheduling decisions with knowledge of all future arrivals. This is a valid relaxation since allowing the optimal scheduler/the adversary to be fractional can only decrease its average flow time. Our goal is to design an online *integral* scheduling policy whose average flow time, for all input sequences, is at most $c \times OPT$. We call c as the *competitive ratio* of the policy. We will achieve this by first developing a fractional online schedule and then converting it into an integral schedule online.

1.1 Resource Augmentation Analysis and Scalable Policies

A simple example shows that no online algorithm can have bounded competitive ratio. To see this, consider a simple 2×2 crossbar switch with two input ports and two output ports. For some large L, at time $t = 0$, L packets arrive at both inputs destined for output 1. From time $t = L$ to $t = 2L - 1$, where one packet arrives per time step destined for output 2; the algorithm is not told which input port these packets arrive at. (Suppose these arrive at input 1.) Beyond this time, there is one packet per time slot arriving at input 1 for output 1 and input 2 for output 2. The optimal algorithm which knows the future, spends the first L steps serving the L packets queued at input 1. From time $t = L$ to $t = 2L - 1$, it schedules the matching of input port 1 to output 2 and input port 2 to output 1. Beyond this time, it has no queued packets, and can schedule each incoming packet in the same time slot. In the limit as $T \to \infty$, $OPT = 1$. Any online algorithm has to guess the behavior of OPT between time $t = L$ and $t = 2L - 1$. Suppose it keeps $x \leq L$ packets on input port 1 at time $t = L$, then it has x packets queued for all time $t \geq L$, leading to average flow time of x. This shows an unbounded competitive ratio even for randomized strategies.

The above example is typical of several scheduling problems when the input is adversarial [8, 7, 1], and motivates the need for *resource augmentation analysis*, a framework introduced by Kalyanasundaram and Pruhs [8]. We say that an online algorithm is given $(1 + \epsilon)$ speed for any $\epsilon > 0$, if in the integral case, the algorithm is allowed to perform an extra round of scheduling every $\lfloor \frac{1}{\epsilon} \rfloor$ time steps; or is allowed to execute schedules at rate

$(1+\epsilon)$ in the fractional case. Equivalently, the online algorithm, given no extra speed, is compared against the optimal scheduler with $(1-\epsilon)$-speed. This view is more natural since we simply constrain the space of feasible schedules used by the optimal policy to have $(1-\epsilon)$ rate, *i.e.*, use $\sum_{\sigma\in\mathcal{S}} \alpha_\sigma \leq 1-\epsilon$ in (2), and give a speed 1 to the online algorithm. This also highlights the fact that resource augmentation is purely an analysis tool – the algorithms we design are oblivious to such resource augmentation, using only the current queue sizes to make scheduling decisions. We call an algorithm *scalable* if for every $\epsilon > 0$, the competitive ratio of the algorithm is $f\left(N, \frac{1}{\epsilon}\right)$ for some function f.

We note that even with resource augmentation, the design of scalable scheduling policies is non-trivial. Consider the *stochastic* setting where packets arrive in each queue according to a Bernoulli process with known rate. This arguably benign setting is widely studied in networks [16, 12, 13, 15]. Consider an 3×3 crossbar switch, and the natural policy that maximizes instantaneous throughput: Consider the pairs of inputs and outputs such that there is at least one packet queued for that input/output pair. Find a maximum size matching of these input/output pairs and schedule this matching, choosing a matching at random if there are multiple maximum matchings. It is known [12] that for Bernoulli traffic where the arrival rates for all inputs and outputs are strictly smaller than one packet per time step, the expected queue size (and hence expected flow time) of this policy can be unbounded. However, for such input, there are other policies with bounded expected queue size (see below for more details). This directly shows that there exists some $\epsilon > 0$ for which the maximum throughput policy, even with $(1+\epsilon)$-speed has unbounded competitive ratio, and is hence not scalable.

1.2 Our Results

Our main result is a myopic scheduling policy for *arbitrary* scheduling constraints $\mathcal{S}$ whose asymptotic competitive ratio only depends on R. Recall that $R = \max_{S\in\mathcal{S}} |S|$ is the maximum number of packets that can be feasibly scheduled any time slot. We first develop a fractional online scheduler, and then convert it into a feasible integral policy.

▶ **Theorem 1.** *There is an online fractional policy, which we name* Sampling Independent Set from Min-Makespan (SISM)*, that is myopic, and is $O(\frac{R^2}{\epsilon^3}+N)$-competitive when compared to the optimal scheduler with $(1-\epsilon)$-speed. More precisely, the online policy has an average flow time $O(\frac{R^2}{\epsilon^3}\mathrm{OPT}+N)$ where* OPT *is the optimal algorithm's average flow time. If the online algorithm does not have to be myopic, the competitive ratio can be improved to $O(\frac{R^2}{\epsilon^4})$.*

We note that if OPT is large, then the competitive ratio depends only on $R = \max_{S\in\mathcal{S}} |S|$, which could be much smaller than N, and is a more robust measure of the complexity of the scheduling constraints $\mathcal{S}$. On the other hand, N can be made larger by simply splitting one queue into several "virtual queues". (A related notion to R, called *rank*, is used in [15] as a measure of the complexity of $\mathcal{S}$.) In fact, N can be exponentially larger than R. As mentioned before, the best known competitive ratio prior to this work was $O(\frac{N}{\epsilon^2})$ [6] which was found in the context of multi-dimensional scheduling. Further, our algorithm is very different from [6] which can't avoid a linear dependency on N; this previous work will be discussed in detail in Section 1.4.

Fractional Scheduling Policy. Our scheduling policy SISM is quite natural. At time step t, let n_{it} denote the (possibly fractional) number of packets queued at queue Q_i. At time t,

assuming no more packets arrive in the future, solve the makespan minimization problem: Find a fractional cover $\{\lambda(S)\}_{S\in\mathcal{S}}$ such that if these schedules are executed (in arbitrary order), then all queues Q_i are emptied, i.e. $\sum_{S:i\in S}\lambda(S) = n_{it}$. We process each independence set S at a rate in proportion to its weight $\{\lambda(S)\}$. An equivalent view which explains the algorithm's name is that each cover (independent set) S exists in $\lambda(S)$ copies, and we sample a cover uniformly, and process it by an infinitesimal amount.

We first show that our algorithm is $O(R^2/\epsilon^3)$-competitive for the fractional average flow time. Roughly speaking, the algorithm incurs a cost equal to the total remaining size of packets for the fractional flow – the discrepancy between the total remaining size and the total number of packets is at most N, hence the N appears in the competitive ratio. Alternatively, by using a standard method of converting fractional flow to integral flow, we can get a competitive ratio with no dependency on N, but the resulting algorithm is no longer myopic.

We note that the SISM has running time poly($|\mathcal{S}|$). However, for several applications, $|\mathcal{S}|$ itself may be exponential in N, the number of queues, leading to an exponential time algorithm. This dependence is unavoidable given the generality of the problem statement: For myopic policies that can not store past scheduling decisions, it is easy to check that maximizing instantaneous throughput encodes computing independent set of a general graph, a problem whose solution cannot be approximated to any constant factor unless $P = NP$. Assuming $P \neq NP$, this implies there is no myopic policy with poly(N) computation per time slot, which has bounded competitive ratio even with constant speed. Nevertheless, for the case of scheduling crossbar switches, our policy has poly(N) computation time per step using the Birkhoff-von Neumann theorem (see for instance [13]). Furthermore, as mentioned above, for settings such as scheduling jobs in data centers where the number of resources in contention is constant, we can assume $\mathcal{S}$ to have constant size.

Integral Policy. As mentioned above, we convert our fractional policy SISM into an integral scheduler on the fly using an emulation technique from [15]. For comparison, best previous policies [4] required speed of at least 2, and were specific to crossbar switches.

▶ **Theorem 2.** *There is an online integral policy that is myopic, and is $O(\frac{R^2}{\epsilon^3} + NR)$-competitive when compared to the optimal scheduler with $(1-\epsilon)$-speed. More precisely, the online policy has an average flow time $O(\frac{R^2}{\epsilon^3}\mathrm{OPT} + NR)$ where OPT is the optimal scheduler's average flow time.*

1.3 Technical Contributions

Our analysis proceeds via establishing amortized local competitiveness [7], a framework first introduced in [2]. Potential functions are often very useful when local competitiveness analysis fails. As one can deduce from (1), we can perform a strictly local competitiveness analysis if we can upper bound the number of packets alive in SISM's queues at any time by the analogous number for the optimal scheduler. However, this approach fails since the optimal scheduler can cleverly group packets and complete them more quickly than our algorithm. Intuitively, in the presence of an arbitrary independence set system $\mathcal{S}$, the number of queued packets can change very dynamically. In such settings, potential functions allow us to compare the online algorithm to the optimal scheduler more robustly over time.

As illustrated in [7], the standard approach to designing a potential function is to establish a rough estimate of the algorithm's flow time assuming that no more packets arrive. We use the minimum makespan needed to complete all queued packets as our estimate. A standard

way of converting this quantity into a potential function is to now replace each packet's remaining size with the *lag* that measures how much the algorithm is behind the optimal scheduler in processing that packet. This conversion makes the potential resilient to discrete events such as new packets arriving, and allows us to focus on how the potential changes as packets are continuously processed. However, the analysis is still complicated since SISM's schedule can be very different from OPT's schedule, which means the algorithm's processing may not change the potential in the right direction when needed.

As mentioned above, the potential we use is based on the minimum makespan when each packet's remaining size replaced with its lag, which can be very different from the minimum makespan schedule constructed by SISM. We therefore need to relate these two schedules, which is not at all obvious. It becomes crucial that SISM sample an independent set uniformly at random from its min-makespan schedule. This ensures two nice properties. First, it decreases the makespan of SISM uniformly with time. More importantly, due to the uniform sampling nature, it processes packets in proportion to their respective remaining sizes. Using these two properties, we show that if the optimal scheduler has very few packets left (and these are exactly the times when the algorithm needs the potential), the min-makepsan schedule for the packet lags (*i.e.*, the potential) mostly consists of independent sets where all packet's size decrease almost uniformly. Hence the potential decreases in the desired direction.

We finally note that we do not know how to use popular linear programming approaches, such as dual fitting [5] for this problem. We do note that it is relatively straightforward to obtain a $O(1)$-competitive algorithm with R-speed using these methods, and the hard part is to obtain a scalable algorithm.

1.4 Related Work

Constrained queueing systems are among the most widely studied settings in scheduling theory. Existing work falls in the realm of queueing theory, and has mostly focused on the stochastic case where packet arrivals are *i.i.d.* according to some stochastic process (most commonly Bernoulli or Poisson). In this context, the key assumption is that the arrival rates into the queues are feasible (in a certain natural sense), and the focus is on designing policies that are *stable*, meaning that the expected flow time is finite. The seminal result of [16] shows that the *maximum weight* policy is stable: This policy defines the weight of a queue as the number of packets waiting at the queue, and at each time step, finds a feasible schedule in $\mathcal{S}$ such that the sum of the weights of the queues in the schedule is maximized. When specialized to a $n \times n$ crossbar switch, this policy finds a maximum weight matching between inputs and outputs [12], where the weight of an input/output pair is the number of packets queued at that input for that output. In this context, for Bernoulli arrivals, this policy has expected queue size (or average flow time) $O(n^2)$ [15].

Most theoretical work on constrained queueing has focused on improving delay bounds in the stochastic setting. The work of [13] constructs a policy with average flow time $O(n \log n)$ for an $n \times n$ crossbar switch. Their algorithm considers a batch of L packets per input and output, finds a makespan minimizing schedule over L time steps for this batch, and runs this schedule for the next L steps. Such a policy is clearly not myopic since the policy is computed in batches; in addition, they set L carefully based on the input arrival rates, making the policy specific to the stochastic setting. Our policy Min-Makespan can be thought of as an online, myopic analog of this policy. Finally, the work of [15] presents a policy with expected flow time $O(n)$, which can be generalized to arbitrary constraints $\mathcal{S}$; their policy is based on modeling the constrained system $\mathcal{S}$ as a network of queues, and using

a proportional fairness type queueing policy called Store and Forward Allocation (SFA) [3, 10] on this queueing network. The SFA policy is a fractional scheduling policy that is specific to stochastic arrivals, and we need new ideas for the adversarial setting. The main contribution of [15] is an emulation of fractional scheduling policies by integral policies, and we use this same technique for converting our fractional schedules to integral ones.

We emphasize that the hard part of our problem is to obtain a *scalable* algorithm, *i.e.*, algorithms that have bounded competitive ratio for any ϵ extra speed, in addition to being myopic. It is relatively simple to obtain competitive algorithms that use speed that depends on $\mathcal{S}$. In particular, for scheduling a crossbar switch, any greedy maximal matching algorithm is 1-competitive with speed 2. In fact, in this same setting, the work of Chuang *et al.* [4] shows something far stronger: With speedup 2, suitably designed stable marriage algorithms are 1-competitive on *any* QoS property of delays (such as weighted flow time, l_k-norms of flow time, etc). However, these algorithms are specific to switch scheduling; furthermore, they have unbounded competitive ratio with speed less than 2, and are hence not scalable.

The constrained queueing problem (fractional version) is a special case of the polytope scheduling problem (PSP) formulated in [5]. The authors show that the proportional fairness (PF) algorithm [9] can be adapted to derive a competitive scheduling algorithm for this general setting. The objective considered is the sum of completion times of the jobs. Using the KKT conditions combined with dual fitting, the authors show that the PF algorithm is constant competitive on this objective even when jobs have arbitrary lengths and weights. The flow time objective we consider corresponds to the difference between completion time and release date of a job, and is typically a more difficult objective to optimize even in simpler settings.

In [6], PSP was reformulated with queues. In PSP-Q, the polytope is defined over queues. This constrains how much each queue can be processed. Then, any two jobs arriving into the same queue are interchangeable in the sense that one job can be replaced with the other job, preserving how much the queue is processed. In [6], it was shown that the Normalized Max-Weight algorithm is $(1+\epsilon)$-speed $O(N/\epsilon^2)$-competitive – roughly speaking, the maximum weight independence set is scheduled assuming that each queue's weight is equal to the number of jobs in the queue. The work in [6] can handle weighted jobs, but the linear dependency on N was unavoidable due to the nature of the algorithm. On the other hand, this paper can only handle packets of the same size, but the competitive ratio only depends on R.

2 Fractional Scheduling Policy: SISM

In this section, we present our fractional scheduling policy Sampling Independent Set from Min-Makespan (SISM), and analyze its performance for the average flow time objective. For convenience, we abuse the notation slightly to let $Q_i(t)$ denote the (possibly fractional) number of packets (or workload) waiting at queue Q_i. See Figure 1 for the description of the algorithm.

To compute a desired fractional cover, we can simply solve a linear programming over the variables $\{\lambda(S)\}_{S\in\mathcal{S}}$. Then, the running time will be a polynomial in $|\mathcal{S}|$. Note that this computation needs to be done only when new packets arrive since otherwise the optimal cover remains the same. As mentioned earlier, for the special case of switch scheduling, such an optimal cover can be computed in polynomial time in N by using the Birkhoff-von Neumann theorem; we can represent the current workload as a square matrix; and obtain a doubly stochastic matrix by adding dummy quantities so that the entries in each row and

Sampling Independent Set from Min-Makespan (SISM)
At each time t,
Compute a fractional cover $\{\lambda(S)\}_{S\in\mathcal{S}}$ that completes the workload $\{Q_i(t)\}_{i\in[N]}$ with the minimum makespan $L_Q(t)$.
(i.e., $L_Q(t) = \min \sum_{S\in\mathcal{S}} \lambda(S)$ s.t. $\sum_{S\in\mathcal{S}:i\in S} \lambda(S) = Q_i(t)$).
Schedule each independent set S at a rate of $\lambda(S)/L_Q(t)$.

Figure 1 The SISM Procedure.

column add up to $\max_i Q_i(t)$ and normalizing the matrix; and decompose it into permutation matrices.

Our scheduling policy has the following nice property.

▶ **Proposition 3.** *The policy* SISM *decreases each $Q_i(t)$ at a rate of $\frac{Q_i(t)}{L_Q(t)}$ at all times when no packets arrive or are completed by* SISM *or the optimal scheduler.*

The proof easily follows by viewing the policy as sampling an independence set uniformly from a multi-set of independent sets where each independence set S exists in $\lambda(S)$ copies, and observing each queue Q_i appears in exactly $Q_i(t)$ independent sets in the multi-set.

2.1 Potential Function

Throughout the analysis, we assume that the optimal scheduler (OPT) is restricted to $(1-10\epsilon)$-speed for $\epsilon \le 1/10$. Our analysis is based on a potential function. To formally define the potential function, we need more notation. Let $Q_i^*(t)$ denote the workload waiting at Q_i at time t in the optimal schedule. Define $Z_i(t) := \max\{Q_i(t) - Q_i^*(t), 0\}$ to be the algorithm's lag on queue Q_i compared to the optimal scheduler. Define $L_Z(t)$ to be the minimum makespan achievable assuming that each queue Q_i has $Z_i(t)$ workload. More precisely,

$$L_Z(t) := \min_{\lambda_Z} \sum_{S\in\mathcal{S}} \lambda_Z(S) \quad \text{s.t.} \quad \sum_{S\in\mathcal{S}:i\in S} \lambda_Z(S) = Z_i(t) \quad \forall i \in [N] \tag{3}$$

The potential function $\Phi(t)$ is defined as follows.

$$\Phi(t) = \frac{R}{\epsilon}(L_Z(t))^2 \tag{4}$$

2.2 High Level Idea

We first give a high-level overview of the analysis. For notational convenience, let time ∞ refer to a sufficiently large time step by which all packets are completed by our policy and the optimal scheduler. This is well-defined since the total number of packets arriving into queues is finite. Our final goal is to show

$$\int_0^\infty V(t)dt \le \frac{2R^2}{\epsilon^3} \int_0^\infty V^*(t)dt, \tag{5}$$

where $V(t) := \sum_i Q_i(t)$ denotes the total workload waiting in our algorithm's queues. Likewise, $V^*(t)$ is analogously defined for the optimal scheduler. The left-hand-side quantity is the algorithm's total fractional flow time, and the right-hand-side quantity is the optimal scheduler's total fractional flow time, which is at most the optimal scheduler's total integral

flow time. Once we establish the bound (5), using the fact that the number of packets alive at each time t is at most $V(t) + N$, we will be able to complete the analysis.

To show (5), it suffices to show that the following standard conditions are satisfied.

1. Boundary condition: $\Phi(0) = \Phi(\infty) = 0$.
2. Discontinue changes: $\Phi(t)$ does not change when a packet arrives, or is completed by the online algorithm or the optimal scheduler.
3. Continuous changes: For every time t when no job arrives or completes, $V(t) + \frac{d}{dt}\Phi(t) \leq \frac{2R^2}{\epsilon^3} V^*(t)$.

The first condition is easy to check since the potential is clearly 0 when no jobs are waiting in the algorithm's or in the optimal scheduler's queues. When a new packet arrives into Q_i, $Q_i(t)$ and $Q_i^*(t)$ both increase by 1, hence $Z_i(t)$ and $L_Z(t)$ remain the same. Completion of packets does not affect the potential since $Z_i(t)$ and $L_Z(t)$ are defined in the continuous time domain. Hence the second condition follows. Note that there are only a finite number of time steps when discontinuous changes occur.

2.3 Competitive Analysis of Queue Size

In view of the above discussion, we will focus on proving the third condition concerning continuous changes. Throughout this section, we will consider an infinitesimal time interval $[t, t + \mathrm{dt}]$ during which no discontinuous changes occur. We will consider two cases. In the first case where the algorithm has a workload comparable to that of the optimal solution, we charge the algorithm's workload plus the possible potential increase to the optimal solution's workload. In the other case, the algorithm's workload in each queue Q_i is very similar to $Z_i(t)$ which is always non-negative and measures how much the algorithm is behind the optimal solution in terms of the workload in queue Q_i. This is the case where the potential helps the algorithm in need. In this case, the algorithm will effectively decrease the makespan $L_Z(t)$ based on lags $\{Z_i(t)\}$ in the potential function, which will cancel off the potential increase due to the optimal solution's processing and give enough credits to pay for the algorithm's workload due to the extra speed the algorithm is given.

We begin with a couple of definitions.

▶ **Definition 4.** A queue Q_i is said to be tight at time t if $Z_i(t) \geq (1 - \epsilon)Q_i(t)$, otherwise loose. An independent set $S \in \mathcal{S}$ is said to be tight at time t if *all* queues in S are tight. Otherwise, S is said to be loose.

The following simple observation will be used throughout the analysis.

▶ **Proposition 5.** $L_Q(t) \leq V(t) \leq R \cdot L_Q(t)$.

To study the continuous changes of $\Phi(t)$, we will first take a close look at the optimal scheduler's effect on $\Phi(t)$, freezing the algorithm's effect. Let $\frac{d}{dt}\Phi(t)\big|_{\mathrm{OPT}}$ denote the continuous change of $\Phi(t)$ due to the optimal scheduler's processing.

▶ **Lemma 6.** $\frac{d}{dt}\Phi(t)\big|_{\mathrm{OPT}} \leq 2(1 - 10\epsilon)\frac{R}{\epsilon} L_Z(t)$.

Proof. Fix a time t and consider an infinitesimal time interval $[t_1 = t, t_2]$ where *no discontinuous changes* occur. Let $L_{Z,1}$ denote the minimum makespan for queues with workload $\{Z_i(t_1)\}_{i\in[N]}$. Let $L'_{Z,2}$ the minimum makespan for queues with workload $\{Z'_i(t_2)\}$ where $Z'_i(t_2) := \max\{Q_i(t_1) - Q_i^*(t_2), 0\} \leq \max\{Q_i(t_1) - Q_i^*(t_1), 0\} + (Q_i^*(t_1) - Q_i^*(t_2))$. Note that $(Q_i^*(t_1) - Q_i^*(t_2))$ is non-negative, and refers to the amount of work that the optimal scheduler

does for queue Q_i during $[t_1, t_2]$. Observe that $L'_{Z,2} \le L_{Z,1} + (1-10\epsilon)(t_2 - t_1)$. This is because one can empty all queues with sizes $Z_i(t_1) + (Q^*_i(t_1) - Q^*_i(t_2))$ by following the schedule that achieves the makespan $L_{Z,1}$ for workload $\{Z_i(t_1)\}$, and the optimal schedule during $[t_1, t_2]$ with 1-speed; recall that the optimal scheduler has $(1-10\epsilon)$-speed. Such a schedule only does more work than the required workload, $\{Z'_i(t_2)\}_{i\in[N]}$. Hence due to downward closedness of $\mathcal{S}$, we have $\frac{d}{dt}L_Z(t)\big|_{\mathrm{OPT}} \le 1-10\epsilon$, which completes the proof. ◀

We now study the more interesting case of continuous changes of $\Phi(t)$ due to the algorithm's processing freezing the optimal scheduler's processing. We consider two cases depending on the magnitude of the volume of loose queues. Let $V_{loose}(t)$ denote the total sum of $Q_i(t)$ over all loose queues Q_i. $V_{tight}(t)$ is similarly defined for tight queues.

Case (i): $V_{loose}(t) \ge (\epsilon/R)V(t)$, i.e. $V_{tight}(t) \le (1-\epsilon/R)V(t)$. By definition, for any loose queue Q_i, we have $Z_i(t) := \max\{Q_i(t) - Q^*_i(t), 0\} \le (1-\epsilon)Q_i(t)$, hence $Q^*_i(t) \ge \epsilon Q_i(t)$. Thus $V^*(t) \ge \sum_{i:loose} Q^*_i(t) \ge \sum_{i:loose} \epsilon Q_i(t) = (\epsilon^2/R)V(t)$. We directly charge $V(t)$ to $V^*(t)$. We also charge $\frac{d}{dt}\Phi(t)\big|_{\mathrm{OPT}}$ to $V^*(t)$. Towards this end, we use Lemma 6 together with the following simple observation which immediately follows from $Z_i(t) \le Q_i(t)$ for all i, and downward closedness of $\mathcal{S}$.

▶ **Proposition 7.** *$L_Z(t) \le L_Q(t)$.*

Hence $\frac{d}{dt}\Phi(t)\big|_{\mathrm{OPT}} \le \frac{2R}{\epsilon}L_Z(t) \le \frac{2R}{\epsilon}L_Q(t) \le \frac{2R}{\epsilon}V(t) \le \frac{2R^2}{\epsilon^3}V^*(t)$.

The algorithm's processing can only decrease $\Phi(t)$, i.e. $\frac{d}{dt}\Phi(t)\big|_{algo} \le 0$, but we do not need it in this case; in the other case, we will need a stronger bound which is stated in Lemma 10. Combining these, the desired third condition easily follows.

Case (ii): Otherwise. This is the more interesting case where the potential plays a crucial role. We begin with showing the following lemma. Intuitively, since $Z_i(t)$ is very close to $Q_i(t)$ over most queues, the makespan for workload $\{Z_i(t)\}$ should be almost as large as the makespan for workload $\{Q_i(t)\}$.

▶ **Lemma 8.** *$L_Z(t) \ge (1-3\epsilon)L_Q(t)$.*

Proof. It is easy to see that the total weight of tight independent sets in $\{\lambda_Q(S)\}$ is at least $L_Q(t) - \frac{\epsilon}{R}V(t) \ge (1-\epsilon)L_Q(t)$, since one unit of loose workload in a queue can make at most one unit of independent sets loose. In other words, among independent sets of total weight $L_Q(t)$, tight independent sets have total weight at least $(1-\epsilon)L_Q(t)$, which lower bounds how much tight independent sets contribute to the makespan. Now for the sake of contradiction, suppose that $L_Z(t) < (1-3\epsilon)L_Q(t)$. This implies that there is a way of scheduling tight queues Q_i with workload $Z_i(t) \ge (1-\epsilon)Q_i(t)$ within $(1-3\epsilon)L_Q(t)$ time steps, so all tight queues Q_i with workload $Q_i(t)$ within $(1-3\epsilon)L_Q(t)/(1-\epsilon) < (1-\epsilon)L_Q(t)$ time steps. Hence we can complete all workload appearing in the tight independent sets more quickly than $(1-\epsilon)L_Q(t)$ time steps, which is a contraction to the minimality of $L_Q(t)$. ◀

For notational convenience, let $\{\lambda_Z(S)\}$ be an optimal fractional cover that achieves the minimum makespan for workload $\{Z_i(t)\}$ as illustrated in (3). We now take a close look at $\{\lambda_Z(S)\}$ focusing on tight independent sets.

▶ **Lemma 9.** *The total weight of tight independent sets in $\{\lambda_Z(S)\}$ is at least $(1-3\epsilon)L_Z(t)$.*

Proof. Recall that the total workload of loose queues is at most $(\epsilon/R)V(t) \le \epsilon L_Q(t)$. The total weight of tight independent sets in $\{\lambda_Z(S)\}$ is then at least $L_Z(t) - \epsilon L_Q(t) \ge L_Z(t) - \epsilon L_Z(t)/(1-3\epsilon) \ge (1-3\epsilon)L_Z(t)$. The first inequality is due to Lemma 8. ◀

Let $y_i(t)$ denote the total weight of tight independent sets in $\{\lambda_Z(S)\}$ that contain queue Q_i. Since our goal is to lower bound the potential's decrease due to A's processing, we can assume that Q_i is processed at a rate of $\frac{y_i(t)}{L_Q(t)} \leq \frac{Z_i(t)}{L_Q(t)} \leq \frac{Q_i(t)}{L_Q(t)}$ due to Proposition 3 and downward closeness of $\mathcal{S}$. For each tight independent set S, imagine that we process queue Q_i in S at a rate of $\frac{\lambda_Z(S)}{L_Q(t)}$ – here note that each Q_i is processed exactly at a rate of $\frac{y_i(t)}{L_Q(t)}$ in total. Hence the weight of tight indepedent set S decreases at a rate of $\frac{\lambda_Z(S)}{L_Q(t)}$, and the total weight of tight independent sets decreases at a rate of $\frac{(1-3\epsilon)L_Z(t)}{L_Q(t)} \geq (1-3\epsilon)^2 \geq 1-6\epsilon$ by summing over all tight independent sets, and subsequently by applying Lemma 9 and 8. Hence we derive the following lemma.

▶ **Lemma 10.** $\frac{d}{dt}L_Z(t)\Big|_{algo} \leq -(1-6\epsilon)$, *and* $\frac{d}{dt}\Phi(t)\Big|_{algo} \leq -2\frac{R}{\epsilon}(1-6\epsilon)L_Z(t)$.

We are now ready to complete the analysis. By Lemma 6, 10, 8, we have $\frac{d}{dt}\Phi(t) = -8R \cdot L_Z(t) \leq -4R \cdot L_Q(t) \leq -4V(t)$. In either of the two cases, we have shown the third condition on $\Phi(t)$ concerning the continuous changes.

2.4 From Queue Size to Flow Time

So far we have shown (5). By processing packets in each queue in first-in-first-out order, we have that $0 \leq A_i(t) - Q_i(t) < 1$ for all i, where $A_i(t)$ denotes the number of packets in queue Q_i at time t. Let $A(t) := \sum_i A_i(t)$. Let $\mathcal{T}$ be the collection of maximal intervals such that $V(t) > 0$ at all times t during T, for each time interval $T \in \mathcal{T}$. Note that $A(t) = 0$ if $V(t) = 0$. Hence the algorithm's total flow time is

$$\sum_{T\in\mathcal{T}}\sum_{t\in T} A(t) \leq \sum_{T\in\mathcal{T}}\sum_{t\in T}\int_{\tau=t}^{t+1}\Big(V(t)+N\Big)d\tau \leq O(\frac{R^2}{\epsilon^3})\int_0^\infty V^*(t)dt + N\sum_{T\in\mathcal{T}}|T|$$

Since our algorithm processes at least one unit of workload at each time if it exists, it is easy to observe that $\sum_{T\in\mathcal{T}}|T|$ is at least $\Omega(1)$ times the total number of packets. Knowing that each packet has flow time at least one, we can upper bound $N\sum_{T\in\mathcal{T}}|T|$ by N times the total number of packets. By dividing both sides of the inequality by the total number of packets, we derive the first part of Theorem 1.

The second part follows by a standard method of converting an algorithm good for fractional flow time into one for integral flow time. In fact, the quantity $\int_0^\infty V(t)dt$ in Equation (5) is the algorithm's total fractional flow time, where each packet incurs a penalty equal to its remaining size. In general, one can translate an online algorithm that is c-competitive with s speed for the average fractional flow time objective online into one that is $O(c/\epsilon)$-competitive with $s(1+\epsilon)$-speed for the average integral flow objective, for any $0 \leq \epsilon < 1$. For example, see [7]. However, after this conversion, the algorithm is no longer myopic.

3 Emulation by Integral Schedules

Once we have a fractional algorithm we can convert it into an integral algorithm with a small loss in the competitive algorithm. We remind the reader that an integral algorithm must schedule exactly one independent set S out of $\mathcal{S}$ at each time t. In contrast, a fractional algorithm schedules an independent set S for any infinitesimal time step dt, and reduces each queue in S by dt. Shah et al. [15] show that given an fractional schedule B, one can obtain an integral schedule on the fly that has an $O(RN)$ upper bound on the algorithm's total lag at each time as opposed to the optimal schedule.

Let $Q_i^B(t)$ be the number of packets waiting in queue Q_i at time t in the integral algorithm B's schedule. The quantity $Q_i^A(t)$ is analogously defined for a given fractional online algorithm A – however, $Q_i^A(t)$ does not have to be integral at an integer time point. Define B's lag as opposed to A as $\Delta_i(t) := \max\{Q_i^B(t) - Q_i^A(t), 0\}$. The integral algorithm B has the following description at any time t.

- Compute a (fractional) min-makespan schedule on $\{\Delta_i(t), i = 1, 2, \ldots, N\}$.
- In the above solution, if some independent set is scheduled to an amount at least 1, schedule this independent set.
- Otherwise, let $W(t) = \{i|\Delta_i(t) \geq 1\}$. Find an independent set that schedules the most number of packets from $W(t)$, and schedule this set.

The following lemma restates Lemma 5.7 and 5.8 in [15].

▶ **Lemma 11.** *For the algorithm B as described above, we have $\sum_i \Delta_i(t) \leq R(N+2)$.*

Note that the above lemma is only concerned with the difference between the queues of the two algorithms A and B. However, we can now use an idea similar to the one we used in Section 2.4. The crucial observation is that the discrepancy $\sum_i \Delta_i(t)$ can occur only at the times where the algorithm B has at least one packet at time t; the number of such times is at least the total number of packets arriving at queues. Theorem 2 easily follows from Theorem 1 and this observation.

4 Conclusions

In this paper, we studied competitive algorithms for constrained queueing systems, which have been extensively studied in stochastic queueing theory. In queueing theory, the goal is to obtain a stable algorithm when the load reaches the inherent "system threshold". A parameter is often used to measure the proximity of the system load to the threshold. For example, in the special case of switch scheduling, the expected load arriving at queues is assumed to have a makespan at most $1 - \epsilon$ for $\epsilon > 0$. The work of [15] shows an expected total queue size of $O(R/\epsilon + RN)$, and the additive term RN is ignored assuming that ϵ is arbitrarily small. In contrast, we give an $O(R^2/\epsilon^3 + RN)$-competitive algorithm for the average flow time objective when compared to the optimal scheduler with $(1 - \epsilon)$-speed. As illustrated in the seminal paper [8] that introduced the resource augmentation analysis model, a slightly weaker adversary/benchmark is also motivated by the system threshold. The goal is to design an online algorithm whose objective is comparable to the optimal offline solution with the least possible speed augmentation for all adversarial inputs. As mentioned before, such an algorithm is said to be scalable. Despite the seemingly similar goals and motivations as well as the seemingly similar analytic bounds, the relation between queueing theory (stability) and competitive analysis (scalability) remains unclear. Our work suggests the possibility of making a rigorous connection between the two concepts.

An open problem is to get tighter upper or lower bounds on the competitive ratio, even in the special case of switch scheduling. The lower bounds in [5, 6] do not hold since it requires packets having varying sizes. The only known negative resultis that the problem does not admit a (bounded) competitive algorithm without speed augmentation. However, we cannot even rule out the existence of $O(\text{poly}(1/\epsilon))$-competitive algorithms with $(1 + \epsilon)$-speed.

References

1. S. Anand, Naveen Garg, and Amit Kumar. Resource augmentation for weighted flow-time explained by dual fitting. In *SODA*, pages 1228–1241, 2012. URL: `http://dl.acm.org/citation.cfm?id=2095213`.
2. Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1):3:1–3:39, March 2007.
3. T. Bonald, L. Massoulié, A. Proutière, and J. Virtamo. A queueing analysis of max-min fairness, proportional fairness and balanced fairness. *Queueing Syst. Theory Appl.*, 53(1-2):65–84, June 2006. `doi:10.1007/s11134-006-7587-7`.
4. Shang-Tse Chuang, Ashish Goel, Nick McKeown, and Balaji Prabhakar. Matching output queueing with a combined input/output-queued switch. *IEEE Journal on Selected Areas in Communications*, 17(6):1030–1039, 1999.
5. Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. In *STOC*, 2014.
6. Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. Competitive flow time algorithms for polyhedral scheduling. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 506–524, 2015. `doi:10.1109/FOCS.2015.38`.
7. Sungjin Im, Benjamin Moseley, and Kirk Pruhs. A tutorial on amortized local competitiveness in online scheduling. *SIGACT News*, 42(2):83–97, 2011. `doi:10.1145/1998037.1998058`.
8. Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47(4):617–643, 2000.
9. F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *The Journal of the Operational Research Society*, 49(3):pp. 237–252, 1998.
10. F.P. Kelly, L. Massoulié, and N.S. Walton. Resource pooling in congested networks: proportional fairness and product form. *Queueing Systems*, 63(1-4):165–194, 2009. `doi:10.1007/s11134-009-9143-8`.
11. Nick McKeown. The islip scheduling algorithm for input-queued switches. *IEEE/ACM Transactions on Networking*, 7(2):188–201, 1999.
12. Nick McKeown, Adisak Mekkittikul, Venkat Anantharam, and Jean Walrand. Achieving 100% throughput in an input-queued switch. *IEEE Transactions on Communications*, 47(8):1260–1267, 1999.
13. Michael J. Neely, Eytan Modiano, and Yuan-Sheng Cheng. Logarithmic delay for $n \times n$ packet switches under the crossbar constraint. *IEEE/ACM Trans. Netw.*, 15(3):657–668, June 2007.
14. Balaji Prabhakar, Nick McKeown, and Ritesh Ahuja. Multicast scheduling for input-queued switches. *IEEE Journal on Selected Areas in Communications*, 15(5):855–866, 1997.
15. Devavrat Shah, Neil Walton, and Yuan Zhong. Optimal queue-size scaling in switched networks. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS'12, pages 17–28, 2012.
16. Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992.

The Linear Voting Model*

Colin Cooper[1] and Nicolas Rivera[2]

1 **Department of Informatics, King's College London, London, UK**
`colin.cooper@kcl.ac.uk`
2 **Department of Informatics, King's College London, London, UK**
`nicolas.rivera@kcl.ac.uk`

Abstract

We study voting models on graphs. In the beginning, the vertices of a given graph have some initial opinion. Over time, the opinions on the vertices change by interactions between graph neighbours. Under suitable conditions the system evolves to a state in which all vertices have the same opinion. In this work, we consider a new model of voting, called the Linear Voting Model. This model can be seen as a generalization of several models of voting, including among others, pull voting and push voting. One advantage of our model is that, even though it is very general, it has a rich structure making the analysis tractable. In particular we are able to solve the basic question about voting, the probability that certain opinion wins the poll, and furthermore, given appropriate conditions, we are able to bound the expected time until some opinion wins.

1998 ACM Subject Classification C.2.4 Distributed Systems, G.2 Discrete Mathematics, G.3 Probability and Statistics

Keywords and phrases Voter model, Interacting particles, Randomized algorithm, Probabilistic voting

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.144

1 Introduction

Graphs are very popular as a simple model of the complex environment in which individuals interact. In this paper we focus in voting models on finite graphs, in which vertices of a given graph have opinions and by interacting with their neighbours they change such opinions. Voting models can be used to mimic real-life situations such as the spread of opinions or infections in a society, the evolution of species or models of particle interaction in physics.

While many models has been proposed in the literature, we do not aim to propose a new particular model, but to unify some of the existing models in a tractable way. With this in mind, we propose a general model of voting, called the Linear Voting Model. This model subsumes several models proposed in the past, including, for example, the push model and the very popular pull model.

Even though the voter model has been widely studied in the case of infinite structures, one of the first rigorous studies on finite structures was made by Donnelly and Welsh [4]. In that work, the authors studied a continuous-time version of the pull voting model and, under the name of infection model, the push voting model. In the continuous time version, each vertex has an exponential clock and when it rings, the vertex selects a random neighbour and pulls its opinion (in the case of pull voting) or pushes its opinion on the neighbour (in the

* Research supported by EPSRC grant EP/M005038/1, "Randomized algorithms for computer networks". N. Rivera was supported by funding from Becas CHILE.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 144; pp. 144:1–144:12

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

case of push voting). On the other hand, Hassin and Peleg [6] and Nakata et al. [8] studied the discrete time version of pull voting, in which vertices do not have a clock but at each round each vertex synchronously pulls an opinion. Both papers considered the two-party model and studied its possible application to distributing computing, in particular to the agreement problem. The focus of [6] and [8] is on the probability that all vertices eventually adopt the opinion which was initially held by a given subset of vertices. They proved that the probability that opinion A wins is $d(A)/d(V)$, where $d(X)$ is the sum of the degrees of the vertices of $X \subseteq V$ and A is the set of vertices whose initial opinion was A.

The consensus time of G, i.e., the time needed for the vertices of a graph G to agree on an opinion during voting, has attracted a lot of attention, especially because a low consensus time implies a better distributed algorithm for the agreement problem. In the continuous-time setting, Oliveira [9] shows that the expected consensus time is $\mathcal{O}(H_{\max})$, where $H_{\max} = \max_{v,u \in V} H(v,u)$ and $H(v,u)$ is the hitting time of u of a random walk starting at vertex v. Furthermore, in a later work [10], Olivera proved that under certain conditions on the underlying graph G, the consensus time is concentrated around $2m(G)$, where $m(G)$ is the meeting time of two independent random walk starting in stationary distribution. It is however, not clear whether the continuous-time results apply to the discrete-time setting. Hassin and Peleg [6] using a dual process, the coalescing random walk, proved that the expected consensus time is $\mathcal{O}(m(G)\log(n))$, where $m(G)$ is the meeting time of independent discrete-time random walks, thus giving $\mathcal{O}(n^3 \log(n))$ in the worst case. By using the same approach, Cooper et al. [2] improved the previous result and proved that the consensus time is $\mathcal{O}(n/(\nu(1-\lambda_2)))$, where n is the number of vertices of G and ν is a parameter that measures the regularity of the degree sequence, ranging from 1 for regular graphs to $\Theta(n)$ for the star graph. The result of Cooper et al. achieves an upper bound of $\mathcal{O}(n^3)$ in the worst case. Berenbrink et al. [1] used a more ad hoc approach and proved that the consensus time is $\mathcal{O}((d_{\text{ave}}/d_{\min})(n/\Phi))$ where Φ is the conductance of the graph, and d_{ave}, $d_{\min}$ are the average and minimum degrees respectively.

The consensus time for the push model has not been so widely studied. Push voting is a particular class of the so-called Moran process. Díaz et al. [3] proved that the the consensus time is $\mathcal{O}(n^4 q)$ where q is the square of the sum of the inverses of the degree sequence of G, giving a consensus time of $\mathcal{O}(n^6)$ in the worst case.

1.1 Our model and results

Let $G = (V, E)$ be a graph with $|V| = n$. Define a configuration of opinions as a $n \times 1$ vector $\xi \in Q^V$, where $Q = \{0, 1\}$ for the two party model, or $Q = \{1, \ldots, n\}$ if we want to allow more parties.

Let $\mathcal{M}(V)$ be the set of all $n \times n$ matrices indexed by the elements of V, with exactly one 1 entry per row and all other elements 0. Also, define $\Pi(V)$ as the set of probability measures on $\mathcal{M}(V)$. If no confusion arises, we will just write $\mathcal{M}$ instead of $\mathcal{M}(V)$ and Π instead $\Pi(V)$.

Let $l \in \Pi$ be a distribution over matrices in $\mathcal{M}$. Given an initial configuration ξ, we define the process $(\xi_t)_{t \geq 0}$, with t running over the non-negative integers, as

$$\xi_t = \begin{cases} \xi, & \text{if } t = 0, \\ M_{t-1}\xi_{t-1}, & \text{if } t > 0, \end{cases} \tag{1}$$

where M_t are i.i.d matrices sampled from l, and $M\xi$ is the standard matrix-vector multiplication. The above process is called a **linear voting model** with parameters (l, ξ) and it

is denoted by $(\xi_t) \sim \mathcal{LVM}(l, \xi)$. Clearly, $\xi_t(v)$ represents the opinion of vertex v at round t. Consider $M \in \mathcal{M}$ and $\xi' = M\xi$, then if al vertices have different opinions, we have that $\xi'(v) = \xi(w)$ if and only if $M(v, w) = 1$. Since M has only one 1 in each row, the voting is well-defined in the sense at every round each vertex adopts the opinion of only one vertex (including itself). Examples of linear voting models include the pull voting (asynchronous or synchronous) and the push voting model.

We proceed to present our contribution. Theorem 1 of this paper gives the probability a particular opinion wins. This generalises the approach used in [6]. Theorem 2 gives an upper bound to the expected consensus time. Our technique is qualitatively different from the approach of previous authors which depended on a detailed dualisation of the voting process, indeed, we follow an approach similar to Levin et al. [7, chapter 17] or Berenbrink [1].

Let $l \in \Pi$ and define the mean matrix H of l as

$$H = H(l) = \sum_{M \in \mathcal{M}} l(M)M.$$

From Lemma 4 we have that H is the transition Matrix of a Markov Chain with state space V. We denote by μ the stationary distribution of H (if any). Define the consensus time τ_{cons} as the first time all the opinions are the same, i.e, there exists c such that $\xi_{\tau_{\text{cons}}}(v) = c$ for all $v \in V$. Observe τ_{cons} is a stopping time and that c is the final opinion of the vertices. We have the following theorem about the winning probability.

▶ **Theorem 1.** *Let $(\xi_t) \sim \mathcal{LVM}(l, \xi)$ be a linear voting model with mean matrix H with $\xi \in \{0, 1\}^V$. Assume that H has a unique stationary distribution μ and that $\tau_{\text{cons}} < \infty$, then*

$$\mathbb{P}(\textit{opinion } 1 \textit{ wins} | \xi_0 = \xi) = \sum_{v \in V} \mu(v)\xi(v).$$

The above theorem solves the winning probability problem under reasonable conditions, so we focus on the consensus time problem.

Consider the two party model and let S_t be the set of vertices with opinion 1 at the beginning of round t. Denote $\mu(S_t) = \sum_{v \in S_t} \mu(v)$, where μ is the stationary distribution of H, and $Z_t = \mu(S_{t+1}) - \mu(S_t)$. Let $\mu^* = \min_{v \in V} \mu(v)$. Define the quantity Ψ as

$$\Psi = \mu^* \min_{\substack{S \subseteq V \\ S \neq \emptyset, V}} \frac{\mathbb{E}(|Z_0||S_0 = S)}{\min\{\mu(S), 1 - \mu(S)\}}, \tag{2}$$

where the minimum is over all $S \subseteq V$ except $S = \emptyset$ and $S = V$. Using the above definitions we prove the following theorem.

▶ **Theorem 2.** *Let $(\xi_t)_{t \geq 0} \sim \mathcal{LVM}(l, \xi)$ with $\xi \in \{0, 1\}^V$ be a voting model with $\Psi > 0$ then*

$$\mathbb{E}(\tau_{\text{cons}}) \leq 64/\Psi.$$

The structure of the paper is as follows. In Section 2 we introduce the model and give some examples to gain some intuition and demonstrate the flexibility of the model. In Section 3, we introduce the necessary notation to prove Theorem 1. In Section 4 we prove Theorem 2.

Notation. $G = (V, E)$ stands for a simple graph. We assume $|V| = n$. For $v \in V$ we denote by $N(v)$ the neighbourhood of v and define $d(v) = |N(v)|$. Moreover, given $X \subseteq V$, we define $d(X)$ as the sum of the degrees of the vertices in X. We use the notation $v \sim w$ to say that v and w are adjacent vertices. Q stands for the set of possible opinions, in general

$Q = \{0, 1\}$ or $Q = \{1, \dots, n\}$. We denote by $\mathcal{M}$ the set of $n \times n$ matrices with exactly one 1 in each row and 0 in the other positions. Let Π be the set of probability distribution on $\mathcal{M}$, and $l \in \Pi$ be a given probability distribution over matrices in $\mathcal{M}$. $M^\top$ denotes the transpose of the matrix M.

2 The linear voting model.

Recall the definition of a linear voting model. Given $l \in \Pi$ and $\xi \in Q^V$ we say $(\xi_t)_{t\geq 0} \sim \mathcal{LVM}(l, \xi)$ if $\xi_0 = \xi$ and $\xi_{t+1} = M_t\xi_t$, $t \geq 0$, where the M_t are i.i.d. samples from l. The following models are examples of linear voting.

(a) Synchronous pull model. At each round each vertex samples a random neighbour and adopts the opinion of such neighbour.

(b) Asynchronous pull model. At each round one vertex v is selected at random, then it samples a random neighbour and v adopts the opinion of this neighbour.

(c) Asynchronous push model. At each round a vertex v is selected at random, then it samples a random neighbour and the neighbour adopts the opinion of v.

(d) Abusive push model. At each round one vertex v is selected at random and the whole neighbourhood adopts the opinion of v.

(e) Pull-push model. At each round one vertex v is selected at random, and two neighbours u_1, u_2 are selected randomly (with replacement). Then at the same time, u_1 adopts the opinion of v while v adopts the opinion of u_2.

▶ Remark. To be precise, the changes in the opinions happen at the end of a round t, prior to round $t + 1$. In particular if v adopts the opinion of w at round t, it means that at round $t + 1$, vertex v has the opinion of w at round t.

▶ **Lemma 3.** *The five models defined above are linear voting models.*

Proof Sketch. We just prove it for the first and second model. For the other models the proof is similar. Let ξ_t be the configuration of opinions at round t. In the synchronous pull voting at each round each vertex v samples a random neighbour $w(v)$ and then v adopts the opinion of $w(v)$. Call ξ_{t+1} the new configuration of opinion. We check that $\xi_{t+1} = M\xi_t$ where the (random) matrix M is given by $M(v, w(v)) = 1$ for all $v \in V$, and 0 for the others entries. It is straightforward to check that $M\xi_t(v) = M(v, w(v))\xi(w(v)) = \xi_t(w(v)) = \xi_{t+1}(v)$ and also that M has only one 1 in each row and thus $M \in \mathcal{M}$.

For the asynchronous pull model, observe that only one vertex v is selected and then v adopts the opinion of a random vertex $w(v)$, while all other vertices keep their opinions unchanged. Call ξ_{t+1} the new configuration. Define M as $M(v, w(v)) = 1$, $M(u, u) = 1$ for all $u \neq v$ and 0 for all other entries (M is like the identity matrix, except in the column of v). It is not hard to check that the random matrix M mimics the asynchronous pull model, i.e. $\xi_{t+1} = M\xi_t$, and that $M \in \mathcal{M}$. ◀

Remember we define the mean matrix of $l \in \Pi$ as $H = H(l) = \sum_{M\in\mathcal{M}} l(M)M$. Since most of the models are described by rules rather than by giving the explicit distribution l, it might be hard to compute $H(l)$. Nevertheless, the following lemma helps us to compute H without exhibiting l explicitly.

▶ **Lemma 4.** *For any distribution l over matrices in $\mathcal{M}$, the matrix $H = H(l)$ is the transition matrix of a Markov chain. Moreover, for every $t \geq 0$, and $v, w \in V$,*

$$H(v, w) = \mathbb{P}(v \text{ adopts the opinion of } w \text{ at round } t)\,. \tag{3}$$

Proof. Note that, as each element of M is a transition matrix (the rows sum up to 1), H is the convex combination of transition matrices and thus is a transition matrix. To prove the second part note that by conditioning on the configuration ξ_t we have that

$$\mathbb{E}(\xi_{t+1}|\xi_t) = \sum_{M\in\mathcal{M}} l(M)(M\xi_t) = \left(\sum_{M\in\mathcal{M}} l(M)M\right)\xi_t = H\xi_t. \tag{4}$$

Choose ξ_t such that the opinion of w is 1 and all other opinions are 0. Then the event $\{v$ adopts the opinion of w at round $t\}$ is equal to $\{\xi_{t+1}(v) = 1\}$. Thus, from equation (4)

$$\mathbb{P}(\xi_{t+1}(v) = 1|\xi_t) = \mathbb{E}(\xi_{t+1}(v)|\xi_t) = (H\xi_t)(v) = \sum_{w\in V} H(v,w)\xi_t(w) = H(v,w). \quad \blacktriangleleft$$

Let P be the transition matrix of a simple random walk on G, A the adjacency matrix of G and let I denote the identity matrix. Let $L = D - A$ be the combinatorial Laplacian where D is the diagonal matrix containing the degree sequence of G. Moreover, let F be the diagonal matrix defined by $F(v,v) = \sum_{w:w\sim v} 1/d(w)$. The next theorem gives the matrix H for the linear voting models used in our examples.

▶ **Theorem 5.** *The mean matrix of the synchronous pull, asynchronous pull, push, abusive push, pull-push models are, respectively,* $H_a = P$ *[6] and*

$$H_b = \frac{n-1}{n}I + \frac{1}{n}P, \quad H_c = I + \frac{1}{n}P^\top - \frac{1}{n}F, \quad H_d = I - \frac{1}{n}L, \quad H_e = \frac{1}{n}(P+P^\top) + \frac{n-1}{n}I - \frac{1}{n}F.$$

Proof Sketch. We compute H_a. Observe that $H_a(v,w)$ is the probability that v adopts the opinion of w. That happens only if the random neighbour selected for v is w. Then $H_a(v,w) = \frac{1}{d(v)}\mathbb{1}_{v\sim w}$, concluding that $H_a = P$. For H_b, remember that in asynchronous pull we select a random vertex v and then v adopts the opinion of a random neighbour $w(v)$. Observe that for a vertex u we have $H_b(u,u)$ is the probability that u adopts the opinion of u, i.e. the probability that u does not change the opinion. That happen with probability $(n-1)/n$, On the other hand if $w \sim v$ then we have $H_b(v,w) = 1/nd(v)$ because v has to be initially selected and then v has to select w from its neighbourhood. We conclude that $H_b = ((n-1)/n)I + (1/n)P$. The other cases are similar. ◀

3 Winning probability

The most basic question in any voting model is, 'who wins?'. In order to answer this question we use some martingale arguments. Assume the two-party model, $Q = \{0,1\}$. Since the mean matrix H of a linear voting model is a transition matrix, then all its eigenvalues lie in $[-1,1]$. We order the eigenvalues in decreasing order, i.e. $1 = \lambda_1 \geq \lambda_2 \ldots, \geq \lambda_n$. Let λ be an eigenvalue of $H^\top$ (H and $H^\top$ have the same eigenvalues) with corresponding eigenvector f, that is $H^\top f = \lambda f$. Given $f, g \in \mathbb{R}^V$, we denote $\langle f, g\rangle = \sum_{v\in V} f(v)g(v)$ the standard inner product. Observe that $Q \subseteq \mathbb{R}$, so if $\xi \in Q^V$ and $f \in \mathbb{R}^V$, the inner product $\langle f, \xi\rangle = \sum_{v\in V} f(v)\xi(v)$ is well-defined.

▶ **Lemma 6.** *The process* $(\langle f, \xi_t\rangle/\lambda^t)_{t\geq 0}$ *is a martingale with respect to* $(\xi_t)_{t\geq 0}$

Proof. Since $\langle f, \xi_t\rangle$ is bounded, we can check that $\mathbb{E}(\langle f, \xi_{t+1}\rangle|\xi_t) = \lambda\langle f, \xi_t\rangle$ and divide both sides by λ^{t+1}. By linearity of (conditional) expectation and equation (4) we have

$$\mathbb{E}(\langle f, \xi_{t+1}\rangle|\xi_t) = \langle f, H\xi_t\rangle = \langle H^\top f, \xi_t\rangle = \lambda\langle f, \xi_t\rangle . \quad \blacktriangleleft$$

Since H is a transition matrix, if the associated Markov chain is recurrent and aperiodic then the Markov chain has a unique stationary distribution. Denote this stationary distribution by μ. It is a classic result of the theory of finite Markov chains that μ, interpreted as a vector, is the unique eigenvector of $H^\top$ with eigenvalue 1. We assume the vector μ is scaled so that $\sum_{v\in V}\mu(v)=1$. Since, among all eigenvectors, μ is the most important we denote by $m_t=\langle\mu,\xi_t\rangle$ the martingale associated with the eigenvalue 1, and we call this martingale the **voting martingale**.

Proof of Theorem 1. Denote by $\mathbf{1}$ and $\mathbf{0}$ the vector where all components are 1 and 0 respectively. Since $(\xi_t)_{t\geq 0}$ always reaches consensus, it converges to $\mathbf{1}$ or $\mathbf{0}$ and thus $(m_t)_{t\geq 0}$ converges to 1 or 0. Moreover, $0\leq m_t=\sum_{v\in V}\mu(v)\xi_t(v)\leq 1$ for every $\xi_t\in\{0,1\}^V$, so $(m_t)_{t\geq 0}$ is a bounded martingale. These two properties of $(m_t)_{t\geq 0}$, together with the fact that τ_{cons} is a stopping time, allows us to apply the optional stopping theorem [5] to conclude $\mathbb{E}(m_0)=\mathbb{E}(m_{\tau_{\text{cons}}})$. Since $\xi_0=\xi$ is a deterministic quantity then $\mathbb{E}(m_0)=m_0$. Moreover

$$\mathbb{E}(m_{\tau_{\text{cons}}})=\langle\mu,\mathbf{1}\rangle\mathbb{P}(\xi_{\tau_{\text{cons}}}=\mathbf{1}|\xi_0=\xi)+\langle\mu,\mathbf{0}\rangle\mathbb{P}(\xi_{\tau_{\text{cons}}}=\mathbf{0}|\xi_0=\xi)=\mathbb{P}(\xi_{\tau_{\text{cons}}}=\mathbf{1}|\xi_0=\xi).$$

Hence $\mathbb{P}(\xi_{\tau_{\text{cons}}}=\mathbf{1}|\xi_0=\xi)=m_0=\langle\mu,\xi\rangle$, therefore

$$\mathbb{P}(\text{opinion 1 wins}|\xi_0=\xi)=\sum_{v\in V}\mu(v)\xi(v).$$ ◀

▶ **Corollary 7.** *Assume the same conditions of Theorem 1 but consider $Q=\{1\ldots,,n\}$. Suppose that $\xi\in Q^V$. Then the probability that $k\in Q$ wins is*

$$\mathbb{P}(\xi_{\tau_{\text{cons}}}=k\mathbf{1}|\xi_0=\xi)=\sum_{v\in V:\xi(v)=k}\mu(v).$$

Proof. Replace opinion k by opinion 1 and all other opinions by opinion 0, and then use Theorem 1 ◀

▶ **Theorem 8.** *Let G be a connected graph. Let A be the set of vertices whose initial opinion is 1. Then, given that the models reach consensus, the probability p that opinion 1 wins is*

(a) *synchronous pull model: $p_a=d(A)/d(V)$*
(b) *asynchronous pull model: $p_b=d(A)/d(V)$*
(c) *push model: $p_c=(\sum_{v\in A}d(v)^{-1})/(\sum_{v\in V}d(v)^{-1})$*
(d) *abusive pushing model: $p_d=|A|/n$*
(e) *pull-push model : $p_e=|A|/n$.*

Proof. We apply Theorem 1. For that we need to find the stationary distribution of the above models. The stationary distribution of P is $\pi(v)=d(v)/d(V)$, that gives us the result for synchronous pull. Observe that $(n-1)/nI+(1/n)P$ is a lazy version of the random walk of P, then it has the same stationary distribution, giving us the result for the asynchronous pull model. For the push model we just guess the stationary distribution and check it. Let $C=1/(\sum_{v\in V}d(v)^{-1})$ and let $\pi'(v)=C/d(v)$, then as $F=F^\top$

$$\begin{aligned}(H_c^\top\pi)(v) &= ((I+\frac{1}{n}P-\frac{1}{n}F)\pi')(v)=\pi'(v)+\frac{1}{n}\sum_{w\in V}P(v,w)\pi'(w)-\frac{1}{n}F(v,v)\pi'(v)\\ &= \pi'(v)+\frac{1}{n}\sum_{w:w\sim v}\frac{1}{d(v)}\frac{C}{d(w)}-\frac{C}{d(v)n}\sum_{w:w\sim v}\frac{1}{d(w)}=\pi'(v)\end{aligned}$$

proving that π' is the stationary distribution of the mean matrix of the push model. For the abusive pushing model observe that as $I-(1/n)L$ is a symmetric matrix, its stationary distribution is uniform. H_e is also symmetric, giving the result for the push-pull model. ◀

4 Consensus Time

In this section we assume the two-party model with opinions $Q = \{0, 1\}$. Let $(\xi_t)_{t\geq 0} \sim \mathcal{LVM}(l, \xi)$ be a linear voting model. Assume $H = H(l)$ has a unique stationary distribution and let $(m_t)_{t\geq 0}$ be the voting martingale defined in Section 3. We use the following convenient notation. Let S_t be the set of vertices with opinion 1 at the beginning of round t, let $\mu(S_t) = m_t = \langle \mu, \xi_t \rangle$, and let $Z_t = \mu(S_{t+1}) - \mu(S_t)$. Note that, since $\mu(S_t)$ is a martingale, $\mathbb{E}(Z_t | S_t = S) = 0$. The random variable Z_t gives us information about the change in the measure of the set S_t. A larger value of $|Z_t|$ implies voting finishes faster.

Let $\eta(S) = \min\{\mu(S), \mu(S^c)\}$, where $\mu(S^c) = 1 - \mu(S)$. Denote by η_t the process $\eta(S_t)$. Since $\mu(S_t) \in [0, 1]$ we have $\eta_t \in [0, 1/2]$. Recall that $\mu(V) = 1$ and $\mu(\emptyset) = 0$. Note that $\eta_{t+1} = \min\{\mu(S_t) + Z_t, \mu(S_t^c) - Z_t\}$. Noting that if $\eta_t = \mu(S_t)$, i.e. $\mu(S_t) \leq \mu(S_t^c)$, then

$$\eta_{t+1} \leq \mu(S_{t+1}) = \mu(S_t) + Z_t = \eta_t + Z_t,$$

and if $\eta_t = \mu(S_t^c)$, the same applies by noticing that $\mu(S_{t+1}^c) - \mu(S_t^c) = -Z_t$, i.e.

$$\eta_{t+1} \leq \mu(S_{t+1}^c) = \mu(S_t^c) - Z_t = \eta_t - Z_t,$$

then in both cases we get

$$\eta_{t+1} \leq \eta_t + \rho_t Z_t, \tag{5}$$

where $\rho_t = \rho(S_t) = 2\mathbb{1}_{\{\mu(S_t) \leq \mu(S_t^c)\}} - 1$. Observe $\rho_t \in \{-1, +1\}$. We are going to study the process $\sqrt{\eta_t}$, in particular, $\mathbb{E}(\sqrt{\eta_t})$. Define $\Upsilon(S)$ by

$$\Upsilon(S) = \mathbb{E}\left(Z_t^2 \mathbb{1}_{\{\rho_t Z_t < 0\}} \middle| S_t = S\right) \tag{6}$$

and define $\Upsilon = \min \frac{\Upsilon(S)}{\eta(S)}$, where the minimum is over all $S \subseteq V$ but $S \neq \emptyset$ and $S \neq V$. With these ingredients we are ready to prove a technical lemma, which is fundamental for the proof of Theorem 2.

▶ **Lemma 9.** *Let $(\xi_t)_{t\geq 0} \sim \mathcal{LVM}(l, \xi)$ with $\xi \in \{0, 1\}^V$ be a voting model with $\Upsilon > 0$ then*

$$\mathbb{E}(\tau_{\text{cons}}) \leq 32/\Upsilon.$$

Proof. We borrow part of the argument from [1]. Let $S \subseteq V$ but $S \neq \emptyset$ and $S \neq V$. By conditioning on $S_t = S$, from equation (5) we have $\eta_{t+1} \leq \eta_t + \rho_t Z_t = \eta(S) + \rho_t Z_t$ (we replace η_t by $\eta(S)$ as $S_t = S$ is fixed). Then, by taking expectations

$$\begin{aligned}
\mathbb{E}(\sqrt{\eta_{t+1}} | S_t = S) &\leq \sqrt{\eta(S)}\mathbb{E}\left(\sqrt{1 + \frac{\rho_t Z_t}{\eta_t}} \middle| S_t = S\right) \\
&= \sqrt{\eta(S)}\mathbb{E}\left(\left(\sqrt{1 + \frac{\rho_t Z_t}{\eta_t}}\right) \mathbb{1}_{\{\rho_t Z_t \geq 0\}} \middle| S_t = S\right) \qquad (7) \\
&\quad + \sqrt{\eta(S)}\mathbb{E}\left(\left(\sqrt{1 + \frac{\rho_t Z_t}{\eta_t}}\right) \mathbb{1}_{\{\rho_t Z_t < 0\}} \middle| S_t = S\right). \qquad (8)
\end{aligned}$$

Let $x = \rho_t Z_t / \eta_t$. It can be checked that $x \geq -1$. Indeed, from equation (5) we have $\rho_t Z_t \geq \eta_{t+1} - \eta_t \geq -\eta_t$, concluding $x \geq -1$.

For $x \geq -1$ the following partial Taylor expansions are valid,

$$\sqrt{1+x} \leq 1 + \frac{x}{2}, \tag{9}$$

$$\sqrt{1+x} \leq 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{x^3}{16}. \tag{10}$$

To upper bound (7) use (9), and for (8) use (10). Recall that, since $\mu(S_t)$ is a martingale, then $\mathbb{E}(Z_t|S_t = S) = 0$. After some rearrangement, we obtain

$$\begin{aligned}
\mathbb{E}(\sqrt{\eta_{t+1}}|S_t = S) &\leq \sqrt{\eta(S)} - \sqrt{\eta(S)}\mathbb{E}\left(\left(\frac{(\rho_t Z_t)^2}{8\eta_t^2} - \frac{(\rho_t Z_t)^3}{16\eta_t^3}\right)\mathbb{1}_{\{\rho_t Z_t<0\}}\middle| S_t = S\right) \\
&\leq \sqrt{\eta(S)} - \sqrt{\eta(S)}\mathbb{E}\left(\frac{Z_t^2}{8\eta_t^2}\mathbb{1}_{\{\rho_t Z_t<0\}}\middle| S_t = S\right) \\
&= \sqrt{\eta(S)} - \frac{\Upsilon(S)}{8\eta(S)^{3/2}} \leq \sqrt{\eta(S)} - \frac{\Upsilon}{8\eta(S)^{1/2}}
\end{aligned} \tag{11}$$

In the second inequality we used the fact that we are working in $\{\rho_t Z_t < 0\}$ and after that we used the definition of $\Upsilon(S)$ from (6) and $\Upsilon = \min(\Upsilon(S)/\eta(S))$. Remember that $\eta(\emptyset) = \eta(V) = 0$, then

$$\begin{aligned}
\mathbb{E}(\sqrt{\eta_{t+1}}) &= \sum_{S\subseteq V} \mathbb{E}(\sqrt{\eta_{t+1}}|S_t = S)\mathbb{P}(S_t = S) = \sum_{S:S\neq\emptyset,V} \mathbb{E}(\sqrt{\eta_{t+1}}|S_t = S)\mathbb{P}(S_t = S) \\
&\leq \sum_{S:S\neq\emptyset,V} \left(\sqrt{\eta(S)} - \frac{\Upsilon}{8\eta(S)^{1/2}}\right)\mathbb{P}(S_t = S) \qquad (12) \\
&= \mathbb{E}(\sqrt{\eta_t}) - \sum_{S:S\neq\emptyset,V} \left(\frac{\Upsilon}{8\eta(S)^{1/2}}\right)\mathbb{P}(S_t = S|\tau_{\text{cons}} > t)\mathbb{P}(\tau_{\text{cons}} > t) \\
&= \mathbb{E}(\sqrt{\eta_t}) - \frac{\Upsilon}{8}\mathbb{E}\left(\frac{1}{\sqrt{\eta_t}}\middle|\tau_{\text{cons}} > t\right)\mathbb{P}(\tau_{\text{cons}} > t), \qquad (13)
\end{aligned}$$

where (12) follows using equation (11). As $1/x$ is convex for $x > 0$, apply Jensen's inequality to the random variable $x = \sqrt{\eta_t}$, to obtain

$$\mathbb{E}\left(\frac{1}{\sqrt{\eta_t}}\middle|\tau_{\text{cons}} > t\right) \geq \frac{1}{\mathbb{E}\left(\sqrt{\eta_t}\middle|\tau_{\text{cons}} > t\right)} = \frac{\mathbb{P}(\tau_{\text{cons}} > t)}{\mathbb{E}\left(\sqrt{\eta_t}\right)}. \tag{14}$$

The last equality holds because the event $\{\tau_{\text{cons}} \leq t\}$ implies that the vertices reached consensus, then $S_t = \emptyset$ or $S_t = V$, hence $\eta_t = 0$, and then

$$\begin{aligned}
\mathbb{E}(\sqrt{\eta_t}) &= \mathbb{E}(\sqrt{\eta_t}|\tau_{\text{cons}} > t)\mathbb{P}(\tau_{\text{cons}} > t) + \mathbb{E}(\sqrt{\eta_t}|\tau_{\text{cons}} \leq t)\mathbb{P}(\tau_{\text{cons}} \leq t) \\
&= \mathbb{E}(\sqrt{\eta_t}|\tau_{\text{cons}} > t)\mathbb{P}(\tau_{\text{cons}} > t).
\end{aligned}$$

By substituting (14) into (13) we obtain

$$\mathbb{E}(\sqrt{\eta_{t+1}}) \leq \mathbb{E}(\sqrt{\eta_t}) - \frac{\Upsilon}{8}\frac{\mathbb{P}(\tau_{\text{cons}} > t)^2}{\mathbb{E}\left(\sqrt{\eta_t}\right)},$$

then as $\eta_t \in [0, 1/2]$

$$\frac{\Upsilon}{8}\mathbb{P}(\tau_{\text{cons}} > t)^2 \leq \mathbb{E}(\sqrt{\eta_t})(\mathbb{E}(\sqrt{\eta_t}) - \mathbb{E}(\sqrt{\eta_{t+1}})) \leq \frac{\mathbb{E}(\sqrt{\eta_t}) - \mathbb{E}(\sqrt{\eta_{t+1}})}{\sqrt{2}}.$$

Summing from $t = 0$ up to a time $T - 1$ we have

$$\Upsilon \sum_{t=0}^{T-1} \mathbb{P}(\tau_{\text{cons}} > t)^2 \leq 8 \frac{\mathbb{E}(\sqrt{\eta_0}) - \mathbb{E}(\sqrt{\eta_T})}{\sqrt{2}} \leq 4. \tag{15}$$

Let T be the minimum time t such that $\mathbb{P}(\tau_{\text{cons}} > t) < 1/2$, then for every $t < T$ we have $\mathbb{P}(\tau_{\text{cons}} > t) \geq 1/2$. Therefore, from equation (15), it holds that

$$T \leq 16/\Upsilon.$$

Note that our bound for T is independent of the initial position, so we assume the worst case. We compute $\mathbb{E}(\tau_{\text{cons}})$ by looking at the process every T steps. If at round T the process finished then $\tau_{\text{cons}} \leq T$, otherwise, we restart the process and look again after T steps until we reach consensus. As the probability the process does not finish in T steps is at most $1/2$, we conclude that

$$\mathbb{E}(\tau_{\text{cons}}) \leq \sum_{k=1}^{\infty} kT \left(\frac{1}{2}\right)^k \leq 2T \leq \frac{32}{\Upsilon}. \qquad \blacktriangleleft$$

We need the following simple lemma.

▶ **Lemma 10.** *Let X be an integrable random variable with mean 0 then*

$$\mathbb{E}(|X|\mathbb{1}_{\{X<0\}}) = \mathbb{E}(|X|)/2.$$

Proof. Let $X^+ = X\mathbb{1}_{\{X>0\}}$ and $X^- = |X|\mathbb{1}_{\{X<0\}}$. Clearly $X = X^+ - X^-$ and $|X| = X^+ + X^-$. Then we have the system of equations

$$\begin{aligned} \mathbb{E}(X^+) - \mathbb{E}(X^-) &= \mathbb{E}(X) = 0, \\ \mathbb{E}(X^+) + \mathbb{E}(X^-) &= \mathbb{E}(|X|). \end{aligned}$$

Then $\mathbb{E}(X^-) = \mathbb{E}(X^+) = \mathbb{E}(|X|)/2$. ◀

We proceed with the proof of Theorem 2

Proof of Theorem 2. From Lemma 9 we have

$$\mathbb{E}(\tau_{\text{cons}}) \leq 32/\Upsilon,$$

where $\Upsilon = \min \frac{\Upsilon(S)}{\eta(S)}$ and the minimum is over all $S \subseteq V$ other than $S = \emptyset$ and $S = V$. Observe that if $|Z_t| > 0$, it means that at least one vertex changes its opinion, thus $|Z_t| \geq \mu^* = \min_{v \in V} \mu(v)$. From there

$$\begin{aligned} \Upsilon(S) &= \mathbb{E}\left(Z_t^2 \mathbb{1}_{\{\rho_t Z_t < 0\}} \middle| S_t = S\right) = \mathbb{E}\left((\rho_t Z_t \mathbb{1}_{\{\rho_t Z_t < 0\}})^2 \middle| S_t = S\right) \\ &\geq \mu^* \mathbb{E}\left(|\rho_t Z_t \mathbb{1}_{\{\rho_t Z_t < 0\}}| \middle| S_t = S\right) \end{aligned} \tag{16}$$

Note that $\mathbb{E}(\rho_t Z_t | S_t = S) = \rho(S)\mathbb{E}(Z_t|S_t) = 0$ because $\mu(S_t)$ is a martingale. Using Lemma 10 in equation (16), gives $\Upsilon(S) \geq \mu^* \mathbb{E}(|Z_t||S_t)/2$. Hence $\Upsilon \geq \mu^* \min \frac{1}{2} \frac{\mathbb{E}(|Z_t||S_t)}{\eta(S)}$. Recalling the definition of Ψ in equation 2, we conclude $\Upsilon \geq \Psi/2$ and therefore

$$\mathbb{E}(\tau_{\text{cons}}) \leq \frac{64}{\Psi}. \qquad \blacktriangleleft$$

We apply the above theorems to our examples. We use the following notation. Given $S \subseteq V$, denote by $E(S : S^c)$ the number of edges going from S to S^c. Denote by $d_S(v)$ the number of vertices of S adjacent to v. Observe that $E(S : S^c) = \sum_{v \in S} d_{S^c}(v) = \sum_{v \in S^c} d_S(v)$. We denote the graph conductance by $\Phi(G) = \min_{S \subseteq V} \frac{E(S:S^c)}{\min\{d(S), d(S^c)\}}$ where $0/0 = \infty$.

▶ **Example 11.** Consider the asynchronous pulling model on a graph G.

$$\mathbb{E}(|Z_t||S_t = S) = \sum_{v\in S} \frac{d(v)}{d(V)} \frac{1}{n} \frac{d_{S^c}(v)}{d(v)} + \sum_{v\in S^c} \frac{d(v)}{d(V)} \frac{1}{n} \frac{d_S(v)}{d(v)}.$$

Why? With probability $1/n$ we select vertex v and this vertex selects a random neighbour w with probability $1/d(v)$, and adopts its opinion. The stationary distribution of v is $\mu(v) = d(v)/d(V)$. If w has the same opinion as v, then $Z_t = 0$, but if w has the opposite opinion then $|Z_t| = d(v)/d(V)$. Then

$$\mathbb{E}(|Z_t||S_t = S) = \frac{1}{nd(V)} \left(\sum_{v\in S} d_{S^c}(v) + \sum_{v\in S^c} d_S(v) \right) = \frac{2E(S : S^c)}{nd(V)} \tag{17}$$

therefore from (2)

$$\Psi = \frac{d_{\min}}{d(V)} \frac{2}{n} \min_S \frac{E(S : S^c)}{\min\{d(S), d(S^c)\}} \tag{18}$$

Hence we conclude that $\mathbb{E}(\tau_{con}) = \mathcal{O}(nd(V)/d_{\min}\Phi)$. This gives a consensus time of $\mathcal{O}(n^2)$ for expanders, which is optimal up to a constant. For the cycle, $\mathcal{O}(n^3)$ optimal as well.

▶ **Example 12.** Consider the push model on a graph G. Let $C = (\sum_{v\in V} d(v)^{-1})^{-1}$.

$$\mathbb{E}(|Z_t||S_t = S) = \sum_{v\in S} \frac{C}{d(v)} \sum_{w:w\sim v, w\in S^c} \frac{1}{nd(w)} + \sum_{v\in S^c} \frac{C}{d(v)} \sum_{w:w\sim v, w\in S} \frac{1}{nd(w)}.$$

The above equation holds because to change the opinion of a vertex $v \in S$, the push model needs to select a vertex $w \in S^c$ adjacent to v and then w needs to push its opinion on v. That happens with probability $1/(nd(w))$. In such case, the change in $|Z_t|$ is $\mu(v) = C/d(v)$. The same applies if $v \in S^c$. Then

$$\mathbb{E}(|Z_t||S_t = S) = 2\frac{C}{n} \sum_{v\in S} \sum_{w\in S^c} \frac{\mathbb{1}_{v\sim w}}{d(v)d(w)}. \tag{19}$$

By using the notation $J(S) = \sum_{v\in S} d(v)^{-1}$ and that the stationary distribution is $\mu(v) = C/d(v)$ we have

$$\Psi = \frac{2C}{nd_{\max}} \min_S \frac{\sum_{v\in S} \sum_{w\in S^c} \frac{\mathbb{1}_{v\sim w}}{d(v)d(w)}}{\min\{J(S), J(S^c)\}}.$$

The parameter Ψ does not seem related to the classical graph parameters.

▶ **Example 13.** We continue with the abusive push model on a graph G.

$$\mathbb{E}(|Z_t||S_t = S) = \sum_{v\in S} \frac{1}{n} \frac{d_{S^c}(v)}{n} + \sum_{v\in S} \frac{1}{n} \frac{d_S(v)}{n}.$$

The above equation holds because with probability $1/n$ we sample a vertex v. Then v pushes its opinion on all its neighbours. Since the stationary distribution for this model is $\mu(v) = 1/n$, then the change in $|Z_t|$ is $d_{S^c}(v)/n$ if $v \in S$ and $d_S(v)/n$ if $v \in S^c$. Then $\mathbb{E}(|Z_t||S_t) = \frac{2}{n^2} E(S : S^c)$. Then it holds that

$$\Psi = \frac{2}{n^2} \min_S \frac{E(S : S^c)}{\min\{|S|, |S^c|\}} \tag{20}$$

The parameter $\min_S \frac{E(S:S^c)}{\min\{|S|,|S^c|\}}$ is very similar to the graph conductance, indeed, for d-regular graphs $\min_S \frac{E(S:S^c)}{\min\{|S|,|S^c|\}} = d\Phi(G)$. In such case we have that

$$\mathbb{E}(\tau_{\text{cons}}) = \mathcal{O}\left(\frac{n^2}{d\Phi}\right).$$

That gives us a $\mathcal{O}(n^2/d)$ time for regular expanders, which is optimal when the degree is constant. For the complete graph it gives us $\mathcal{O}(n)$, which is far from optimal, since the abusive push model finishes in just one round on the complete graph. For a cycle it gives us a $\mathcal{O}(n^3)$ time which is optimal.

▶ **Example 14.** Our final example is for the pull-push model. In this model the stationary distribution is uniform. Then the only way to produce a positive change in $|Z_t|$ is that when the random vertex v is chosen to pull and push, it selects one neighbour in S and the other in S^c. In that case, the change in $|Z_t|$ will be of $1/n$, then

$$\mathbb{E}(|Z_t||S_t = S) \quad = \quad \sum_{v\in V} \frac{1}{n^2} \frac{d_{S^c}(v) d_S(v)}{d(v)^2}.$$

Then

$$\Psi = \frac{1}{n^2} \min_S \sum_{v\in V} \frac{d_{S^c}(v) d_S(v)}{d(v)^2} \Big/ \min\{|S|, |S^c|\}.$$

Once again Ψ does not seem related to the classical graph parameters.

5 Discussion

In this paper we introduced and studied the linear voting model. The model can be seen as a generalisation of many models of voting. Despite its generality, the process is tractable and we can compute the probability that a given opinion wins. Moreover, by using a suitable potential function we were able to provide a bound for the expected consensus time. Furthermore, applying this bound in specific cases led to classical graph parameters, such as conductance, as well to other less familiar, or even new, parameters.

Future work includes the study of particular models on interesting graph families, like expanders, transitive graphs or random graphs, as well as the development of new techniques to analyse the model.

References

1 Petra Berenbrink, George Giakkoupis, Anne-Marie Kermarrec, and Frederik Mallmann-Trenn. Bounds on the voter model in dynamic networks. *arXiv preprint arXiv:1603.01895*, 2016.

2 Colin Cooper, Robert Elsasser, Hirotaka Ono, and Tomasz Radzik. Coalescing random walks and voting on connected graphs. *SIAM Journal on Discrete Mathematics*, 27(4):1748–1758, 2013.

3 Josep Díaz, Leslie Ann Goldberg, George B Mertzios, David Richerby, Maria Serna, and Paul G Spirakis. Approximating fixation probabilities in the generalized moran process. *Algorithmica*, 69(1):78–91, 2014.

4 Peter Donnelly and Dominic Welsh. Finite particle systems and infection models. *Mathematical Proceedings of the Cambridge Philosophical Society*, 94(01):167–182, 1983.

5 Rick Durrett. *Probability: theory and examples*. Cambridge university press, 2010.

6 Yehuda Hassin and David Peleg. Distributed probabilistic polling and applications to proportionate agreement. *Information and Computation*, 171(2):248–268, 2001.

7 David Asher Levin, Yuval Peres, and Elizabeth Lee Wilmer. *Markov chains and mixing times*. AMS Bookstore, 2009.

8 Toshio Nakata, Hiroshi Imahayashi, and Masafumi Yamashita. Probabilistic local majority voting for the agreement problem on finite graphs. In *Computing and Combinatorics*, pages 330–338. Springer, 1999.

9 Roberto Oliveira. On the coalescence time of reversible random walks. *Transactions of the American Mathematical Society*, 364(4):2109–2128, 2012.

10 Roberto Oliveira. Mean field conditions for coalescing random walks. *The Annals of Probability*, 41(5):3420–3461, 2013.

Discordant Voting Processes on Finite Graphs*†

Colin Cooper[1], Martin Dyer[2], Alan Frieze[3], and Nicolás Rivera[4]

1 **Department of Informatics, King's College London, London, UK**
colin.cooper@kcl.ac.uk
2 **School of Computing,University of Leeds, Leeds, UK**
M.E.Dyer@leeds.ac.uk
3 **Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA, USA**
alan@random.math.cmu.edu
4 **Department of Informatics, King's College London, London, UK**
nicolas.rivera@kcl.ac.uk

Abstract

We consider an asynchronous voting process on graphs which we call discordant voting, and which can be described as follows. Initially each vertex holds one of two opinions, red or blue say. Neighbouring vertices with different opinions interact pairwise. After an interaction both vertices have the same colour. The quantity of interest is T, the time to reach consensus, i.e. the number of interactions needed for all vertices have the same colour.

An edge whose endpoint colours differ (i.e. one vertex is coloured red and the other one blue) is said to be discordant. A vertex is discordant if its is incident with a discordant edge. In discordant voting, all interactions are based on discordant edges. Because the voting process is asynchronous there are several ways to update the colours of the interacting vertices.

- *Push:* Pick a random discordant vertex and push its colour to a random discordant neighbour.
- *Pull:* Pick a random discordant vertex and pull the colour of a random discordant neighbour.
- *Oblivious:* Pick a random endpoint of a random discordant edge and push the colour to the other end point.

We show that $\mathbf{E}T$, the expected time to reach consensus, depends strongly on the underlying graph and the update rule. For connected graphs on n vertices, and an initial half red, half blue colouring the following hold. For oblivious voting, $\mathbf{E}T = n^2/4$ independent of the underlying graph. For the complete graph K_n, the push protocol has $\mathbf{E}T = \Theta(n \log n)$, whereas the pull protocol has $\mathbf{E}T = \Theta(2^n)$. For the cycle C_n all three protocols have $\mathbf{E}T = \Theta(n^2)$. For the star graph however, the pull protocol has $\mathbf{E}T = O(n^2)$, whereas the push protocol is slower with $\mathbf{E}T = \Theta(n^2 \log n)$.

The wide variation in $\mathbf{E}T$ for the pull protocol is to be contrasted with the well known model of synchronous pull voting, for which $\mathbf{E}T = O(n)$ on many classes of expanders.

1998 ACM Subject Classification C.2.4 Distributed Systems, F.2 Analysis of algorithms, G.2 Discrete mathematics

Keywords and phrases Distributed consensus, Voter model, Interacting particles, Randomized algorithm

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.145

* A full version of this paper is available at http://arxiv.org/abs/1604.06884.
† This work was supported in part by EPSRC grant EP/M005038/1, "Randomized algorithms for computer networks", NSF grant DMS0753472,and Becas CHILE.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 145; pp. 145:1–145:13


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

1 Introduction

We consider a type of asynchronous distributed voting process on graphs which we call discordant voting, and which can be described as follows. Initially each vertex holds one of two opinions, red or blue say. Neighbouring vertices of different colours, i.e. whose opinions differ, interact pairwise. After an interaction both vertices have the same colour. If, at some step, all vertices have the same colour, we say that a consensus has been reached.

The problem of reaching consensus in graph by means of local interactions is an abstraction of the behavior of both human society and computer networks. As a consequence the process of voting on graphs has been widely studied. Distributed voting finds application in various fields of computing including consensus and leader election in large networks [4, 12], serialisation of read and write in replicated data-bases [10], and the analysis of social behavior in game theory [21]. Voting algorithms are usually simple, fault-tolerant, and easy to implement [12, 14]. Recently, there has been considerable interest in *population protocols*. In this model the interacting vertices can make limited computations using a finite state machine to address a wide range of problems in distributed computing, see e.g. [2].

The classical model, synchronous pull voting, is reasonably well understood. If the colours of the vertices are initially distinct, the randomized process takes $\Theta(n)$ expected steps to reach consensus on many classes of expander graphs on n vertices. This holds for the complete graph K_n (Aldous [1]), and almost all r-regular random graphs [7]. For general results based on the eigenvalue gap and variance of the degree sequence see [6]. Hassin and Peleg [12] and Nakata *et al.* [17] considered the two-party pull voting model on connected graphs, and discussed its application to consensus problems in distributed systems.

In contrast to the case of synchronous voting, where only the pull protocol is well defined, for asynchronous voting, there are at least three ways to update the colours of the interacting vertices.

- *Push:* Pick a random vertex and push its colour to a random neighbour.
- *Pull:* Pick a random vertex and pull the colour of a random neighbour.
- *Oblivious:* Pick a random endpoint of a random edge and push the colour to the other end point.

Discordant voting originated in the complex networks community as a model of social evolution (see e.g. [11], [18]). The general version of the model allows *rewiring*. The interacting vertices can break edges joining them and reconnect elsewhere. This serves as a model of social interaction in which vertices will either change their opinion or their friends.

Holme and Newman [13] investigated discordant voting as a model of a self-organizing network which restructures based on the acceptance or rejection of differing opinions among social groups. At each step, a random discordant edge uv is selected, and an endpoint $x \in \{u, v\}$ chosen with probability $1/2$. With probability $1 - \alpha$ the opinion of x is pushed to the other endpoint y, and with probability α, y breaks the edge and rewires to a random vertex with the same opinion as itself. Simulations suggested the existence of threshold behavior in α. This was investigated further by Durrett et al. [8] for sparse random graphs of constant average degree 4. The paper studies two rewiring strategies, rewire-to-random, and rewire-to-same, and finds experimental evidence of a phase transition in both cases. Basu and Sly [3] made a formal analysis of rewiring for Erdos-Renyi graphs $G(n, 1/2)$ with $1 - \alpha = \beta/n$, $\beta > 0$ constant. They found that for either strategy, if β is sufficiently small the network quickly disconnects maintaining the initial proportions. As β increases the minority proportion decreases, and in rewire-to-random a positive fraction of both opinions survive.

Although discordant voting seems a natural model of local interaction, its behavior, is

not well understood even in the simplest cases. The aim of this paper is a fundamental study of expected time to consensus in the absence of rewiring. As discordant voting always chooses an edge between the red and blue sets, it should be more efficient, and thus finish faster than an asynchronous pull voting process which ignores this information, and takes $\Omega(n^2)$ steps on many classes of sparse graphs (see [5]). However, we find the performance of discordant voting protocols vary considerably with the structure of the underlying graph, and sometimes in a quite counter-intuitive way.

We suppose that the initial vertex colours in the two-party voting model are red and blue, and let $R(t), B(t)$ denote the sets of vertices with the given colours at any step t. For the oblivious protocol, the expected time to completion is the same for any connected graph on n vertices and is independent of graph structure or the number of edges. It depends only on the initial number of vertices of each colour ($R(0)$, $B(0)$). Whenever a discordant edge is chosen, the number of blue vertices in the graph increases (resp. decreases) by one with probability $1/2$. This is equivalent to an unbiased random walk on the line $(0, 1, ..., n)$ with absorbing barriers, starting from $R(0) = r$ red vertices. Thus $\mathbf{E}T = r(n - r)$ (see Feller [9, XIV.3]).

▶ Remark. *Oblivious protocol.* Let T be the time to consensus in the two-party asynchronous discordant voting process starting from any initial coloring with an equal number of red and blue vertices $R = r, B = n - r$. For any connected n vertex graph, $\mathbf{E}T(\text{Oblivious}) = r(n - r)$.

In stark contrast to the oblivious protocol, the discordant push and pull protocols can exhibit very different expected times to consensus, and which depend strongly on the underlying graph in question.

▶ **Theorem 1.** *Let T be the time to consensus of the asynchronous discordant voting process starting from any initial coloring with an equal number of red and blue vertices $R = B = n/2$. For the complete graph K_n, and for random graphs $G_{n,p}$, $np \geq \log n\sqrt{n}$, $\mathbf{E}T(Push) = \Theta(n \log n)$, and $\mathbf{E}T(Pull) = \Theta(2^n)$.*

For reasons of brevity we do not reproduce the proof for $G_{n,p}$ here, but will make it available in the full version of this paper. The interesting point is that for the complete graph K_n and random graphs $G_{n,p}$ the different protocols give very different expected completion times, which vary from $\Theta(n \log n)$ for push, to $\Theta(n^2)$ for oblivious to $\Theta(2^n)$ for pull. On the basis of this evidence, our initial view was that there should be a meta-theorem of the "push is faster than oblivious, oblivious is faster than pull" type. Intuitively, this is supported by the following argument. Suppose red (R) is the larger colour class. Choosing a discordant vertex uniformly at random, favors the selection of the larger class. In the push process, red vertices push their opinion more often, which tends to increase the size of R. Conversely, the pull process tends to re-balance the set sizes. If R is larger, it is recoloured more often.

If the graph has limited expansion, the behavior of discordant voting differs considerably from the above examples. For the cycle C_n, all three protocols are similar.

▶ **Theorem 2.** *Let T be the time to consensus of the asynchronous discordant voting process starting from any initial coloring with an equal number of red and blue vertices $R = B = n/2$. For any of the Push, Pull or Oblivious protocols on the cycle C_n, $\mathbf{E}T = \Theta(n^2)$.*

At this point we were left with a difficult choice. Either to produce evidence for a relationship of the form $\mathbf{E}T(\text{Push}) = O(\mathbf{E}T(\text{Pull}))$, or to refute it. Mossel and Roch [16] found slow convergence of the iterated prisoners dialemma problem (IPD) on caterpillar trees. Intuitively push voting is aggressive, whereas pull voting is altruistic, and thus similar to cooperation in IPD. Motivated by this, we found the star graph S_n as a counter example.

▶ **Theorem 3.** *Let T be the time to consensus in the two-party asynchronous discordant voting process starting from any initial coloring with an equal number of red and blue vertices $R = B = n/2$. For the star graph S_n, $\mathbf{E}T(Push) = \Theta(n^2 \log n)$, and $\mathbf{E}T(Pull) = O(n^2)$.*

For stars, experiments show a clear difference in $\mathbf{E}T$ for three protocols. For cycles the difference is smaller and depends on the initial colouring. See Fig. 4 of Section 5.1.

A major problem in analysing discordant voting, is that the effect of recolouring a vertex is not always monotone. For each of the graphs studied, the way to bound $\mathbf{E}T$ differs. The proof of the pull voting result for the cycle C_n in particular, is somewhat delicate, and requires an analysis of the optimum of a linear program based on a potential function.

Asynchronous discordant voting model

We next give a formal definition of the discordant voting process. Given a graph $G = (V, E)$, with $n = |V|$. Each vertex $v \in V$ is labelled with an *opinion* $X(v) \in \{0, 1\}$. We call X a *configuration* of opinions. We can think of the opinions as having colours; e.g. red (0) and blue (1), or black (0) and white (1) (see e.g. Figure 2). An edge $e = uv \in E$ is *discordant* if $X(u) \neq X(v)$. Let $K(X)$ denote the set of discordant edges at time t. A vertex v is discordant if it is incident with any discordant edge, and $D(X)$ will denote the set of discordant vertices in X. We consider three random update rules for opinions X_t at time t.

Push: Choose $v_t \in D(X_t)$, uniformly at random, and a discordant neighbour u_t of v_t uniformly at random. Let $X_{t+1}(u_t) \leftarrow X_t(v_t)$, and $X_{t+1}(w) \leftarrow X_t(w)$ otherwise.

Pull: Choose $v_t \in D(X_t)$, uniformly at random, and a discordant neighbour u_t of v_t uniformly at random. Let $X_{t+1}(v_t) \leftarrow X_t(u_t)$, and $X_{t+1}(w) \leftarrow X_t(w)$ otherwise.

Oblivious: Choose $\{u_t, v_t\} \in K(X_t)$ uniformly at random. With probability $1/2$, $X_{t+1}(v_t) \leftarrow X_t(u_t)$, with probability $1/2$, $X_{t+1}(u_t) \leftarrow X_t(v_t)$, and $X_{t+1}(w) \leftarrow X_t(w)$ otherwise.

These three processes are Markov chains on the configurations in G, in which the opinion of exactly one vertex is changed at each step. Assuming G is connected, there are two absorbing states, when $X(v) = 0$ for all $v \in V$, or $X(v) = 1$ for all $v \in V$, where no discordant vertices exist. When the process reaches either of these states, we say that is has converged. Let T be the step at which convergence occurs. Our object of study is $\mathbf{E}T$.

Structure of the paper. In Section 2 we prove results for a Birth-and-Death chain which we call the Push chain. This chain can be coupled with many aspects of the discordant voting process. We then prove Theorems 1, 2 and 3 in that order.

2 Birth-and-Death chains

A Markov chain $(X_t)_{t\geq 0}$ is said to be a Birth-and-Death chain on state space $S = \{0, \ldots, N\}$ if given $X_t = i$ then the possible values of X_{t+1} are $i + 1, i$ or $i - 1$ with probability p_i and q_i respectively. We assume that $q_0 = p_N = 0$, and $p_0 = 1$, $q_N = 1$, and $p_i > 0$, $q_i > 0$ otherwise. Denote by $\mathbf{E}_i T_j$ the expected hitting time of state j starting from state i, i.e. $T_j = \min\{t \geq 0 : X_t = j,\ X_0 = i\}$. We summarize the results we require on Birth-and-Death chains (see Peres, Levin and Wilmer [15, 2.5]).

A probability distribution π satisfies the detailed balance condition, if

$$\pi(i)P(i, j) = \pi(j)P(j, i), \text{ for all } i, j \in S. \tag{1}$$

Birth-and-Death chains with $p_i = P(i, i+1), q_i = P(i, i-1)$ can be shown to satisfy the detailed balance equations. It follows from this, (see e.g. [15]) that

$$\mathbf{E}_{i-1}T_i = \frac{1}{q_i\pi(i)}\sum_{k=0}^{i-1}\pi(k) \tag{2}$$

An equivalent formulation (see [15]) is $\mathbf{E}_0T_1 = 1/p_0 = 1$ and in general

$$E_{i-1}T_i = \sum_{k=0}^{i-1}\frac{1}{p_k}\frac{q_{k+1}\cdots q_{i-1}}{p_{k+1}\cdots p_{i-1}}, \qquad \text{for } i \in \{1,\ldots,N\}. \tag{3}$$

In writing this expression we follow the convention that if $k = i-1$ then $\frac{q_{k+1}\cdots q_{i-1}}{p_{k+1}\cdots p_{i-1}} = 1$ so that the last term is $1/p_{i-1}$. Note also that the final index k on p_k is $k = N-1$, i.e. we never divide by $p_N = 0$.

Starting from state 0, let T_M be the number of transitions needed to reach state M for the first time. For any $M \le N$, we have that $\mathbf{E}_0T_M = \sum_{i=1}^{M}\mathbf{E}_{i-1}T_i$. For example, $\mathbf{E}_0T_1 = \frac{1}{p_0} = 1$ and $\mathbf{E}_0T_2 = 1 + \frac{1}{p_1} + \frac{q_1}{p_0p_1}$ etc. Thus, for $M \ge 1$

$$\mathbf{E}_0T_M = \sum_{i=1}^{M}\mathbf{E}_{i-1}T_i = \sum_{i=1}^{M}\sum_{k=0}^{i-1}\frac{1}{p_k}\prod_{j=k+1}^{i-1}\frac{q_j}{p_j}. \tag{4}$$

We next define a Birth-and-Death chain, the push chain, which features in our analysis. The chain has states $\{0, 1, ..., i, ..., N\}$ where $N = n/2$ (assume $n \ge 2$ even). The transition probabilities from state i given by $P(i, i+1)$, $Q(i, i+1) = 1 - P(i, i+1)$.

Let Z_t be the state occupied by the push chain at step $t \ge 0$. Let $\delta \in \{-1, 0, +1\}$ be fixed. When applying results for the push chain in our proofs, we will state the value of δ we use. The transition probability $p_i = P(i, i+1)$ from $Z_t = i$, is given by

$$p_i = \begin{cases} 1, & \text{if } i = 0 \\ 1/2 + i/n + \delta/n, & \text{if } i \in \{1,\ldots,n/2-1\} \\ 0, & \text{if } i = n/2 \end{cases}. \tag{5}$$

For a proof of the following lemma see [5].

▶ **Lemma 4.** *Let $\mathbf{E}_0T_M$ be the expected hitting time of M in the push chain Z_t starting from state 0. For any $M \le N$,*

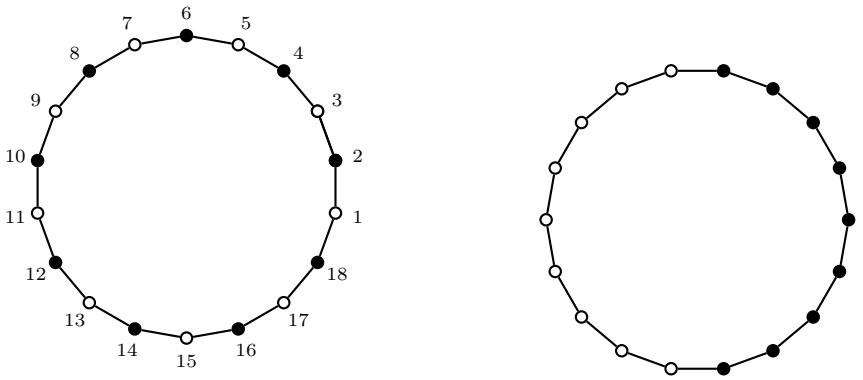
$$\mathbf{E}_0T_M \le 2N\log M + O(1). \tag{6}$$

For any $\sqrt{N} \le M = o(N^{3/4})$, there exists a constant $C > 0$ such that

$$\mathbf{E}_0T_M \ge C(N\log(M/\sqrt{N}) + \sqrt{N}). \tag{7}$$

3 Voting on the complete graph K_n

For the complete graph K_n, the probability B increases at a given step is $B(t)/n$, whereas in the pull process it is $R(t)/n = 1 - B(t)/n$. The chain defined by $Y_t = \max\{R(t), B(t)\} - n/2$ is a Birth-and-Death chain. We study the time that takes Y_t to reach $N = n/2$ starting from 0.



(a) All $X(i) = i \bmod 2$ **(b)** Lower bound configuration

Figure 1 A cycle with $n = 18$. Example colourings.

Push process. For the push model, the process Y_t is identical to the push chain Z_t with transitions p_i given by (5), with $\delta = 0$. The result of Theorem 1 that $\mathbf{E}T(\text{Push}) = \Theta(n \log n)$ follows from Lemma 4.

Pull process. As pull is the opposite of push, the pull process Y_t has transitions given by $\overline{p}_i = 1 - p_i$, i.e. . Thus $\overline{p}_0 = 1$, $\overline{p}_i = 1/2 - i/n$ if $i \in \{1, \dots, N-1\}$, and $\overline{p}_N = 0$.

Let $w_k = \binom{n}{N+k}, k = 0, 1, \dots, N$. Then w_k satisfies the detailed balance equation (1). Hence we have $\pi(k) = w_k/W$, where $W = w_0 + w_1 + \cdots + w_N$. It follows from (2) that

$$\mathbf{E}_{i-1}T_i = \frac{2n}{n+2i} \cdot \frac{1}{\binom{n}{N+i}} \cdot \sum_{k=0}^{i-1} \binom{n}{N+k}.$$

Putting $i = N$ we have

$$\mathbf{E}_{N-1}T_N = \sum_{k=0}^{N-1} \binom{n}{N+k} = \frac{1}{2}\left(2^n - 2 + \binom{n}{N}\right) = \Omega(2^n).$$

On the other hand, an upper bound

$$\sum_{i=1}^{N} \mathbf{E}_{i-1}T_i \le 2 \cdot 2^n \cdot \sum_{i=1}^{N} \frac{1}{\binom{n}{N+i}} = O(2^n),$$

follows from a result of Sury [19], that

$$\sum_{i=1}^{N} \frac{1}{\binom{n}{N+k}} = \frac{n+1}{2^n} \sum_{i=0}^{n} \frac{2^i}{i+1} = O(1).$$

4 Voting on the cycle

An n-cycle G, with $V = [n]$, has $E = \{(i, i+1) : i \in [n]\}$, where we identify vertex $n+i$ with vertex i. See Fig. 1(i).

If $X(i) \neq X(i+1) = X(i+2) = \cdots = X(j) \neq X(j+1)$, we say $i+1, i+2, \dots, j$ is a *run* of vertices of length $(j-i)$ $(1 \le j - i < n)$. Note that the number of runs is equal to the number of discordant edges $k(X)$. Also k is even, since red and blue runs must alternate,

so we can write $r(X) = \frac{1}{2}k(X)$, and $k_0 = 2r_0 = k(X_0)$. Thus $r(X)$ is the number of paths of a given colour. A *singleton* is a run of length 1. Since they lie in two discordant edges, singletons require special treatment. Let $s(X)$ denote the number of singletons. There are $\kappa = 2k - s$ discordant vertices, so $k \le \kappa \le 2k$.

We wish to determine the convergence time T for an arbitrary configuration X_0 of the push or pull process to reach an absorbing state X_T with $X_T(i) = X_T(1)$ $(i \in [n])$. In this process, the run lengths behave rather like symmetric random walks on the line. However, an analysis using classical random walk techniques [9] seems problematic. There are two main difficulties. Firstly, the k "walks" (run lengths) are correlated. If a run is long, the adjacent runs are likely to be shorter, and vice versa. Secondly, when the recoloured vertex is a singleton, the three adjacent runs are combined, so three walks suddenly merge into one. One of the three runs is a singleton, but the other two may have arbitrary lengths. We use the random walk view only for a lower bound on the convergence time.

▶ **Lemma 5.** *Let G is an n-cycle, with $n = 2N$ even, and suppose the process starts with $X_0(i) = 0$ $(i = 1, \ldots, N)$, $X_0(i) = 1$ $(i = N+1, \ldots, n)$, then $\mathbf{E}[T] = \Omega(n^2)$.*

Let L_t be the length of (say) the red run at step t, so $L_0 = N$, (see Fig. 1(ii)), and $L_T \in \{0, n\}$. The number of runs $k(X_t)$ can only be reduced from two to zero if either $L_t = 1$ or $L_t = n-1$, when one of the runs is a singleton. Up to this point, L_t is a symmetric simple random walk and the push and pull processes proceed identically. Thus $\mathbf{E}[T]$ is bounded below by the expected time for a symmetric simple random walk started at N to reach either 1 or $(n-1)$. By Remark 1, $\mathbf{E}[T] \ge (N-1)^2 = \Omega(n^2)$.

4.1 Upper bound for push voting: Proof that $\mathbf{E}[T] = O(n^2)$

Let the k runs in X have lengths $\ell_1, \ell_2, \ldots, \ell_k$ respectively, thus $\sum_{i=1}^{k} \ell_i = n$. Thus T is the first t for which $k(X_t) = r(X_t) = 0$, (a cycle is not a path). For an upper bound on $\mathbf{E}[T]$, we define a *potential function*

$$\psi(X) \;=\; \sum_{i=1}^{k} \sqrt{\ell_i}\,,$$

where $\psi(X) = 0$ if and only if $k(X) = 0$. The important feature of ψ is that it is a separable and strictly concave function of the ℓ_i $(i \in [k])$.

▶ **Lemma 6.** *For any configuration X on the n-cycle with k runs, $\psi(X) \le \sqrt{kn}$.*

Proof. If $k = 0$, this is clearly true. Otherwise, if $k \ge 2$, by concavity we have $\psi(X)/k = \frac{1}{k}\sum_{i=1}^{k}\sqrt{\ell_i} \le \sqrt{\frac{1}{k}\sum_{i=1}^{k}\ell_i} = \sqrt{n/k}$, so $\psi(X) \le \sqrt{kn}$. ◀

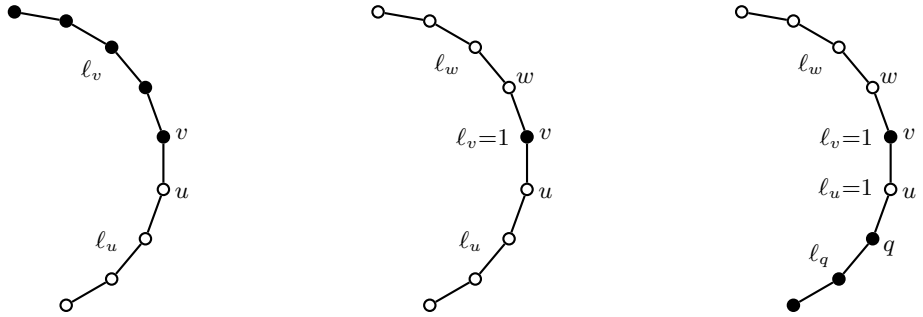
Observe that $k(X_{t+1}) = k(X_t)$ at step t of either the push or pull process, unless the recoloured vertex is a singleton, in which case we may have $k(X_{t+1}) = k(X_t) - 2$. Thus $\{t : k(X_t) = 2r\}$ is an interval $[t_r, t_{r-1})$, which we will call *phase* r of the process.

Let $v_t = v \in D(X_t)$ be the active vertex, i.e. the vertex selected to push in the push rule, or pull in the pull rule. Let δ_v be the expected change in ψ, i.e.

$$\delta_v \;=\; \mathbf{E}[\,\psi(X_{t+1}) - \psi(X_t) \mid v_t = v\,].$$

If there are $\kappa = 2k - s$ discordant vertices, the total expected change δ in ψ is

$$\delta \;=\; \mathbf{E}[\psi(X_{t+1}) - \psi(X_t)] \;=\; \frac{1}{\kappa}\sum_{v \in D} \delta_v. \tag{8}$$



(A) u and v not singleton **(B)** u not singleton, v singleton **(C)** u and v both singleton

Figure 2 Cases for discordant edge uv.

We will show that δ is negative, so $\psi(X_t)$ is monotonically decreasing with t, in expectation. Unfortunately we cannot simply bound δ_v for each $v \in D$, since it is possible to have $\delta_v > 0$. Thus we will consider discordant *edges*. We partition the set K of discordant edges uv into three subsets, Note that k can change only if $uv \in B \cup C$.

(A) $A = \{uv : u \text{ and } v \text{ not singleton}\}$;
(B) $B = \{uv : u \text{ not singleton, } v \text{ singleton}\}$;
(C) $C = \{uv : u \text{ and } v \text{ both singleton}\}$.

See Fig. 2. Let ℓ_z be the length of the run containing discordant vertex z, for $z \in \{u, v, w, q\}$.

Now let

$$\lambda_{uv} = \begin{cases} \sqrt{\ell_u} + \sqrt{\ell_v}, & uv \in A\,; \\ \sqrt{\ell_u} + \frac{1}{2}\sqrt{\ell_v}, & uv \in B\,; \\ \frac{1}{2}\sqrt{\ell_u} + \frac{1}{2}\sqrt{\ell_v}, & uv \in C\,. \end{cases} \qquad \delta_{uv} = \begin{cases} \delta_u + \delta_v, & uv \in A\,; \\ \delta_u + \frac{1}{2}\delta_v, & uv \in B\,; \\ \frac{1}{2}\delta_u + \frac{1}{2}\delta_v, & uv \in C\,. \end{cases}$$

Each singleton is in two discordant edges, all other discordant vertices in one, and each run is bounded by two discordant vertices. Therefore

$$\psi = \tfrac{1}{2}\sum_{v\in D}\sqrt{\ell_v} = \sum_{uv\in K}\lambda_{uv}\,, \qquad \delta = \frac{1}{\kappa}\sum_{v\in D}\delta_v = \frac{1}{\kappa}\sum_{uv\in K}\delta_{uv}\,.$$

We will show that $\delta_{uv} < 0$ for all $uv \in K$. For the proof of the following lemma see [5].

▶ **Lemma 7.** *For all three cases (A)–(C), and for all $uv \in K$,*
For push voting, $\delta_{uv} < -\frac{1}{5}(\ell_v^{-3/2} + \ell_u^{-3/2})$. For pull voting, $\delta_{uv} < -\frac{1}{10}(\ell_v^{-3/2} + \ell_u^{-3/2})$.

The following proof that $\mathbf{E}[T] = O(n^2)$ is for push voting. The upper bound on $\mathbf{E}[T]$ for pull voting is at most twice that for push. Using Lemma 7 we evaluate δ in (8).

$$\delta = \frac{1}{\kappa}\sum_{v\in D}\delta_v = \frac{1}{\kappa}\sum_{uv\in K}\delta_{uv} \le -\frac{1}{5\kappa}\sum_{uv\in K}(\ell_v^{-3/2} + \ell_u^{-3/2}) < -\frac{1}{5\kappa}\sum_{v\in D}\ell_v^{-3/2}\,.$$

Thus

$$\mathbf{E}[\psi(X_{t+1})] < \psi(X_t) - \frac{1}{5\kappa}\sum_{v\in D}\ell_v^{-3/2}\,.$$

Since $f(x) = x^{-3}$ is a convex function, $\mathbf{E}[f(X)] \ge f(\mathbf{E}[X])$ by Jensen's inequality [20, 6.6], so

$$\frac{1}{\kappa}\sum_{v\in D}\ell_v^{-3/2} \ge \Big(\frac{1}{\kappa}\sum_{v\in D}\sqrt{\ell_v}\Big)^{-3} = \Big(\frac{\kappa}{2\psi(X_t)}\Big)^3 \ge \Big(\frac{k}{2\psi(X_t)}\Big)^3,$$

Therefore,

$$\mathbf{E}[\psi(X_{t+1})] \;<\; \psi(X_t) - \frac{1}{5}\Big(\frac{k}{2\psi(X_t)}\Big)^3 \;=\; \psi(X_t) - \frac{k^3}{40\psi(X_t)^3}\,. \tag{9}$$

Recall that for $r \in [r_0]$, phase r of the process, during which the number of runs is $k = 2r$, is the interval $[t_r, t_{r-1})$. During phase r, by Lemma 6, $\psi(X_t) \le \sqrt{kn}$. Using this in (9) gives

$$\mathbf{E}[\psi(X_{t+1})] - \psi(X_t) \;\le\; -\tfrac{1}{40}k^3/(kn)^{3/2} \;=\; -\tfrac{1}{40}(k/n)^{3/2}\,. \tag{10}$$

Let $\gamma_r = \frac{1}{40}(2r/n)^{3/2}$. Then (10) implies that $Y_t = \psi(X_t) + (t - t_r)\gamma_r$ is a supermartingale [20, 10.3] during phase r, and t_{r-1} is a stopping time. Let $\varphi_r = \mathbf{E}[\psi(X_{t_r})]$, and let $m_r = \mathbf{E}[t_{r-1} - t_r]$. The optional stopping theorem [20, 10.10] implies that

$$\varphi_{r-1} + \gamma_r m_r \;=\; \mathbf{E}[\psi(X_{t_{r-1}}) + \gamma_r(t_{r-1} - t_r)] \;\le\; \mathbf{E}[\psi(X_{t_r})] \;=\; \varphi_r\,,$$

which implies

$$\varphi_r - \varphi_{r-1} \;\ge\; \gamma_r m_r \;=\; \tfrac{1}{40}m_r(2r/n)^{3/2} \qquad (r \in [r_0])\,. \tag{11}$$

Note, in particular, that $\varphi_r \ge \varphi_{r-1}$ for all $r \in [r_0]$. When $r_0 = \frac{1}{2}k(X_0)$, $t_{r_0} = 0$ and, since $r(X_T) = k(X_T) = 0$, when $t_0 = T$ then $\varphi_0 = 0$.

Let $x_r = \varphi_r - \varphi_{r-1} \ge 0$, for $r \in [r_0]$, so $\varphi_r = \sum_{i=1}^{r} x_j \le \sqrt{2rn}$. Also, from (11), we have $m_r \le 40x_r(n/2r)^{3/2} = 10\sqrt{2}\,n^{3/2}x_r/r^{3/2}$, so $\mathbf{E}[T] = \sum_{j=1}^{r_0} m_j < 10\sqrt{2}\,n^{3/2}\sum_{j=1}^{r_0} x_r/r^{3/2}$.

Thus $E[T]$ is bounded above by $T^\star$, the optimal value of the following linear program.

$$\begin{aligned} T^\star \;&=\; \max\; 10\sqrt{2}\,n^{3/2}\textstyle\sum_{r=1}^{r_0} x_r/r^{3/2} \\ \text{such that}\;\; \textstyle\sum_{j=1}^{r} x_j \;&\le\; \sqrt{2rn} && (r \in [r_0]) \\ x_j \;&\ge\; 0 && (j \in [r_0])\,. \end{aligned} \tag{12}$$

This linear program can be solved by a greedy procedure.

▶ **Lemma 8.** *Let $0 < b_1 < b_2 < \cdots < b_\nu$ and $c_1 > c_2 > \cdots > c_\nu > 0$. Then the linear program $\max \sum_{j=1}^{\nu} c_j x_j$ subject to $\sum_{j=1}^{r} x_j \le b_r$, $x_r \ge 0$ $(r \in [\nu])$ has optimal solution $x_1 = b_1$, $x_j = b_j - b_{j-1}$ $(j = 2, 3, \ldots, \nu)$.*

Proof. This solution has objective function value $c_1b_1 + c_2(b_2 - b_1) + \cdots + c_\nu(b_\nu - b_{\nu-1})$. The dual linear program is $\min \sum_{i=1}^{\nu} b_i y_i$ subject to $\sum_{i=j}^{\nu} y_i \ge c_j$, $y_j \ge 0$ $(j \in [\nu])$, and has feasible solution $y_\nu = c_\nu$, $y_j = c_j - c_{j+1}$ $(j \in [\nu - 1])$. Then the dual objective function has value $b_\nu c_\nu + b_{\nu-1}(c_{\nu-1} - c_\nu) + \cdots + b_1(c_1 - c_2)$. However,

$$c_1b_1 + c_2(b_2 - b_1) + \cdots + c_\nu(b_\nu - b_{\nu-1}) \;=\; b_\nu c_\nu + b_{\nu-1}(c_{\nu-1} - c_\nu) + \cdots + b_1(c_1 - c_2)\,.$$

Since the objective function values are equal, it follows that the two solutions are optimal in the primal and dual respectively. ◀

Thus, the optimal solution to (12) is $x_r = \sqrt{2nr} - \sqrt{2n(r-1)} = \sqrt{2nr}(1 - \sqrt{1 - 1/r}) \le \sqrt{2n/r}$, for $r \in [r_0]$, since $1 - y \le \sqrt{1-y}$ for $0 \le y \le 1$. Thus

$$T^\star \;\le\; 10\sqrt{2}\,n^{3/2}\sum_{j=1}^{r_0} x_r/r^{3/2} \;\le\; 10\sqrt{2}\,n^{3/2}\sum_{j=1}^{r_0}\sqrt{2n}/\big(\sqrt{r}\,r^{3/2}\big) \;=\; 20n^2\sum_{r=1}^{r_0} 1/r^2 \;<\; (10\pi^2/3)n^2\,,$$

since $\sum_{r=1}^{\infty} 1/r^2 = \pi^2/6$. Thus we have an absolute bound of $\mathbf{E}[T] = O(n^2)$.

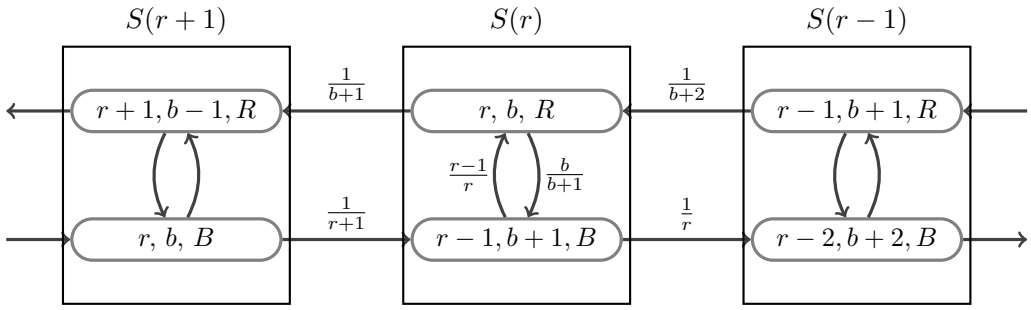


Figure 3 Star graph: Pseudo-states for the push process.

5 Theorem 3: Voting on the star graph S_n

In this section we prove $\mathbf{E}T(\text{Push}) = \Theta(n^2 \log n)$. The result that $\mathbf{E}T(\text{Pull}) = O(n^2)$ is given in [5].

Let (r, b, X) denote the coloring of the star graph S_n on n vertices in which there are r red vertices $b = n - r$ blue vertices. The central vertex has colour $X \in \{R, B\}$. In the case of the push process, the transitions from state (r, b, R) are to state $(r+1, b-1, R)$ with probability $1/(b+1)$ and to state $(r-1, b+1, B)$ with probability $b/(b+1)$. The transitions from state $(r-1, b+1, B)$ are to (r, b, R) with probability $(r-1)/r$ and to $(r-2, b+2, B)$ with probability $1/r$. For the purposes of discussion we group the states $(r, R) = (r, b, R)$ and $(r-1, B) = (r-1, b+1, B)$ into a single pseudo-state $S(r)$.

The transitions probabilities within or between $S(r+1)$ or $S(r-1)$ are shown in Figure 3, and are derived as follows. Let $X, Y \in \{R, B\}$. For a particle occupying a state (of colour) X in $S(r)$ let $P_X(Y, r)$ be the probability of exit from $S(r)$ via state Y. For example $P_R(R, r)$ is the probability that a particle starting at (r, R) eventually exits from $S(r)$ via state (r, R) to state $(r+1, R)$ in $S(r+1)$. Thus

$$P_R(R, r) = \frac{1}{b+1}\left(1 + \frac{b}{b+1}\frac{r-1}{r} + \cdots + \left(\frac{b}{b+1}\frac{r-1}{r}\right)^k + \cdots\right),$$

so that

$$P_R(R, r) = \frac{1}{b+1}\frac{1}{1 - [b(r-1)/(b+1)r]} = \frac{r}{n}.$$

Similarly let $P_B(R, r)$ be the probability that a particle currently at $(r-1, B)$ in $S(r)$ moves from $S(r)$ to $(r+1, R)$ in $S(r+1)$. Then

$$P_B(R, r) = \frac{r-1}{r}P_R(R, r) = \frac{r-1}{n}.$$

In summary, starting from state $X \in \{R, B\}$ of $S(r)$, for $1 \le r \le n-1$ the transition probability $p_X(r)$ from $S(r)$ to $S(r+1)$ (resp. transition probability $p_X(b)$ from $S(r)$ to $S(r-1)$) is given by

$$p_X(r) = \frac{r - 1_{(X=B)}}{n}, \qquad p_X(b) = \frac{b + 1_{(X=B)}}{n}. \tag{13}$$

States $(0, B)$ (i.e. $S(0)$) and (n, R) (i.e. $S(n)$) are absorbing.

Let $i = \max(r, b) - n/2$. To obtain lower and upper bounds on the number of transitions *between pseudo-states* $S(r)$ *before absorption*, we can couple the process with a biassed

random walk on the line $L = \{0, 1, ..., n/2\}$ with a reflecting barrier at 0 and an absorbing barrier at $n/2$. We assume n is even here. For $0 < i < n/2$, let p_i be the probability of a transition from i to $i+1$ on L, and let $q_i = 1 - p_i$ be the probability of a transition from i to $i-1$. It follows from (13) that to obtain bounds on the number of transitions between pseudo-states $S(r)$ before absorption we can use a value of p_i given by

$$p_i = 1/2 + (i+1)/n \quad \text{Lower bound}, \qquad p_i = 1/2 + (i-1)/n \quad \text{Upper bound}. \tag{14}$$

We next consider the number of loops, for example $(r, R) \to (r-1, B) \to (r, R)$, made within $S(r)$ before exit. For a particle starting from state X of $S(r)$ let $C_{XY} = C_{XY}(r)$ be the number of loops before exit at state Y. Let $\lambda = \frac{b}{b+1}\frac{r-1}{r}$ and $\rho = \lambda/(1-\lambda)^2$, then

$$\mathbf{E}C_{RR} = \sum_{k\geq 0} \frac{1}{b+1} k\lambda^k = \frac{1}{b+1}\frac{\lambda}{(1-\lambda)^2} = \rho\frac{1}{b+1}.$$

Similarly,

$$\mathbf{E}C_{BR} = \rho\frac{r-1}{r(b+1)}, \qquad \mathbf{E}C_{RB} = \rho\frac{b}{r(b+1)}, \qquad \mathbf{E}C_{BB} = \rho\frac{1}{r}.$$

The conditional expectations $\mu_{XY}(r) = \mathbf{E}C_{XY}(r)/P_X(Y, r)$ are given by

$$\mu_{XY}(r) = \begin{cases} \rho\frac{n}{r}\frac{1}{b+1}, & XY = RR \\ \rho\frac{n}{r}\frac{1}{b+1}, & XY = BR \\ \rho\frac{n}{n-r}\frac{b}{r(b+1)}, & XY = RB \\ \rho\frac{n}{n-r+1}\frac{1}{r}, & XY = BB \end{cases}. \tag{15}$$

The value of $\rho = (rb(r-1)(b+1))/n^2$. In particular if $b, r = (1 + o(1))n/2$ then, whatever colours X, Y

$$\mu_{XY}(r) = (1 + o(1))\frac{n}{4}. \tag{16}$$

Let $N = n/2$. Starting from $r = b = N$ let T'_N be the number of transitions between states $S(r)$ to reach $\max(r, b) = N + n/2$. Referring to (14), we consider a biassed random walk with transition probabilities of $Z = \max\{r, b\} - n/2$ given by

$$p_i = \begin{cases} 1, & \text{if } i = 0 \\ 1/2 + i/n + \delta/n, & \text{if } i \in \{1, \ldots, n/2 - 1\} \\ 0, & \text{if } i = n/2 \end{cases}, \tag{17}$$

where we set $\delta = 1$ for a lower bound on the number of steps T' to absorption, and $\delta = -1$ for an upper bound. The walk in (17) is the push chain Z_t with transitions given by (5) as analysed Section 2. Referring to (5) and (4) we set $\delta = 0$ for a lower bound on $\mathbf{E}_0 T_M$. For $M = N^{3/4}$, from Lemma 4,

$$\mathbf{E}_0 T_M \geq \Theta(1) \sum_{i=\sqrt{N}}^{M} \frac{N}{i} \geq \Theta(N) \log \frac{M}{\sqrt{N}} = \Theta(n \log n).$$

For all states $i = \sqrt{N}, ..., N^{3/4}$, the corresponding value of $r = (1 + o(1))n/2$. Referring to (16), whatever the type of transition XY between $S(r)$ and neighbouring states, $\mu_{XY}(r) = (1 + o(1))n/4$. Let $\mu = \min_{X,Y}(\mu_{XY}(r) : n/2 \leq r \leq M)$, then $\mu \geq n/5$. As $\mathbf{E}_0 T_N \geq \mathbf{E}_0 T_M = \Theta(n \log n)$ we have that

$$\mathbf{E}T(Push) \geq \mu\, \mathbf{E}_0 T_M = \Omega(n^2 \log n).$$

The upper bound follows by a similar argument. Put $\delta = -1$ in (5), and use Lemma 4.

5.1 Discordant voting: Simulation results for star graph and cycle

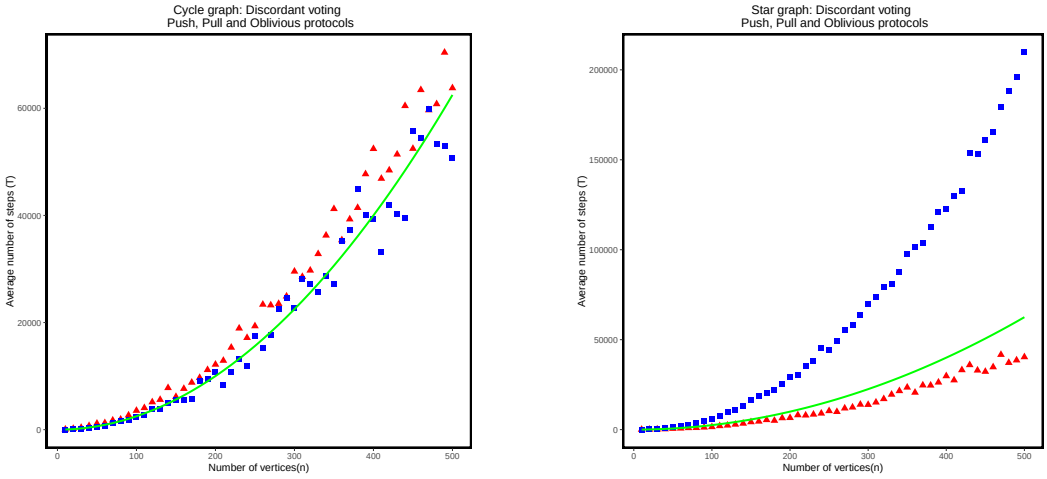


Figure 4 Legend: Push (Square), Pull (Triangle), Oblivious $\mathbf{E}T = n^2/4$ (Solid line). Left plot: Cycle, initial colouring alternating red-blue (see Fig.1(i)), Right plot: Star graph, random colouring $R(0) = B(0) = n/2$. Each plot point consists of at least 15 replications.

References

1 D. Aldous and J. Fill. Reversible markov chains and random walks on graphs, 2002. Unfinished monograph, recompiled 2014, available at `http://www.stat.berkeley.edu/~aldous/RWG/book.html`.

2 J. Aspnes and E. Ruppert. An introduction to population protocols. In *Middleware for Network Eccentric and Mobile Applications*, pages 97–120. Springer, 2009.

3 R. Basu and A. Sly. Evolving voter model on dense random graphs. *arXiv preprint. arXiv:1501.03134*, 2015.

4 S. Brahma, S. Macharla, S. Pal, and S. Singh. Fair leader election by randomized voting. In *Distributed Computing and Internet Technology*, pages 22–31. Springer, 2004.

5 C. Cooper, M. Dyer, A. Frieze, and N. Rivera. Discordant voting processes on finite graphs (full version). *arXiv preprint. arxiv.org/abs/1604.06884*, 2015.

6 C. Cooper, R. Elsasser, H. Ono, and T. Radzik. Coalescing random walks and voting on connected graphs. *SIAM Journal on Discrete Mathematics*, 27(4):1748–1758, 2013.

7 C. Cooper, A. Frieze, and T. Radzik. Multiple random walks in random regular graphs. *SIAM Journal on Discrete Mathematics*, 23(4):1738–1761, 2009.

8 R. Durrett, J. Gleeson, A. Lloyd, P. Mucha, F. Shi, D. Sivakoff, J. Socolar, and C. Varghese. Graph fission in an evolving voter model. *Proceedings of the National Academy of Sciences*, 109(10):3682–3687, 2012.

9 W. Feller. *An introduction to probability theory and its applications*, volume 2. John Wiley & Sons, 2008.

10 D. Gifford. Weighted voting for replicated data. In *Proceedings of the seventh ACM symposium on Operating systems principles*, pages 150–162. ACM, 1979.

11 T. Gross and H. Sayama. *Adaptive networks*. Springer, 2009.

12 Y. Hassin and D. Peleg. Distributed probabilistic polling and applications to proportionate agreement. *Information and Computation*, 171(2):248–268, 2001.

13 P. Holme and M. Newman. Nonequilibrium phase transition in the coevolution of networks and opinions. *Physical Review E*, 74(5):056108, 2006.

14 B. Johnson. *Design & analysis of fault tolerant digital systems*. Addison-Wesley Longman Publishing Co., Inc., 1988.

15 D. Levin, Y. Peres, and E. Wilmer. *Markov chains and mixing times*. AMS Bookstore, 2009.

16 E. Mossel and S. Roch. Slow emergence of cooperation for win-stay lose-shift on trees. *Machine Learning*, 67(1-2):7–22, 2007.

17 T. Nakata, H. Imahayashi, and M. Yamashita. Probabilistic local majority voting for the agreement problem on finite graphs. In *Computing and Combinatorics*, pages 330–338. Springer, 1999.

18 H. Sayama, I. Pestov, J. Schmidt, B. Bush, C. Wong, J. Yamanoi, and T. Gross. Modeling complex systems with adaptive networks. *Computers & Mathematics with Applications*, 65(10):1645–1664, 2013.

19 B. Sury. Sum of the reciprocals of the binomial coefficients. *European Journal of Combinatorics*, 14(4):351–353, 1993.

20 D. Williams. *Probability with martingales*. Cambridge University Press, 1991.

21 Deng X and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994.

Bounds on the Voter Model in Dynamic Networks*

Petra Berenbrink[1], George Giakkoupis[2], Anne-Marie Kermarrec[3], and Frederik Mallmann-Trenn[4]

1 **Simon Fraser University, Burnaby, Canada**
petra@sfu.ca
2 **INRIA, Rennes, France**
george.giakkoupis@inria.fr
3 **INRIA, Rennes, France**
anne-marie.kermarrec@inria.fr
4 **Simon Fraser University, Burnaby, Canada; and École Normale Supérieure, Paris, France**
fmallman@sfu.ca

Abstract

In the *voter model*, each node of a graph has an opinion, and in every round each node chooses independently a random neighbour and adopts its opinion. We are interested in the *consensus time*, which is the first point in time where all nodes have the same opinion. We consider dynamic graphs in which the edges are rewired in every round (by an adversary) giving rise to the graph sequence $G_1, G_2, \ldots$, where we assume that G_i has conductance at least ϕ_i. We assume that the degrees of nodes don't change over time as one can show that the consensus time can become super-exponential otherwise. In the case of a sequence of d-regular graphs, we obtain asymptotically tight results. Even for some static graphs, such as the cycle, our results improve the state of the art. Here we show that the expected number of rounds until all nodes have the same opinion is bounded by $O(m/(d_{min} \cdot \phi))$, for any graph with m edges, conductance ϕ, and degrees at least d_{min}. In addition, we consider a *biased* dynamic voter model, where each opinion i is associated with a probability P_i, and when a node chooses a neighbour with that opinion, it adopts opinion i with probability P_i (otherwise the node keeps its current opinion). We show for any regular dynamic graph, that if there is an $\epsilon > 0$ difference between the highest and second highest opinion probabilities, and at least $\Omega(\log n)$ nodes have initially the opinion with the highest probability, then all nodes adopt w.h.p. that opinion. We obtain a bound on the convergence time, which becomes $O(\log n/\phi)$ for static graphs.

1998 ACM Subject Classification C.2.2 Network Protocols

Keywords and phrases Voting, Distributed Computing, Conductance, Dynamic Graphs, Consensus

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.146

1 Introduction

In this paper, we investigate the spread of opinions in a connected and undirected graph using the *voter model*. The standard voter model works in synchronous rounds and is defined as follows. At the beginning, every node has one opinion from the set $\{0, \ldots, n-1\}$, and

* See https://arxiv.org/abs/1603.01895 for the full version.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 146; pp. 146:1–146:15

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

in every round, each node chooses one of its neighbours uniformly at random and adopts its opinion. In this model, one is usually interested in the *consensus time* and the *fixation probability*. The consensus time is the number of rounds it takes until all nodes have the same opinion. The fixation probability of opinion i is the probability that this opinion prevails, meaning that all other opinions vanish. This probability is known to be proportional to the sum of the degrees of the nodes starting with opinion i [13, 22].

The voter model is the dual of the *coalescing random walk model* which can be described as follows. Initially, there is a pebble on every node of the graph. In every round, every pebble chooses a neighbour uniformly at random and moves to that node. Whenever two or more pebbles meet at the same node, they are merged into a single pebble which continues performing a random walk. The process terminates when only one pebble remains. The time it takes until only one pebble remains is called *coalescing time*. It is known that the coalescing time for a graph G equals the consensus time of the voter model on G when initially each node has a distinct opinion [2, 19].

In this paper we consider the voter model and a *biased* variant where the opinions have different popularity. We express the consensus time as a function of the graph *conductance* ϕ.

We assume a dynamic graph model where the edges of the graph can be rewired by an adversary in every round, as long as the adversary respects the given degree sequence and the given conductance for all generated graphs. We show that consensus is reached with constant probability after τ rounds, where τ is the first round such that the sum of conductances up to round τ is at least m/d_{min}, where m is the number of edges. For static graphs the above bound simplifies to $O(m/(d_{min} \cdot \phi))$, where d_{min} is the minimum degree.

For the biased model we assume a *regular* dynamic graph G. Similar to [19, 16] the opinions have a *popularity*, which is expressed as a probability with which nodes adopt opinions. Again, every node chooses one of its neighbours uniformly at random, but this time it adopts the neighbour's opinion with a probability that equals the popularity of this opinion (otherwise the node keeps its current opinion). We assume that the popularity of the most popular opinion is 1, and every other opinion has a popularity of at most $1 - \epsilon$ (for an arbitrarily small but constant $\epsilon > 0$). We also assume that at least $\Omega(\log n)$ nodes start with the most popular opinion. Then we show that the most popular opinion prevails w.h.p.[1] after τ rounds, where τ is the first round such that the sum of conductances up to round τ is of order $O(\log n)$. For static graphs the above bound simplifies as follows: the most popular opinion prevails w.h.p. in $O(\log n/\phi)$ rounds, if at least $\Omega(\log n)$ nodes start with that opinion.

1.1 Related work

A sequential version of the voter model was introduced in [14] and can be described as follows. In every round, a single node is chosen uniformly at random and this node changes its opinion to that of a random neighbour. The authors of [14] study infinite grid graphs. This was generalised to arbitrary graphs in [9] where it is shown among other things that the probability for opinion i to prevail is proportional to the sum of the degrees of the nodes having opinion i at the beginning of the process.

The standard voter model was first analysed in [13]. The authors of [13] bound the expected coalescing time (and thus the expected consensus time) in terms of the expected meeting time t_{meet} of two random walks and show a bound of $O(t_{meet} \cdot \log n) = O(n^3 \log n)$.

[1] An event happens *with high probability (w.h.p.)* if its probability is at least $1 - 1/n$.

Note that the meeting time is an obvious lower bound on the coalescing time, and thus a lower bound on the consensus time when all nodes have distinct opinions initially. The authors of [4] provide an improved upper bound of $O\big(\frac{1}{1-\lambda_2}(\log^4 n + \rho)\big)$ on the expected coalescing time for any graph G, where λ_2 is the second eigenvalue of the transition matrix of a random walk on G, and $\rho = \big(\sum_{u\in V(G)} d(u)\big)^2 / \sum_{u\in V(G)} d^2(u)$ is the ratio of the square of the sum of node degrees over the sum of the squared degrees. The value of ρ ranges from $\Theta(1)$, for the star graph, to n, for regular graphs.

The authors of [20, 2, 19] investigate coalescing random walks in a continuous setting where the movement of the pebbles are modelled by independent Poisson processes with a rate of 1. In [2], it is shown a lower bound of $\Omega(m/d_{max})$ and an upper bound of $O(t_{hit}\cdot\log n)$ for the expected coalescing time. Here m is the number of edges in the graph, d_{max} is the maximum degree, and t_{hit} is the (expected) hitting time. In [23], it is shown that the expected coalescing time is bounded by $O(t_{hit})$.

In[19] the authors consider the biased voter model in the continuous setting and two opinions. They show that for d-dimensional lattices the probability for the less popular opinion to prevail is exponentially small. In [16], it is shown that in this setting the expected consensus time is exponential for the line.

The authors of [5] consider a modification of the standard voter model with two opinions, which they call *two-sample voting*. In every round, each node chooses two of its neighbours randomly and adopts their opinion only if they both agree. For regular graphs and random regular graphs, it is shown that two-sample voting has a consensus time of $O(\log n)$ if the initial imbalance between the nodes having the two opinions is large enough. There are several other works on the setting where every node contacts in every round two or more neighbours before adapting its opinion [1, 7, 6, 10].

There are several other models which are related to the voter model, most notably the *Moran process* and *rumor spreading* in the phone call model. In the case of the Moran process, a population resides on the vertices of a graph. The initial population consists of one mutant with fitness r and the rest of the nodes are non-mutants with fitness 1. In every round, a node is chosen at random with probability proportional to its fitness. This node then reproduces by placing a copy of itself on a randomly chosen neighbour, replacing the individual that was there. The main quantities of interest are the probability that the mutant occupies the whole graph (fixation) or vanishes (extinction), together with the time before either of the two states is reached (absorption time). There are several publications considering the fixation probabilities [15, 21, 8].

Rumor spreading in the phone call model works as follows. Every node v opens a channel to a randomly chosen neighbour u. The channel can be used for transmissions in both directions. A transmission from v to u is called *push* transmission and a transmission from u to v is called *pull*. There is a vast amount of papers analysing rumor spreading on different graphs. The result that is most relevant to ours is that broadcasting of a message in the whole network is completed in $O(\log n/\phi)$ rounds w.h.p, where ϕ is the conductance (see Section 1.2 for a definition) of the network. In [12], the authors study rumor spreading in dynamic networks, where the edges in every round are distributed by an adaptive adversary. They show that broadcasting terminates w.h.p. in a round t if the sum of conductances up to round t is of order $\log n$. Here, the sequence of graphs $G_1, G_2, \ldots$ have the same vertex set of size n, but possibly distinct edge sets. The authors assume that the degrees and the conductance may change over time. We refer the reader to the next section for a discussion of the differences. Dynamic graphs have received ample attention in various areas [3, 17, 24, 18].

1.2 Model and New Results

In this paper we show results for the standard voter model and biased voter model in dynamic graphs. Our protocols work in synchronous steps. The consensus time T is defined at the first time step at which all nodes have the same opinion.

Standard Voter Model

Our first result concerns the standard voter model in dynamic graphs. Our protocol works as follows. In every synchronous time step every node chooses a neighbour u.a.r. and adopts its opinion with probability $1/2$.[2]

We assume that the dynamic graphs $\mathscr{G} = G_1, G_2, \ldots$ are generated by an adversary. We assume that each graph has n nodes and the nodes are numbered from 1 to n. The sequence of conductances $\phi_1, \phi_2, \ldots$ is given in advance, as well as a degree sequence $d_1, d_2, \ldots, d_n$. The adversary is now allowed to create every graph G_i by redistributing the edges of the graph. The constraints are that each graph G_i has to have conductance ϕ_i and node j has to have degree d_j (the degrees of the nodes do not change over time). Note that the sequence of the conductances is fixed and, hence, cannot be regarded as a random variable in the following. For the redistribution of the edges we assume that the adversary knows the distribution of all opinions during all previous rounds.

Note that our model for dynamic graphs is motivated by the model presented in [12]. They allow the adversary to determine the edge set at every round, without having to respect the node degrees and conductances.

We show (Observation 1) that, allowing the adversary to change the node degrees over time can results in super-exponential voting time. Since this changes the behaviour significantly, we assume that the degrees of nodes are fixed. Furthermore, in contrary to [12], we assume that (bounds on) the conductance of (the graph at any time step) are fixed/given beforehand. Whether one can obtain the same results, if the conductance of the graph is determined by an adaptive adversary remains an open question. The reason we consider an adversarial dynamic graph model is in order to understand how the voting time can be influenced in the worst-case. Another interesting model would be to assume that in every round the nodes are connected to random neighbours. One obstacle to such a model seems to be to guarantee that neighbours are chosen u.a.r. and the degrees of nodes do not change. For the case of regular random dynamic graphs our techniques easily carry over since the graph will have constant conductance w.h.p. in any such round since the graph is essentially a random regular graph in every round.

For the (adversarial) dynamic model we show the following result bounding the consensus time T.

▶ **Theorem 1** (upper bound). *Consider the Standard Voter model and in the dynamic graph model. Assume $\kappa \leq n$ opinions are arbitrarily distributed over the nodes of G_1. Let ϕ_t be a lower bound on the conductance at time step t. Let $b > 0$ be a suitable chosen constant. Then, with a probability of $1/2$ we have that $T \leq \min\{\tau, \tau'\}$, where*

(i) *τ is the first round so that $\sum_{t=1}^{\tau} \phi_t \geq b \cdot m/d_{min}$.* (part 1)

(ii) *τ' is the first round so that $\sum_{t=1}^{\tau'} \phi_t^2 \geq b \cdot n \log n$.* (part 2)

For static graphs ($G_{i+1} = G_i$ for all i), we have $T \leq \min\{m/(d_{min} \cdot \phi), n \log n/\phi^2\}$.

[2] The factor of $1/2$ ensures that the process converges on bipartite graphs.

For *static* d-regular graphs, where the graph doesn't change over time, the above bound becomes $O(n/\phi)$, which is tight when either ϕ or d are constants (see Observation 2). Theorem 1 gives the first tight bounds for cycles and circulant graphs C_n^k (node i is adjacent to the nodes $i \pm 1, \ldots, i \pm k \mod n$) with degree $2k$ (k constant). For these graphs the consensus time is $\Theta(n^2)$, which matches our upper bound from Theorem 1.[3] For a comparison with the results of [4] note that $\phi^2 \leq 1 - \lambda_2 \leq 2\phi$. In particular, for the cycle $\phi = 1/n$ and $1/(1-\lambda_2) = \Theta(1/n^2)$. Hence, for this graph, our bound is by a factor of n smaller. Note that, due to the duality between the voter model and coalescing random walks, the result also holds for the coalescing time. In contrast to [4, 5], the above result is shown using a potential function argument, whereas the authors of [4, 5] show their results for coalescing random walks and fixed graphs. The advantage of analysing the process directly is, that our techniques allow us to obtain the results for the dynamic setting.

The next result shows that the bound of Theorem 1 is asymptotically tight if the adversary is allowed to change the node degrees over time.

▶ **Theorem 2** (lower bound). *Consider the Standard Voter model in the dynamic graph model. Assume that $\kappa \leq n$ opinions are arbitrarily distributed over the nodes of G_1. Let ϕ_t be an upper bound on the conductance at time step t. Let $b > 0$ be a suitable constant and assume τ'' is the first round such that $\sum_{t=1}^{\tau''} \phi_t \geq bn$. Then, with a probability of at least $1/2$, there are still nodes with different opinions in $G_{\tau''}$.*

Biased Voter Model

In the *biased* voter model we again assume that there are $\kappa \leq n$ distinct opinions initially. For $0 \leq i \leq \kappa - 1$, opinion i has popularity α_i and we assume that $\alpha_0 = 1 > \alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_{\kappa-1}$. We call opinion 0 the *preferred opinion*. The process works as follows. In every round, every node chooses a neighbour uniformly at random and adopts its opinion i with probability α_i.

We assume that the dynamic d-regular graphs $\mathscr{G} = G_1, G_2, \ldots$ are generated by an adversary. We assume that the sequence of ϕ_t is given in advance, where ϕ_i is a lower bound on the conductance of G_i. The adversary is now allowed to create the sequence of graphs by redistributing the edges of the graph in every step. The constraints are that each graph G_i has n nodes and has to have conductance at least ϕ_i. Note that we assume that the sequence of the conductances is fixed and, hence, it is not a random variable in the following.

The following result shows that consensus is reached considerably faster in the biased voter model, as long as the bias $1 - \alpha_1$ is bounded away from 0, and at least a logarithmic number of nodes have the preferred opinion initially.

▶ **Theorem 3.** *Consider the Biased Voter model in the dynamic regular graph model. Assume $\kappa \leq n$ opinions are arbitrarily distributed over the nodes of G_1. Let ϕ_t be a lower bound on the conductance at time step t. Assume that $\alpha_1 \leq 1 - \epsilon$, for an arbitrary small constant $\epsilon > 0$. Assume the initial number of nodes with the preferred opinion is at least $c \log n$, for some constant $c = c(\alpha_1)$. Then the preferred opinion prevails w.h.p. in at most τ''' steps, where τ''' is the first round so that $\sum_{t=1}^{\tau'''} \phi_t \geq b \log n$, for some constant b. For static graphs ($G_{i+1} = G_i$ for all i), we have w.h.p. $T = O(\log n/\phi)$.*

[3] The lower bound of $\Omega(n^2)$ follows from the fact that two coalescing random walks starting on opposite sites of a cycle require in expectation time $\Omega(n^2)$ to meet.

The assumption on the initial size of the preferred opinion is crucial for the time bound $T = O(\log n/\phi)$, in the sense that there are instances where the expected consensus time is at least $T = \Omega(n/\phi)$ if the size of the preferred opinion is small.[4]

The rumor spreading process can be viewed as an instance of the biased voter model with two opinions having popularity 1 and 0, respectively. However, the techniques used for the analysis of rumor spreading do not extend to the voter model. This is due to the fact that rumor spreading is a progressive process, where nodes can change their opinion only once, from "uninformed" to "informed", whereas they can change their opinions over and over again in the case of the voter model. Note that the above bound is the same as the bound for rumor spreading of [11] (although the latter bound holds for general graphs, rather than just for regular ones). Hence, our above bound is tight for regular graphs with conductance ϕ, since the rumor spreading lower bound of $\Omega(\log n/\phi)$ is also a lower bound for biased voting in our model.

2 Analysis of the Voter Model

In this section we show the upper and lower bound for the standard voter model. We begin with some definitions. Let $G = (V, E)$. For a fixed set $S \subseteq V$ we define $cut(S, V \setminus S)$ to be the set of edges between the sets $S \subseteq V$ and $V \setminus S$ and let λ_u be the number of neighbours of u in $V \setminus S$. Let $\mathrm{vol}(S) = \sum_{u \in S} d_u$. The *conductance* of G is defined as

$$\phi = \phi(G) = \min \left\{ \frac{\sum_{u \in U} \lambda_u}{\mathrm{vol}(U)} : U \subset V \text{ with } 0 < \mathrm{vol}(U) \leq m \right\}.$$

We note $1/n^2 \leq \phi \leq 1$. We denote by $v_t^{(i)}$ the set of nodes that have opinion i after the first t rounds and $t \geq 0$. If we refer to the random variable we use $V_t^{(i)}$ instead.

First we show Theorem 1 for $\kappa = 2$ (two opinions), which we call 0 and 1 in the following. Then we generalise the result to an arbitrary number of opinions. We model the system with a Markov chain $M_{t \geq 0} = (V_t^{(0)}, V_t^{(1)})_{t \geq 0}$.

Let s_t denote the set having the smaller volume, i.e., $s_t = v_t^{(0)}$ if $\mathrm{vol}(v_t^{(0)}) \leq \mathrm{vol}(v_t^{(1)})$, and $s_t = v_t^{(1)}$ otherwise. Note that we use $s_t, v_t^{(0)}$ and $v_t^{(1)}$ whenever the state at time t is fixed, and $S_t, V_t^{(0)}$ and $V_t^{(1)}$ for the corresponding random variables. For $u \in v_t^{(0)}$, $\lambda_{u,t}$ is the number of neighbours of u in $V \setminus v^1(t)$ and for $u \in v_t^{(1)}$, $\lambda_{u,t}$ is the number of neighbours of u in $V \setminus v_t^{(0)}$; d_u is the degree of u (the degrees do not change over time).

To analyse the process we use a potential function. Simply using the volume of nodes sharing the same opinion as the potential function will not work. It is easy to calculate that the expected volume of nodes with a given opinion does not change in one step. Instead, we use a convex function on the number of nodes with the minority opinion. We define

$$\Psi(S_t) = \sqrt{\mathrm{vol}(S_t)}.$$

In Lemma 4 we first calculate the one-step potential drop of $\Psi(S_t)$. Then we show that every opinion either prevails or vanishes once the sum of conductances is proportional to the volume of nodes having that opinion (see Lemma 5), which we use later to prove Part 1 of Theorem 1. Additionally, Lemma 4 is used to prove Lemma 7 which allows us to prove Part 2 of Theorem 1.

[4] Consider a 3-regular graph and n opinions where all other $\alpha_1 = \alpha_2 = \cdots = \alpha_{n-1} = 1/2$. The preferred opinion vanishes with constant probability and the bound for the standard voter model of Observation 2 applies.

▶ **Lemma 4.** *Assume $s_t \neq \emptyset$ and $\kappa = 2$. Then*

$$\mathbf{E}[\Psi(S_{t+1}) \mid S_t = s_t] \leq \Psi(s_t) - \frac{\sum_{u \in V} \lambda_{u,t} \cdot d_u}{32 \cdot (\Psi(s_t))^3}.$$

Proof. W.l.o.g. we assume that opinion 0 is the minority opinion, i.e. $0 < \mathrm{vol}(V_t^{(0)}) \leq \mathrm{vol}(V_t^{(1)})$. To simplify the notation we omit the index t in this proof and write $v^{(0)}$ instead of $v_t(0)$, $v^{(1)}$ for $V \setminus v_t^{(0)}$, and λ_u instead of $\lambda_{u,t}$. Hence, $s_t = v^{(0)}$ and $\Psi(s_t) = \sqrt{\mathrm{vol}(v^{(0)})}$. Note that for $t = 0$ we have $\mathrm{vol}(v^{(0)}) = \Psi(s_t)^2$. Furthermore, we fix $S_t = s_t$ in the following (and condition on it). We define m as the number of edges. Then we have

$$\begin{aligned}
\mathbf{E}[\Psi(S_{t+1}) - \Psi(s_t) \mid S_t = s_t] &= \mathbf{E}[\sqrt{\mathrm{vol}(S_{t+1})} - \sqrt{\mathrm{vol}(s_t)}] \\
&= \mathbf{E}\left[\sqrt{\min\left\{\mathrm{vol}(V_{t+1}^{(0)}), m - \mathrm{vol}(V_{t+1}^{(0)})\right\}} - \sqrt{\mathrm{vol}(s_t)}\right] \\
&\leq \mathbf{E}\left[\sqrt{\mathrm{vol}(V_{t+1}^{(0)})} - \sqrt{\mathrm{vol}(v^{(0)})}\right] \qquad (1)
\end{aligned}$$

Now we define

$$X_u = \begin{cases} d_u & w.p.\ \frac{\lambda_u}{2 \cdot d_u}\ if\ u \in v^{(1)} \\ -d_u & w.p.\ \frac{\lambda_u}{2 \cdot d_u}\ if\ u \in v^{(0)} \\ 0 & otherwise \end{cases}$$

and $\Delta = \sum_{u \in V} X_u$. Note that we have $\Delta = \mathrm{vol}(V_{t+1}^{(0)}) - \mathrm{vol}(v^{(0)})$ and

$$\begin{aligned}
\mathbf{E}\left[\sqrt{\mathrm{vol}(V_{t+1}^{(0)})} - \sqrt{\mathrm{vol}(v^{(0)})}\right] &= \mathbf{E}\left[\sqrt{\mathrm{vol}(v^{(0)}) + \Delta} - \sqrt{\mathrm{vol}(v^{(0)})}\right] \\
&= \mathbf{E}\left[\sqrt{\mathrm{vol}(v^{(0)})}\left(\sqrt{1 + \tfrac{\Delta}{\mathrm{vol}(v^{(0)})}} - 1\right)\right] \\
&= \Psi(s_t) \cdot \mathbf{E}[\sqrt{1 + \Delta/\Psi(s_t)^2} - 1].
\end{aligned}$$

Unfortunately we cannot bound $\Psi(s_t) \cdot \mathbf{E}[\sqrt{1 + \Delta/\Psi(s_t)^2} - 1]$ directly. Instead, we define a family of random variables which is closely related to X_u.

$$Y_u = \begin{cases} \lambda_u & w.p.\ \frac{1}{2} \quad if\ u \in v^{(1)} \\ -d_u & w.p.\ \frac{\lambda_u}{2 \cdot d_u} \quad if\ u \in v^{(0)} \\ 0 & otherwise \end{cases}$$

Similarly, we define $\Delta' = \sum_{u \in V} Y(u)$. Note that $|E[Y_u]| = \lambda_u/2$ for both $u \in v^{(1)}$ and $u \in v^{(0)}$. In the full version we show that $\mathbf{E}[\sqrt{1 + \Delta/\Psi(s_t)^2}] \leq \mathbf{E}[\sqrt{1 + \Delta'/\Psi(s_t)^2}]$, which results in $\mathbf{E}[\Psi(S_{t+1}) - \Psi(s_t) \mid S_t = s_t] \leq \Psi(s_t) \cdot \mathbf{E}[\sqrt{1 + \Delta'/\Psi(s_t)^2} - 1]$ From the Taylor expansion $\sqrt{1+x} \leq 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{x^3}{16}$, $x \geq -1$ it follows that

$$\mathbf{E}[\Psi(S_{t+1}) - \Psi(s_t) \mid S_t = s_t] \quad \leq \quad \Psi(s_t) \cdot \mathbf{E}\left[\tfrac{\Delta'}{2\Psi(s_t)^2} - \tfrac{(\Delta')^2}{8\Psi(s_t)^4} + \tfrac{(\Delta')^3}{16\Psi(s_t)^6}\right].$$

It remains to bound $\mathbf{E}[\Delta']$, $\mathbf{E}[(\Delta')^2]$, and $\mathbf{E}[(\Delta')^3]$.

- $\mathbf{E}[\Delta']$: We have $\mathbf{E}[\Delta'] = \sum_{u \in V} E[Y_u] = \sum_{u \in v^{(1)}} \frac{\lambda_u}{2} - \sum_{v \in v^{(0)}} \frac{\lambda_v}{2} = 0$, where the last equality holds since $\sum_{u \in v^{(1)}} \lambda_u$ and $\sum_{u \in v^{(1)}} \lambda_u$ both count the number of edges crossing the cut between $v^{(0)}$ and $v^{(1)}$.

- $\mathbf{E}[(\Delta')^2]$: since $E[(Y_u)^2] = (\lambda_u)^2/2$ for $u \in v^{(1)}$ and $E[(Y_u)^2] = -d_u \cdot \lambda_u/2$ for $u \in v^{(0)}$ we have

$$\begin{aligned}\mathbf{E}[(\Delta')^2] &= \sum_{u\in V} \mathbf{Var}[Y_u] + (\mathbf{E}[Y_u])^2 = \sum_{u\in V} \mathbf{Var}[Y_u] + 0 = \sum_{u\in V} (\mathbf{E}[(Y_u)^2] - (\mathbf{E}[Y_u])^2) \\ &= \sum_{u\in v^{(0)}} (\mathbf{E}[(Y_u)^2] - (\mathbf{E}[Y_u])^2) + \sum_{u\in v^{(1)}} (\mathbf{E}[(Y_u)^2] - (\mathbf{E}[Y_u])^2) \\ &= \sum_{u\in v^{(0)}} \frac{\lambda_u d_u}{2} - \sum_{u\in v^{(0)}} \frac{\lambda_u^2}{4} + \sum_{u\in v^{(1)}} \frac{\lambda_u^2}{4} \geq \sum_{u\in v^{(0)}} \frac{\lambda_u d_u}{4}. \end{aligned} \tag{2}$$

- $\mathbf{E}[\Delta'^3]$: In the full version we show that $\mathbf{E}[\Delta'^3] = \sum_{u\in V} \left(\mathbf{E}[(Y_u)^3] - 3\,\mathbf{E}[(Y_u)^2] \cdot \mathbf{E}[Y_u] + 2\,\mathbf{E}[Y_u]^3\right)$. Note that $E[(Y_u)^3] = \frac{1}{2}(\lambda_u)^3$ for $u \in v^{(1)}$ and $E[(Y_u)^3] = -\frac{1}{2}\lambda_u \cdot (d_u)^2$ for $u \in v^{(0)}$. Hence,

$$\begin{aligned}\mathbf{E}[\Delta'^3] = &\sum_{u\in v^{(0)}} \left(-\frac{1}{2}\lambda_u \cdot (d_u)^2 + \frac{3}{4}(\lambda_u)^2 \cdot d_u - \frac{1}{4}\lambda_u^3\right) \\ &+ \sum_{u\in v^{(1)}} \left(\frac{1}{2}(\lambda_u)^3 - \frac{3}{4}(\lambda_u)^3 + \frac{1}{4}(\lambda_u)^3\right) \leq 0, \end{aligned} \tag{3}$$

  where the first sum is bounded by 0 because $\lambda_u \leq d_u$.

  Combining all the above estimations we get

$$\begin{aligned}\mathbf{E}[\Psi(S_{t+1}) - \Psi(s_t) \mid S_t = s_t] &\leq \Psi(s_t) \cdot \mathbf{E}\left[\frac{\Delta'}{2\Psi(s_t)^2} - \frac{\Delta'^2}{8\Psi(s_t)^4} + \frac{\Delta'^3}{16\Psi(s_t)^6}\right] \\ &\leq -\frac{\sum_{u\in v^{(0)}} \lambda_u d_u}{32\Psi(s_t)^3}.\end{aligned}$$

This completes the proof of Lemma 4. ◀

2.1 Part 1 of Theorem 1.

Using Lemma 4 we show that a given opinion either prevails or vanishes with constant probability as soon as the sum of ϕ_t is proportional to the volume of the nodes having that opinion.

▶ **Lemma 5.** *Assume that $s_{\hat{t}}$ is fixed for an arbitrary $(\hat{t} \geq 0)$ and $\kappa = 2$.*
Let $\tau^* = \min\left\{t' : \sum_{i=\hat{t}}^{t'} \phi_i \geq 129 \cdot vol(s_{\hat{t}})/d_{min}\right\}$. Then $\Pr\left(T \leq \tau^* + \hat{t}\right) \geq 1/2$.

In particular, if the graph is static with conductance ϕ, then $\Pr\left(T \leq \frac{129 \cdot vol(s_{\hat{t}})}{\phi \cdot d_{min}} + \hat{t}\right) \geq 1/2$.

Proof. From the definition of $\Psi(s_t)$ and ϕ_t it follows for all t that $\Psi(s_t)^2 = \sum_{u\in v^{(0)}} d_u = \mathrm{vol}(v^{(0)})$ and $\phi_t \leq \sum_{u\in v^{(0)}} \lambda_{u,t}/\mathrm{vol}(v^{(0)})$. Hence, $\Psi(s_t)^2 \cdot \phi_t \cdot d_{min} \leq \sum_{u\in v^{(0)}} \lambda_{u,t} \cdot d_u$. Together with Lemma 4 we derive for $s_t \neq \emptyset$

$$\mathbf{E}[\Psi(S_{t+1}) \mid S_t = s_t] \leq \Psi(s_t) - \frac{\sum_{u\in V} \lambda_{u,t} \cdot d_u}{32 \cdot (\Psi(s_t))^3} \leq \Psi(s_t) - \frac{d_{min} \cdot \phi_t}{32 \cdot \Psi(s_t)}. \tag{4}$$

Recall that $T = \min_t\{S_t = \emptyset\}$. In the following we use the expression $T > t$ to denote the event $s_t \neq \emptyset$. Using the law of total probability we get

$$\mathbf{E}[\Psi(S_{t+1}) | T > t] = \mathbf{E}\left[\Psi(S_t) - \frac{d_{min} \cdot \phi_t}{32 \cdot \Psi(S_t)} \Big| T > t\right]$$

and using Jensen's inequality we get

$$\begin{aligned}\mathbf{E}[\Psi(S_{t+1}) \mid T > t] &= \mathbf{E}[\Psi(S_t) \mid T > t] - \mathbf{E}\left[\frac{d_{min} \cdot \phi_t}{32 \cdot \Psi(S_t)} \mid T > t\right] \\ &\leq \mathbf{E}[\Psi(S_t) \mid T > t] - \frac{d_{min} \cdot \phi_t}{32 \cdot \mathbf{E}[\Psi(S_t) \cdot \mid T > t]}.\end{aligned}$$

Since $\mathbf{E}[\Psi(S_t) \mid T \leq t] = 0$ we have

$$\begin{aligned}\mathbf{E}[\Psi(S_t)] &= \mathbf{E}[\Psi(S_t) \mid T > t] \cdot \Pr[T > t] + \mathbf{E}[\Psi(S_t) \mid T \leq t] \cdot \Pr[T \leq t] \\ &= \mathbf{E}[\Psi(S_t) \mid T > t] \cdot \Pr[T > t] + 0.\end{aligned}$$

Hence,

$$\frac{\mathbf{E}[\Psi(S_{t+1})]}{\Pr(T > t)} \leq \frac{\mathbf{E}[\Psi(S_t)]}{\Pr(T > t)} - \frac{d_{min} \cdot \phi_t \cdot \Pr(T > t)}{32\,\mathbf{E}[\Psi(S_t)]}$$

and

$$\mathbf{E}[\Psi(S_{t+1})] \leq \mathbf{E}[\Psi(S_t)] - \frac{d_{min} \cdot \phi_t \cdot (\Pr(T > t))^2}{32\,\mathbf{E}[\Psi(S_t)]}.$$

Let $t^* = \min\{t\colon \Pr(T > t) < 1/2\}$. In the following we use contradiction to show

$$t^* \leq \max\{t : \sum_{\hat{t} \leq t < t^*} \phi_t \leq 128 \cdot \mathrm{vol}(s_{\hat{t}})/d_{min}\}.$$

Assume the inequality is not satisfied. With $t = t^* - 1$ we get

$$\mathbf{E}[\Psi(S_{t^*})] \leq \mathbf{E}[\Psi(S_{t^*-1})] - \frac{d_{min} \cdot \phi_t \cdot (\Pr(T > t^* - 1))^2}{32\,\mathbf{E}[\Psi(S_{t^*-1})]} \leq \mathbf{E}[\Psi(S_{t^*-1})] - \frac{d_{min} \cdot \phi_{t^*} \cdot (1/4)}{32\,\mathbf{E}[\Psi(S_{t^*-1})]}.$$

Applying this equation iteratively, we obtain

$$\mathbf{E}[\Psi(S_{t^*})] \leq \mathbf{E}[\Psi(S_{\hat{t}})] - \sum_{\hat{t} \leq t < t^*} \frac{d_{min} \cdot \phi_t \cdot 1/4}{32\,\mathbf{E}[\Psi(S_t)]} \leq \mathbf{E}[\Psi(S_{\hat{t}})] - \frac{d_{min} \cdot \sum_{\hat{t} \leq t < t^*} \phi_t}{128\,\mathbf{E}[\Psi(S_{\hat{t}})]}. \tag{5}$$

Using the definition of $\mathbf{E}[\Psi(S_{\hat{t}})] = \sqrt{\mathrm{vol}(s_{\hat{t}})}$ and the definition of t^* we get

$$\mathbf{E}[\Psi(S_{t^*})] < \sqrt{\mathrm{vol}(s_{\hat{t}})} - \frac{d_{min} \cdot 128 \cdot \mathrm{vol}(s_{\hat{t}})}{128 \cdot d_{min} \cdot \sqrt{\mathrm{vol}(s_{\hat{t}})}} = \sqrt{\mathrm{vol}(s_{\hat{t}})} - \frac{\mathrm{vol}(s_{\hat{t}})}{\sqrt{\mathrm{vol}(s_{\hat{t}})}} = 0.$$

This is a contradiction since $\mathbf{E}[\Psi(S_{t^*})]$ is non-negative.

From the definition of t^*, we obtain $\Pr\left(T > \tau^* + \hat{t}\right) < 1/2$, completing the proof of Lemma 5. ◀

Now we are ready to show the first part of the theorem.

Proof of Part 1 of Theorem 1. We divide the τ rounds into phases. Phase i starts at time $\tau_i = \min\{t : \sum_{j=1}^{t} \phi_j \geq 2i\}$ for $i \geq 0$ and ends at $\tau_{i+1} - 1$. Since $\phi_j \leq 1$ for all $j \geq 0$ we have $\tau_0 < \tau_1 < \ldots$ and $\sum_{j=\tau_i}^{\tau_{i+1}} \phi_j \geq 1$ for $i \geq 0$. Let ℓ_t be the number of distinct opinions at the beginning of phase t. Hence, $\ell_0 = \kappa$.

We show in Lemma 6 below that the expected number of phases before the number of opinions drops by a factor of 5/6 is bounded by $6c \cdot \mathrm{vol}(V)/(\ell_t \cdot d_{min})$. For $i \geq 1$ let T_i be the number of phases needed so that the number of opinions drops to $(5/6)^i \cdot \ell_0$. Then only

one opinion remains after $\log_{6/5}\kappa$ many of these meta-phases. Then, for a suitably chosen constant b,

$$\mathbf{E}[T] \quad = \quad \sum_{j=1}^{\log_{6/5}\kappa} E[T_j] \leq \sum_{j=1}^{\log_{6/5}\kappa - 1} \frac{6c \cdot \mathrm{vol}(V)}{\ell_j \cdot d_{min}} \leq \sum_{j=1}^{\log_{6/5}\kappa} \frac{6c\,\mathrm{vol}(V)}{(5/6)^j \cdot \ell_0 \cdot d_{min}} = \frac{b \cdot m}{4 \cdot d_{min}}.$$

By Markov inequality, consensus is reached w.p. at least $1/2$ after $b \cdot m/(2d_{min})$ phases. By definition of τ and the definition of the phases, we have that the number of phases up to time step τ is at least $b \cdot m/(2d_{min})$. Thus, consensus is reached w.p. at least $1/2$ after τ time steps, which finishes the proof. ◀

▶ **Lemma 6.** *Fix a phase t and assume $c = 129$ and $\ell_t > 1$. The expected number of phases before the number of opinions drops to $5/6 \cdot \ell_t$ is bounded by $6c \cdot vol(V)/(\ell_t \cdot d_{min})$.*

Proof. Consider a point when there are ℓ' opinions left, with $5/6 \cdot \ell < \ell' \leq \ell$. Among those ℓ' opinions, there are at least $\ell' - \ell/3$ opinions i such that the volume of nodes with opinion i is at most $3 \cdot \mathrm{vol}(V)/\ell$. Let S denote the set of these opinions and let Z_i be an indicator variable which is 1 if opinion $i \in S$ vanished after $s = 3c \cdot \mathrm{vol}(V)/(\ell \cdot d_{min})$ phases and $Z_i = 0$ if it prevails. To estimate Z_i we consider the process where we have two opinions only. All nodes with opinion i retain their opinion and all other nodes have opinion 0. It is easy to see that in both processes the set of nodes with opinion i remains exactly the same. Hence, we can use Lemma 5 to show that with probability at least $1/2$, after s phases opinion i either vanishes or prevails. Hence,

$$\mathbf{E}\left[\Sigma_{j \in S} Z_j\right] = \Sigma_{j \in S}\, \mathbf{E}[Z_j] \geq |S|/2 \geq (\ell' - \ell/3)/2.$$

Using Markov's inequality we get that with probability $1/2$ at least $(\ell' - \ell/3)/4$ opinions vanish within s phases, and the number of opinions remaining is at most $\ell' - (\ell' - \ell/3)/4 = 3/4 \cdot \ell' + \ell/12 \leq 5/6 \cdot \ell$. The expected number of phases until $5/6 \cdot \ell$ opinions can be bounded by $\sum_{i=1}^{\infty} 2^{-i} \cdot s \leq 2s = \frac{6c \cdot \mathrm{vol}(V)}{\ell \cdot d_{min}}$. ◀

2.2 Part 2 of Theorem 1

We first bound the expected potential drop in round $t+1$, i.e., we bound $\mathbf{E}[\Psi(S_{t+1}) - \Psi(s_t) \mid S_t = s_t]$. This time however, we express the drop as a function which is linear in $\Psi(s_t)$. This allows us to bound the expected size of the potential at time τ', i.e., $\mathbf{E}[\Psi(S_{\tau'})]$, directly. From the expected size of the potential at time τ' we derive the desired bound on $Pr\,(T \leq \tau')$. The proof can be found in the full version.

▶ **Lemma 7.** *Assume $\kappa = 2$. We have $\Pr(T \leq \tau') \geq 1/n^2$. In particular, if the graph is static with conductance ϕ, then $\Pr\left(T \leq \frac{96 \cdot n \log n}{\phi^2}\right) \geq 1 - 1/n^2$.*

We now prove Part 2 of Theorem 1 which generalises to $\kappa > 2$.

Proof of Part 2 of Theorem 1. We define a parameterized version of the consensus time T. We define $T(\kappa) = \min\{t \colon \Psi(S_t) = 0 : \text{the number of different opinions at time } t \text{ is } \kappa\}$ for $\kappa \leq n$. We want to show that $\Pr(T(\kappa) \leq \tau') \geq 1 - 1/n$. From Lemma 7 we have that, that $\Pr(T(2) \leq \tau') \geq 1 - 1/n^2$. We define the 0/1 random variable Z_i to be one if opinion i vanishes or is the only remaining opinion after τ' rounds and $Z_i = 0$ otherwise. We have that $\Pr(Z_i = 1) \geq 1 - 1/n^2$ for all $i \leq \kappa$. We derive $\Pr(T(\kappa) \leq \tau') = \Pr(\wedge_{i \leq \kappa} Z_i) \geq 1 - 1/n$, by union bound. This yields the claim. ◀

2.3 Lower Bounds

In this section, we give the intuition behind the proof of Theorem 2 and state two additional observations. Recall that Theorem 2 states that our bound for regular graphs is tight for the adaptive adversary, even for $k = 2$. The first observation shows that the expected consensus time can be super-exponential if the adversary is allowed to change the degree sequence. The second observation can be regarded as a (weaker) counter part of Theorem 2 showing a lower bound of $\Omega(n/\phi)$ for static graphs, assuming that either d or ϕ is constant.

We now give the intuition behind the proof of Theorem 2 and refer the reader to the full version for the actual proof. The high level approach is as follows. For every step t we define an adaptive adversary that chooses G_{t+1} after observing $V_t^{(0)}$ and $V_t^{(1)}$. The adversary chooses G_{t+1} such that the cut between $V_t^{(0)}$ and $V_t^{(1)}$ is of order of $\Theta(\phi_t \cdot dn)$. We show that such a graph exists when the number of nodes in both $V_t^{(0)}$ and $V_t^{(1)}$ is at least of linear size (in n). By this choice the adversary ensures that the expected potential drop of $\Psi(S_{t+1})$ at most $-c\phi_t d/\Psi(s_t)$ for some constant c. Then we use the expected potential drop, together with the optional stopping theorem, to derive our lower bound.

In the following we observe that if the adversary is allowed to change the degrees, then the expected consensus time is super-exponential. A proof sketch can be found in the full version.

▶ **Observation 1.** *There is a sequence $G_1 = (V, E_1)$, $G_2 = (V, E_2)$, ... be a sequence of graphs with n nodes, where the edges $E_1, E_2, \ldots$ are distributed by an adaptive adversary, such that the expected consensus time is at least $\Omega((n/c)^{n/c})$ for some constant c.*

The bound of Theorem 1 for static regular graphs of $O(n/\phi)$ is tight for regular graphs if either the degree or the conductance is constant. A proof sketch can be found in the full version.

▶ **Observation 2.** *For every n, $d \geq 3$, and constant ϕ, there exists a d-regular graph G with n nodes and a constant conductance such that the expected consensus time on G is $\Omega(n)$. Furthermore, for every even n, $\phi > 1/n$, and constant d, there exists a (static) d-regular graph G with $\Theta(n)$ nodes and a conductance of $\Theta(\phi)$ such that the expected consensus time on G is $\Omega(n/\phi)$.*

3 Analysis of the Biased Voter Model

In this section, we prove Theorem 3. We show that the set S_t of nodes with the preferred opinion grows roughly at a rate of $1 + \Theta(\phi_t)$, as long as S_t has at least logarithmic size. For the analysis we break each round down into several steps, where exactly one node which has at least one neighbour in the opposite set is considered. Instead of analysing the growth of S_t for every round we consider larger time *intervals* consisting of a suitably chosen number of steps. We change the process slightly by assuming that there is always one node with the preferred opinion to allow for an easier analysis. If all other opinions vanish, then node 1 is set to opinion 1. Note that this will only increase the runtime of the process. We also assume that if the preferred opinion vanishes totally, node 1 is set back to the preferred opinion. This alters the process, but as we show later, this event does not happen w.h.p.

The proof unfolds in the following way. First, we define formally the *step sequence* $\mathscr{S}$. Second, we define (Definition 8) a step sequence $\mathscr{S}$ to be *good* if, intuitively speaking, the preferred opinion grows quickly enough in any sufficiently large subsequence of $\mathscr{S}$. Afterward, we show that if $\mathscr{S}$ is a good step sequence, then the preferred opinion prevails in at most

τ''' rounds (Lemma 10). Finally, we show that $\mathscr{S}$ is indeed a good step sequence w.h.p. (Lemma 11).

We now give some definitions. Again, we denote by S_t the random set of nodes that have the preferred opinion right after the first t rounds, and let $S'_t = V \setminus S_t$. For a fixed time step t we write s_t and s'_t. We define the *boundary* ∂s_t as the subset of nodes in s'_t which are adjacent to at least one node from s_t. We use the symmetric definition for $\partial s'_t$. For each $u \in V$, let $\lambda_{u,t}$ be the number of edges incident with u crossing the cut $cut(s_t, s'_t)$, or equivalently, the number of u's neighbours that have a different opinion than u's before round t.

We divide each round t into $|s_t| + |s'_t|$ *steps*, in every step a single node v from either ∂s_t or $\partial s'_t$ randomly chooses a neighbour u and adopts its opinion with the corresponding bias. Note that we assume that v sees u's opinion referring to the *beginning* of the round, even if u was considered before v and changed its opinion in the meantime. It is convenient to label the steps independently of the round in which they take place. Hence, step i denotes the i-th step counted from the *beginning* of the first round. Also u_i refers to the node considered in step i and $\lambda_i = \lambda_{u_i,t}$. We define the indicator variable o_i with $o_i = 1$ if u_i has the preferred opinion and $o_i = 0$ otherwise. Let

$$\Lambda(i) = \sum_{j=1}^{i} (1 - o_i) \cdot \lambda_i \qquad \text{and} \qquad \Lambda'(i) = \sum_{j=1}^{i} o_i \cdot \lambda_i \,.$$

Unfortunately, the order in which the nodes are considered in a round is important for our analysis and cannot be arbitrary. Note that such an ordering does not affect the outcome of the process since the probabilities for a node to switch its opinion only depends on the distribution of opinions at the beginning of the round.

Intuitively, we order the nodes in s_t and s'_t such that the sum of the degrees of nodes which are already considered from s_t and the sum of the degrees of nodes already considered from s'_t differs by at most d, i.e.,

$$|\Lambda_i - \Lambda'_i| \leq d. \tag{6}$$

The following rule determines the node to be considered in step $j+1$: if $\Lambda(j) \leq \Lambda'(j)$, then the (not yet considered) node $v \in \partial s_t$ is with smallest identifier is considered. Otherwise the node $v \in \partial s'_t$ with the smallest identifier is considered. Note that at the first step i of any round we have $\Lambda_i = \Lambda'_i$. This guarantees that (6) holds. The *step sequence* $\mathscr{S}$ is now defined as a sequence of tuples, i.e., $\mathscr{S} = (u_1, Z_1),\ (u_2, Z_2),\ \ldots$, where $Z_j = 1$ if u_j changed its opinion in step j and $Z_j = 0$ otherwise for all $j \geq 1$. Observe that when given the initial assignment and the sequence up to step i, then we know the *configuration* $\mathscr{C}_i$ of the system, i.e., the opinions of all nodes at step i and in which round step i occurred.

In our analysis we consider the increase in the number of nodes with the preferred opinion in time intervals which contain a sufficiently large number of steps, instead of considering one round after the other. The following definitions identify these intervals.

For all $i, k \geq 0$ where $\mathscr{C}_i$ is fixed, we define the random variable $S_{i,k} := \min\{j \colon \Lambda_j - \Lambda_i \geq k\}$, which is the first time step such that nodes with a degree-sum of at least k were considered. Let $I_{i,k} = [i+1, S_{i,k}]$ be the corresponding interval where we note that the length is a random variable. We proceed by showing an easy observation proven in the full version.

▶ **Observation 3.** *The number of steps in the interval $I_{i,k}$ is at most $2k + 2d$, i.e., $|I_{i,k}| \leq 2k + 2d$. Furthermore, $\Lambda'(S_{i,k}) - \Lambda'(i) \leq k + 2d$.*

Fix $\mathscr{C}_i$ and let $X_{i,k}$ be the total number of times during interval $I_{i,k}$ that a switch from a non-preferred opinion to the preferred one occurs; and define $X'_{i,k}$ similarly for the reverse switches. Finally, we define $Y_{i,k} = X_{i,k} - X'_{i,k}$; thus $Y_{i,k}$ is the increase in number of nodes that have the preferred opinion during the time interval $I_{i,k}$.

Define $\ell = \frac{132\beta \log n}{(1-\alpha_1)^2}$ and $\beta' = \frac{600d}{\alpha_1 \cdot (1-\alpha_1)^2}$. In the following we define a *good* sequence.

▶ **Definition 8.** We call the sequence $\mathscr{S}$ of steps *good* if it has all of the following properties for all $i \leq T' = 2\beta' \cdot n$. Consider the first T' steps of $\mathscr{S}$ (fix $\mathscr{C}_{T'}$). Then,

(a) $Y_{0,T'} \geq 2n$ (the preferred opinion prevails in at most T' steps).
(b) $Y_{0,i} + |S_0| > 1$ (the preferred opinion never vanishes).
(c) For any $1 \leq k \leq T'$, $Y_{i,k} \geq -\ell$ (the number of nodes with the preferred opinion never drops by ℓ).
(d) For any $\ell \leq \gamma \leq T'$, $Y_{i,k} > \gamma$, where $k = \gamma \cdot \beta'$ (the nodes with the preferred opinion increase).

This definition allows us to prove in a convenient way that a step sequence $\mathscr{S}$ is w.h.p. good: For each property, we simply consider each (sufficiently large) subsequence S separately and we show that w.h.p. S has the desired property. We achieve this by using a concentration bound on $Y_{i,k}$ which we establish in Lemma 9. Afterward, we take a union bound over all of these subsequences and properties. Using the union bound allows us to show the desired properties in all subsequences in spite of the emerging dependencies. This is done in Lemma 10.

We now show the concentration bounds on $Y_{i,k}$. These bounds rely on the Chernoff-type bound established in the full version. This Chernoff-type bound shows concentration for variables having the property that the sum of the conditional probabilities of the variables, given all previous variables, is always bounded (from above or below) by some b. The bound might be of general interest and its proof can be found in the full version.

▶ **Lemma 9.** *Fix configuration $\mathscr{C}_i$. Then,*

(a) *For $k = \gamma \frac{256d}{\alpha_1 \cdot (1-\alpha_1)^2}$ with $\gamma \geq 1$ it holds that* $\Pr\left(Y_{i,k} < \gamma\right) \leq \exp\left(-\gamma\right)$.
(b) *For $k \geq 0$, any $b' = \alpha_1 \cdot (k + 2d)/d$, and any $\delta > 0$ it holds that*

$$\Pr\left(Y_{i,k} < -(1+\delta)b'\right) \leq \exp\left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{b'}.$$

The following two lemmas are proven in the full version.

▶ **Lemma 10.** *Let $\mathscr{S}$ be a step sequence. Then $\mathscr{S}$ is good with high probability.*

▶ **Lemma 11.** *If $\mathscr{S}$ is a good step sequence, then in at most T' time steps, the preferred opinion prevails and the T' time steps occur before round τ'''.*

Proof of Theorem 3. The claim follows from Lemma 10 together with Lemma 11. ◀

References

1 Mohammed Amin Abdullah and Moez Draief. Global majority consensus by local majority polling on graphs of a given degree sequence. *Discrete Applied Mathematics*, 180:1–10, 2015.

2 D. Aldous and J. Fill. Reversible markov chains and random walks on graphs, 2002. Unpublished. `http://www.stat.berkeley.edu/~aldous/RWG/book.html`.

3 Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 121–132, 2008. doi:10.1007/978-3-540-70575-8_11.

4 C. Cooper, R. Elsässer, H. Ono, and T. Radzik. Coalescing random walks and voting on connected graphs. *SIAM J. Discrete Math.*, 27(4):1748–1758, 2013. doi:10.1137/120900368.

5 C. Cooper, R. Elsässer, and T. Radzik. The power of two choices in distributed voting. In *ICALP*, pages 435–446, 2014.

6 Colin Cooper, Robert Elsässer, Tomasz Radzik, Nicolás Rivera, and Takeharu Shiraga. Fast Consensus for Voting on General Expander Graphs. In *Proc. DISC '15*, pages 248–262, 2015.

7 James Cruise and Ayalvadi Ganesh. Probabilistic consensus via polling and majority rules. *Queueing Systems*, 78(2):99–120, 2014.

8 J. Díaz, L. Goldberg, G. Mertzios, D. Richerby, M. Serna, and P. Spirakis. Approximating fixation probabilities in the generalized moran process. *Algorithmica*, 69(1):78–91, 2014. doi:10.1007/s00453-012-9722-7.

9 P. Donnelly and D. Welsh. Finite particle systems and infection models. *Mathematical Proceedings of the Cambridge Philosophical Society*, 94:167–182, 1983.

10 R. Elsässer, T. Friedetzky, D. Kaaser, F. Mallmann-Trenn, and H. Trinker. Efficient k-Party Voting with Two Choices. *ArXiv e-prints*, February 2016. arXiv:1602.04667.

11 G. Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *STACS*, pages 57–68, 2011.

12 G. Giakkoupis, T. Sauerwald, and A. Stauffer. Randomized rumor spreading in dynamic graphs. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 495–507, 2014. doi:10.1007/978-3-662-43951-7_42.

13 Y. Hassin and D. Peleg. Distributed probabilistic polling and applications to proportionate agreement. *Information and Computation*, 171(2):248–268, 2001. doi:http://dx.doi.org/10.1006/inco.2001.3088.

14 R. Holley and T. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *The Annals of Probability*, 3(4):643–663, 1975. URL: http://www.jstor.org/stable/2959329.

15 B. Houchmandzadeh and M. Vallade. The fixation probability of a beneficial mutation in a geographically structured population. *New Journal of Physics*, 13(7):073020, 2011. URL: http://stacks.iop.org/1367-2630/13/i=7/a=073020.

16 M. Kearns and J. Tan. Biased voting and the democratic primary problem. In *WINE*, pages 639–652, 2008.

17 F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 513–522, 2010. doi:10.1145/1806689.1806760.

18 H. Lam, Z. Liu, M. Mitzenmacher, X. Sun, and Y. Wang. Information dissemination via random walks in d-dimensional space. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'12, pages 1612–1622. SIAM, 2012. URL: http://dl.acm.org/citation.cfm?id=2095116.2095244.

19 N. Lanchier and C. Neuhauser. Voter model and biased voter model in heterogeneous environments. *Journal of Applied Probability*, 44(3):770–787, 2007.

20 T. Liggett. *Interacting Particle Systems*. Springer Berlin Heidelberg, 1985. doi:10.1007/b138374.

21 G. Mertzios and P. Spirakis. Strong bounds for evolution in networks. In *ICALP*, pages 669–680, 2013.

22 T. Nakata, H. Imahayashi, and M. Yamashita. A probabilistic local majority polling game on weighted directed graphs with an application to the distributed agreement problem. *Networks*, 35(4):266–273, 2000. doi:10.1002/1097-0037(200007)35:4<266::AID-NET5>3.0.CO;2-4.

23 R. Oliveira. On the coalescence time of reversible random walks. *Trans. Am. Math. Soc.*, 364(4):2109–2128, 2012.

24 Y Peres, A. Sinclair, P. Sousi, and A. Stauffer. Mobile geometric graphs: detection, coverage and percolation. *Probability Theory and Related Fields*, 156(1-2):273–305, 2013. doi:10.1007/s00440-012-0428-1.

Bootstrap Percolation on Geometric Inhomogeneous Random Graphs*

Christoph Koch[†1] **and Johannes Lengler**[2]

1 **Institute of Discrete Mathematics, Graz University of Technology, Graz, Austria**
ckoch@math.tugraz.at

2 **Department of Computer Science, ETH Zürich, Zürich, Switzerland**
johannes.lengler@inf.ethz.ch

Abstract

Geometric inhomogeneous random graphs (GIRGs) are a model for scale-free networks with underlying geometry. We study bootstrap percolation on these graphs, which is a process modelling the spread of an infection of vertices starting within a (small) local region. We show that the process exhibits a phase transition in terms of the initial infection rate in this region. We determine the speed of the process in the supercritical case, up to lower order terms, and show that its evolution is fundamentally influenced by the underlying geometry. For vertices with given position and expected degree, we determine the infection time up to lower order terms. Finally, we show how this knowledge can be used to contain the infection locally by removing relatively few edges from the graph. This is the first time that the role of geometry on bootstrap percolation is analysed mathematically for geometric scale-free networks.

1998 ACM Subject Classification C.2.1. Network Architecture and Design

Keywords and phrases Geometric inhomogeneous random graphs, scale-free network, bootstrap percolation, localised infection process, metastability threshold

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.147

1 Introduction

One of the most challenging and intriguing questions about large real-world networks is how activity spreads through the network. "Activity" in this context can mean many things, including infections in a population network, opinions and rumours in social networks, viruses in computer networks, action potentials in neural networks, and many more. While all these networks seem very different, in the last two decades there was growing evidence that most of them share fundamental properties [4, 24]. The most famous property is that the networks are *scale-free*, i.e. the degrees follow a power-law distribution $\Pr[\deg(v) \geq d] \approx d^{1-\beta}$, typically for some $2 < \beta < 3$. Other properties include a large connected component which is a small world (poly-logarithmic diameter) and an ultra-small world (constant or poly-loglog average distance), that the networks have small separators and a large clustering coefficient. We refer the reader to [15] for more detailed discussions.

Classical models for random graphs fail to have these common properties. For example, Erdős-Rényi graphs or Watts-Strogatz graphs do not have power-law degrees, while Chung-Lu

* A preprint of the full version of this paper is available at http://arxiv.org/abs/1603.02057.

† Most of the research was conducted during the first author's stay at ETH Zürich. The first author is supported by Austrian Science Fund (FWF): P26826 and W1230.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 147; pp. 147:1–147:15

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

graphs and preferential attachment (PA) graphs fail to have large clustering coefficients or small separators. The latter properties typically arise in real-world networks from an underlying geometry, either spatial or more abstract, e.g., two nodes in a social networks might be considered "close" if they share similar professions or hobbies. It is well-known that in real-world networks the spread of activity (of the flu, of viral marketing, ...) is crucially governed by the underlying spatial or abstract geometry [41]. For this reason, the explanatory power of classical models is limited in this context.

In recent years models have been developed which overcome the previously mentioned limitations, most notably *hyperbolic random graphs* (HypRGs) [13, 12, 10, 44] and their generalisation[1] *geometric inhomogeneous random graphs* (GIRGs) [15][2], and *spatial preferential attachment* (SPA) models [2, 22, 35]. Apart from the power-law exponent β, these models come with a second parameter $\alpha > 1$, which models how strongly the edges are predicted by their distance. Due to their novelty, there are only very few theoretical results on how the geometry impacts the spreading of activity through these networks.

In this paper we make a first step by analysing a specific process, *bootstrap percolation* [20], on the recent and very general GIRG model. In this process, an initial set of *infected* (or *active*) vertices iteratively infects all vertices which have at least k infected neighbours, where $k \geq 2$ is a parameter. It was originally developed to model various physical phenomena (see [1] for a short review), but has by now also become an established model for the spreading of activity in networks, for example for the spreading of beliefs [32, 25, 48, 45], behaviour [30, 31], or viral marketing [38] in social networks (see also [19]), of contagion in economic networks [6], of failures in physical networks of infrastructure [52] or compute architecture [39, 28], of action potentials in neuronal networks (e.g, [47, 49, 5, 21, 50, 43, 26, 27], see also [40] for a review), and of infections in life networks [25].

1.1 Our contribution

We investigate bootstrap percolation on GIRGs with an expected number of n vertices. We fix a ball B in the underlying geometric space, and we initially infect each vertex in B independently with probability ρ. In this way, we model that an infection (a rumour, an opinion, ...) often starts in some local region, and from there spreads to larger parts of the network. In Theorem 1 we determine a threshold ρ_c such that in the *supercritical case* $\rho \gg \rho_c$ whp[3] a linear fraction of the graph is infected eventually, and in the *subcritical case* $\rho \ll \rho_c$ infection ceases immediately. In the *critical case* $\rho = \Theta(\rho_c)$ both options occur with non-vanishing probability. If there are enough (at least k) "local hubs" in the starting region, i.e. vertices of relatively large expected degree, then they become infected and facilitate the process. On the other hand, without such local hubs the initial infection is not dense enough, and comes to a halt.

For the supercritical case, we show that it only takes $O(\log\log n)$ rounds until a constant fraction of all vertices is infected, and we determine the number of rounds until this happens up to a factor $1 \pm o(1)$ in Theorem 2. For the matching lower bound in this result, we need the technical condition $\alpha > \beta - 1$, i.e. edge-formation may not depend too weakly on the

[1] It is non-obvious that GIRGs are a generalization of HypRGs, see [15, Theorem 6.3].

[2] Other than in [15] we do not condition on the number of vertices to be exactly n, which leads to slightly less technical proofs.

[3] with high probability, i.e. with probability tending to 1 as $n \to \infty$. All unspecified limits and asymptotics will be with respect to $n \to \infty$. For example, for a function $f = f(n)$ the notation $f = O(1)$ means that there is $n_0 > 0$ and an absolute constant $C > 0$ that depends only the constant parameters $\alpha, \beta, d, w_{\min}, k$ of the model, such that $f(n) \leq C$ for all $n \geq n_0$. Similarly, $f = \omega(1)$ means $\lim_{n\to\infty} f(n) = \infty$ etc.

geometry. Notably, if the starting region B is sufficiently small then the number of rounds agrees (up to minor terms) with the average distance in the network. In particular, it does not depend on the infection rate ρ, as long as ρ is supercritical.

Finally we demonstrate that the way the infection spreads is strongly governed by the geometry of the process, again under the assumption $\alpha > \beta - 1$. Starting from B, the infection is carried most quickly by local hubs. Once the local hubs in a region are infected, they pass on their infection **a)** to other hubs that are even further away, and **b)** locally to nodes of increasingly lower degree, until a constant fraction of all vertices the region is infected. Indeed, given a vertex v (i.e. given its expected degree and its distance from B), and assuming that v is not too close to B, we can predict whp (Theorem 4) in which round it will become infected, again up to a factor $1 \pm o(1)$. In real applications such knowledge is invaluable: for example, assume that a policy-maker only knows initial time and place of the infection, i.e. she knows the region B and the current round i. In particular, she does not know ρ, she does not know the graph, and she has no detailed knowledge about who is infected. Then we show that she is able to identify a region B' in which the infection can be quarantined. In other words, by removing (from round i onwards) all edges crossing the boundary of B' whp the infection remains contained in B'. The number of edges to be deleted is relatively small: it can be much smaller than n (in fact, any function $f = f(n)$ satisfying $f = \omega(1)$ can be an upper bound, if i and $\mathrm{Vol}(B)$ are sufficiently small), and it is even much smaller than the number of edges *inside* of B', as was already noted in [15].

1.2 Related work

The GIRG model was introduced in [15], and we rely on many results from this paper. The average distance of a GIRG (which, as we show, agrees with the time until the bootstrap percolation process has infected a constant portion of all vertices) was determined in [16] in a much more general setup.

Bootstrap percolation has been intensively studied theoretically and experimentally on a multitude of networks, including trees [20, 9], lattices [3, 8], Erdős-Rényi graphs [36], various geometric graphs [49, 42, 14, 29], and scale-free networks [23, 11, 7, 38]. On geometric scale-free networks there are some experimental results [18], but little is known theoretically. Recently, Candellero and Fountoulakis [17] determined the threshold for bootstrap percolation on HypRGs (in the threshold case $\alpha = \infty$, cf. below), but they assumed that the initial infection takes place *globally*, i.e. whether any vertex is infected initially is independent of its position, and not *locally* as in our paper, where no vertex outside of a certain geometric region is infected initially. This has two major consequences. Firstly, in the global setting, the (expected) number of initially infected vertices needs to be polynomial in n in order for the infection to start spreading significantly; while in our setting every ball containing an expected number of $\omega(1)$ vertices can initiate a large infection whp. Secondly, using our knowledge about how the process evolves in time with respect to the geometry, we show that the infection time of any vertex is mainly governed by its geometric position and its weight. On the other hand, with a global initial infection the infection times only depend on the expected degrees. Note that we do not encode these expected degrees as geometric information (in contrast to [17]), but rather in the weights. Similarly, the questions studied in this paper do not apply for non-geometric random graph models.

While there is plenty of experimental literature and also some mean-field heuristics on other activity spreading processes on geometric scale-free networks (e.g., [51, 53, 34, 54, 33, 46]), rigorous mathematical treatments are non-existent with the notable exception of [37], where rumour spreading is analysed in an SPA model with a push and a push&pull protocol.

2 Model and notation

Graph model

A GIRG is a graph $G = (V, E)$ where both the vertex set V and the edge set E are random. Each vertex v is represented by a pair (x_v, w_v) consisting of a *position* x_v (in some *ground space*) and a *weight* $w_v \in \mathbb{R}_{>0}$.

Ground space and positions. We fix a (constant) dimension $d \geq 1$ and consider the d-dimensional torus $\mathbb{T}^d = \mathbb{R}^d/\mathbb{Z}^d$ as the ground space. We usually think of it as the d-dimensional cube $[0,1]^d$ where opposite boundaries are identified and measure distances by the ∞-norm on $\mathbb{T}^d$, i.e. for $x, y \in [0,1]^d$ set $\|x - y\| := \max_{1 \leq i \leq d} \min\{|x_i - y_i|, 1 - |x_i - y_i|\}$.

The set of vertices and their positions are given by a homogeneous Poisson point process on $\mathbb{T}^d$ with intensity $n \in \mathbb{N}$. More formally, for any (Lebesgue-)measurable set $B \subseteq \mathbb{T}^d$, let $V \cap B$ denote (with slight abuse of notation) the set of vertices with positions in B. Then $|V \cap B|$ is Poisson distributed with mean $n\mathrm{Vol}(B)$, i.e. for any integer $m \geq 0$ we have

$$\Pr\left[|V \cap B| = m\right] = \Pr[\mathrm{Po}\left(n\mathrm{Vol}(B)\right) = m] = \frac{(n\mathrm{Vol}(B))^m \exp(-n\mathrm{Vol}(B))}{m!},$$

and if B and B' are disjoint measurable subsets of $\mathbb{T}^d$ then $|V \cap B|$ and $|V \cap B'|$ are independent. Note in particular that the total number of vertices $|V|$ is Poisson distributed with mean n, i.e. it is also random. An important property of this process is the following: Given a random vertex[4] $v = (x_v, w_v)$, if we condition on $x_v \in B$, where B is some measurable subset of $[0,1]^d$, then the position x_v is uniformly distributed in B.

Weights. For each vertex, we draw independently a weight from some distribution $\mathcal{D}$ on $\mathbb{R}_{>0}$. We say that the weights follow a *weak power-law* for some exponent $\beta \in (2,3)$ if a $\mathcal{D}$-distributed random variable D satisfies the following two conditions: There is a constant $w_{\min} \in \mathbb{R}_{>0}$ such that $\Pr\left[D \geq w_{\min}\right] = 1$, and for every constant $\gamma > 0$ there are constants $0 < c_1 \leq c_2$ such that

$$c_1 w^{1-\beta-\gamma} \leq \Pr\left[D \geq w\right] \leq c_2 w^{1-\beta+\gamma} \tag{1}$$

for all $w \geq w_{\min}$. If this condition is also satisfied for $\gamma = 0$, then we say that the weights follow a *strong power-law*.

Edges. Next we fix a constant $\alpha \in \mathbb{R}_{>1} \cup \{\infty\}$. Then (conditional on the Poisson point process) two distinct vertices $u = (x_u, w_u)$ and $v = (x_v, w_v)$ form an edge independently of all other pairs with probability $p(x_u, x_v, w_u, w_v)$, where the function p satisfies

$$p(x_u, x_v, w_u, w_v) = \Theta(1) \min\left\{\left(\frac{w_u w_v}{\|x_u - x_v\|^d n}\right)^\alpha, 1\right\},$$

if $\alpha < \infty$. In the *threshold model* $\alpha = \infty$ we instead require that p satisfies

$$p(x_u, x_v, w_u, w_v) = \begin{cases} \Omega(1) & \text{if } \|x_u - x_v\| \leq C_1 \left(\frac{w_u w_v}{n}\right)^{1/d} \\ 0 & \text{if } \|x_u - x_v\| > C_2 \left(\frac{w_u w_v}{n}\right)^{1/d} \end{cases}$$

for some constants $0 < C_1 \leq C_2$. Note that for $C_1 \neq C_2$ the edge probability may be arbitrary in the interval $\left(C_1 \left(\frac{w_u w_v}{n}\right)^{1/d}, C_2 \left(\frac{w_u w_v}{n}\right)^{1/d}\right)$.

[4] By abuse of notation, x_v and w_v may either denote random variables or values.

Bootstrap percolation

Let $k \geq 2$ be a constant, let $B_0 \subseteq \mathbb{T}^d$ be measurable, and let $0 \leq \rho \leq 1$. Then the bootstrap percolation process with *threshold* k, *starting region* B_0, and *initial infection rate* ρ is the following process. For every integer $i \geq 0$ there is a set $V^i \subseteq V$ of vertices which are *infected* (or *active*) at time i. The process starts with a random set $V^0 \subseteq V$ which contains each vertex in $V \cap B_0$ independently with probability ρ, and which contains no other vertices. For all integers $i \geq 0$ we then define the set V^{i+1} iteratively by

$$V^{i+1} := V^i \cup \left\{ v \in V \;\middle|\; v \text{ has at least } k \text{ neighbours in } V^i \right\}.$$

Moreover, we set $V^\infty := \bigcup_{i \in \mathbb{N}} V^i$, and for convenience of notation we extend this definition to real parameters $i \in \mathbb{R}_{>0}$ by setting $V^i := V^{\lceil i \rceil}$. For a vertex $v \in V$, we define its *infection time* as $L_v := \inf \left\{ i \geq 0 \;\middle|\; v \in V^i \right\}$ and $L_v := \infty$ if the infimum does not exist.

We denote by $\nu = \nu(n) := n\mathrm{Vol}(B_0)$ the expected number of vertices in B_0. Throughout the paper we will assume that B_0 is a ball, which is – without loss of generality due to symmetry of $\mathbb{T}^d$ – centred at 0. Moreover, we will assume that $\nu = \omega(1)$.

Further notation

We denote the neighbourhood of a vertex $v \in V$ by $N(v) := \{u \in V \mid \{u, v\} \in E\}$. Furthermore, for any two sets of vertices U_1 and U_2, we denote the set of edges between them by $E(U_1, U_2) := \{e = \{u_1, u_2\} \mid u_1 \in U_1, u_2 \in U_2\}$. For any $\lambda \geq 0$ and any closed ball $B \subseteq \mathbb{T}^d$ of radius $r \geq 0$ centred at 0 we denote by λB the closed ball of radius λr around 0. By abuse of notation, if $S \subset V$ and $B \subseteq \mathbb{T}^d$ then $S \cap B := \{v \in S \mid x_v \in B\}$.

3 Main results

First of all we show that bootstrap percolation on a *GIRG* has a threshold with respect to the initial infection rate ρ. Since HypRGs are a special instance of GIRGs, this contains in particular the result of [17] on (threshold) HypRGs, where the case $\nu = n$ was studied.

▶ **Theorem 1.** *Consider a bootstrap percolation process on a GIRG $G = (V, E)$ with constant parameters $\alpha, \beta, d, w_{min}, k$, initial infection rate $\rho = \rho(n) \in [0, 1]$, and initial infection region B_0 with volume ν/n, where $\nu = \nu(n) = \omega(1)$. We set*

$$\rho_c = \rho_c(n) := \nu^{-\frac{1}{\beta - 1}}.$$

If the weights follow a strong power-law, then as $n \to \infty$ we have:

- **(i)** *If $\rho = \omega(\rho_c)$, then $|V^\infty| = \Theta(n)$ whp.*
- **(ii)** *If $\rho = \Theta(\rho_c)$, then $|V^\infty| = \Theta(n)$ with probability $\Omega(1)$, but also $V^\infty = V^0$ with probability $\Omega(1)$.*
- **(iii)** *If $\rho = o(\rho_c)$, then $V^\infty = V^0$ whp.*

If the weights follow a weak power-law, then as $n \to \infty$ we have:

- **(iv)** *If there is a constant $\delta > 0$ such that $\rho \geq \rho_c^{1+\delta}$, then $|V^\infty| = \Theta(n)$ whp.*
- **(v)** *If there is a constant $\delta > 0$ such that $\rho \leq \rho_c^{1-\delta}$, then $V^\infty = V^0$ whp.*

Whenever we refer to the *supercritical* regime we mean case (i) and (iv). Similarly, (iii) and (v) form the *subcritical* regime and (ii) is the *critical* regime. Note in particular that there is a supercritical regime regardless of how small the expected number ν of vertices in the starting region is, provided that $\nu = \omega(1)$. This is in sharp contrast to non-geometric

graphs like Chung-Lu graphs, where the expected number of initially infected vertices must be polynomial in n (if the set of initially infected vertices is chosen at random).

Indeed the proof of Theorem 1 will grant a deeper insight into the evolution of the process. Since the process whp stops immediately in the subcritical regime, we may restrict ourselves to the other cases. We show a doubly logarithmic upper bound on the number of rounds until a constant fraction of all vertices are infected. Furthermore, we prove that this bound is tight up to minor order terms if the influence of the underlying geometry on the random graphs is sufficiently strong ($\alpha > \beta - 1$). Remarkably, the bounds do not depend on the initial infection rate ρ, as long as ρ is supercritical. Moreover, if the expected number ν of vertices in the starting region is sufficiently small (if $\log\log\nu = o(\log\log n)$), then the bound coincides with the average distance in the graph, again up to minor order terms.

▶ **Theorem 2.** *In the situation of Theorem 1, let $\varepsilon > 0$ be constant and set*

$$i_\infty := \frac{\log\log_\nu n + \log\log n}{|\log(\beta-2)|}.$$

Then in the supercritical regime whp, and in the critical regime with probability $\Omega(1)$ we have $|V^{(1+\varepsilon)i_\infty}| = \Theta(n)$, as $n \to \infty$.
If furthermore $\alpha > \beta - 1$ and $\nu = n^{o(1)}$ then in all regimes we have whp $|V^{(1-\varepsilon)i_\infty}| = o(n)$, as $n \to \infty$.

In fact, we can still refine the statement of Theorem 2 tremendously, at least in the case $\alpha > \beta - 1$. In the following, we determine for every fixed vertex v its infection time L_v, up to minor order terms (with the restriction that v may not be too close to the starting region). We will show that it is given by the following expression (see also Remark 6 below).

▶ **Definition 3.** For any $x \in \mathbb{T}^d \setminus B_0$ and $w \in \mathbb{R}_{>0}$ we define

$$\Lambda(x,w) := \begin{cases} \max\left\{0,\ \frac{\log\log_\nu\left(\|x\|^d n/w\right)}{|\log(\beta-2)|}\right\}, & \text{if } w > (\|x\|^d n)^{1/(\beta-1)}, \\ \frac{2\log\log_\nu(\|x\|^d n) - \log\log_\nu w}{|\log(\beta-2)|}, & \text{if } w \le (\|x\|^d n)^{1/(\beta-1)}. \end{cases} \tag{2}$$

In the first case we use the convention that the second term is $-\infty$ if $\|x\|^d n/w < 1$, and thus does not contribute to the maximum.

Note that in the second case, the sign of $\log\log_\nu w$ may be either positive or negative. However, then we have the lower bound $\Lambda(x,w) \ge \log\log_\nu(\|x\|^d n)/|\log(\beta-2)| + O(1)$ due to the upper bound of w and thus, in particular $\Lambda(x,w) \ge 0$, since $x \in \mathbb{T}^d \setminus B_0$.

▶ **Theorem 4.** *Assume we are in the situation of Theorem 1 in the supercritical regime. Let $v = (x_v, w_v)$ be any fixed vertex such that $x_v \in \mathbb{T}^d \setminus B_0$, $w_v = \omega(1)$ and $\Lambda(x_v, w_v) \le \log_2(\|x_v\|^d n/\nu^{2/(\beta-2)})$. Then, as $n \to \infty$, the infection time L_v satisfies whp*

$$L_v \le (1 + o(1))\Lambda(x_v, w_v) + O(1).$$

If additionally $\alpha > \beta - 1$ then, as $n \to \infty$ we also have whp

$$L_v \ge (1 - o(1))\Lambda(x_v, w_v) - O(1).$$

As in Theorem 2, the bounds do not depend on the initial infection rate ρ, as long as it is supercritical.

▶ Remark 5. The technical restrictions in Theorem 4 are necessary: if a vertex v has weight $w_v = O(1)$ then the number of neighbours is Poisson distributed with mean $\Theta(w_v)$ (see Lemma 8), so v is even isolated with probability $\Omega(1)$. In particular, we cannot expected that whp v is ever infected.

The restriction $\Lambda(x_v, w_v) \leq \log_2(\|x_v\|^d n/\nu^{2/(\beta-2)})$ ensures that v is not too close to the starting region. If v is too close, then it may have neighbours inside of B_0, and in this case it does depend on ρ when they are infected. (And of course, this process iterates.) The term $\log_2(\|x_v\|^d n/\nu^{2/(\beta-2)})$ is not tight and could be improved at the cost of more technical proofs. However, there are already rather few vertices that violate the condition $\Lambda(x_v, w_v) \leq \log_2(\|x_v\|^d n/\nu^{2/(\beta-2)})$. For example, recall that it only takes $O(\log\log n)$ steps until a constant fraction of all vertices are infected. At this time, we only exclude vertices which satisfy $\|x_v\|^d n \leq \nu^{2/(\beta-2)} \cdot (\log n)^{O(1)}$, so the expected number of affected vertices is also at most $\nu^{2/(\beta-2)} \cdot (\log n)^{O(1)}$. Even this is a gross overestimate, since the vertices close to the origin have much smaller infection times L_v, and thus only very few of them are affected by the condition.

▶ Remark 6. The first case in Definition 3 is not needed if we restrict ourselves to vertices as they typically appear in GIRGs. More precisely, as we will see in Lemma 10, whp all vertices $v = (x_v, w_v) \in V \cap (\mathbb{T}^d \setminus B_0)$ satisfy $w_v \leq (\|x_v\|^d n)^{1/(\beta-1-\eta)}$ where $\eta > 0$ is an arbitrary constant. In the border case $(\|x_v\|^d n)^{1/(\beta-1)} \leq w_v \leq (\|x_v\|^d n)^{1/(\beta-1-\eta)}$ both expressions in (2) agree up to additive constants, i.e.

$$\Lambda(x_v, w_v) = \frac{2\log\log_\nu(\|x_v\|^d n) - \log\log_\nu w_v}{|\log(\beta-2)|} \pm O(1). \tag{3}$$

Therefore, we could also use (3) as definition for Λ if we would exclude vertices which are unlikely to exist in Theorem 4 .

Finally, we give a strategy how to contain the infection within a certain region when only the starting set and the current round are known, but not the set of infected vertices. Note that the number of edges that need to be removed is substantially smaller than the expected number $\widetilde{\nu}_i$ of vertices in a containment area $\widetilde{B}_i$, see Definition 11.

▶ **Theorem 7.** *Assume that we are in the situation of Theorem 1, and that $\alpha > \beta - 1$. If the starting region B_0 is known, then by removing all edges crossing the boundary of $\widetilde{B}_i$ before round $i+1$, whp (as $n \to \infty$) the infection is contained in $\widetilde{B}_i$. The expected number of edges crossing the boundary of $\widetilde{B}_i$ is $\widetilde{\nu}_i^{\max\{3-\beta, 1-1/d\}+o(1)}$.*

4 Basic properties of GIRGs

In this section we list briefly some basic properties of GIRGs (without proofs). The first lemma, based on [16, Lemma 4.4 and Theorem 7.3], tells us that the expected degree of a vertex equals its weight, up to constant factors. Moreover, it gives the marginal probability that two vertices u, v of fixed weights but random positions in $\mathbb{T}^d$ are adjacent. This probability remains the same if the position of one (but not both) of the vertices is fixed.

▶ **Lemma 8.** *Let $v = (x_v, w_v)$ be a vertex with fixed weight and position. Then $\deg(v)$ is Poisson distributed with mean $\Theta(w_v)$. Moreover, if $u = (x_u, w_u)$ is a vertex with fixed weight, but with random position $x_u \in \mathbb{T}^d$, then*

$$\Pr[\{u,v\} \in E \mid w_u, w_v, x_v] = \Theta\left(\min\left\{\frac{w_u w_v}{n}, 1\right\}\right). \tag{4}$$

Note in particular that the right hand side of (4) *is independent of* x_v*, so the same formula still applies if also the position* x_v *of* v *is randomised.*

We often need to bound the expected number of neighbours of a given vertex in some geometric region, which we may do by the following lemma.

▶ **Lemma 9.** *Let* $\eta > 0$ *and* $C > 1$ *be constants, define* $m := \min\{\alpha, \beta - 1 - \eta\}$ *and consider a closed ball* $B \subseteq \mathbb{T}^d$ *of radius* $r > 0$ *centred at* 0*. Let* $v = (x_v, w_v)$ *be a vertex with fixed weight and position. Then*

$$\mathbb{E}\left[|N(v) \cap B|\right] = O(n\mathrm{Vol}(B)) \cdot \begin{cases} \min\left\{\frac{w_v}{n\mathrm{Vol}(B)}, 1\right\}, & \text{if } \|x_v\| \le Cr, \\ \min\left\{\left(\frac{w_v}{\|x_v\|^d n}\right)^m, 1\right\} & \text{if } \|x_v\| \ge Cr. \end{cases}$$

The last lemma states that whp there are no vertices whose weight is much larger than their distance from the origin.

▶ **Lemma 10.** *Let* $\eta > 0$ *be a constant and consider a closed ball* $B \subseteq \mathbb{T}^d$ *of radius* $r > 0$ *centred at* 0*, satisfying* $n\mathrm{Vol}(B) = \Theta(r^d n) = \omega(1)$*. Then with probability* $1 - (r^d n)^{-\Omega(\eta)}$ *there is no vertex* $v = (x_v, w_v)$ *with* $x_v \in \mathbb{T}^d \setminus B$ *and* $w_v \ge (\|x_v\|^d n)^{1/(\beta-1-\eta)}$*.*

5 Proof outline

5.1 Intuition

Due to space limitations we can only give a very rough sketch of the main ideas. We warn the reader that the statements as they are formulated in this section are not literally true, but they are only true if appropriate error margins (slightly smaller/larger weights or regions) are taken into account. The same holds for definitions within this section. The rigorous definitions and statements with full technical details can be found in Section 5.2 and 5.3.

For the subcritical regime, we distinguish between high-weight vertices ($w_v = \omega(w_0)$, where $w_0 := \nu^{1/(\beta-1)}$) and low-weight vertices ($w_v = O(w_0)$). By an easy computation, the expected number of low-weight vertices in B_0 that are infected in round 1 is $o(1)$, so by Markov's inequality no low-weight vertex becomes infected whp. On the other hand, whp no high-weight vertex exists in B_0, and the expected number of infected vertices outside of B_0 is also $o(1)$ because they are too far away from infected vertices. In order words, whp no vertex is infected in round 1.

In the critical regime, the calculation is similar, but if there exist vertices of weight $\Theta(w_0)$ then these vertices are infected with probability $\Omega(1)$. The number of vertices of weight $\Theta(w_0)$ is Poisson distributed with mean $\Theta(1)$, so it may happen (both with probability $\Omega(1)$) that either no such vertex exists (so percolation stops) or that there are at least k such vertices, and all of them are infected. In the supercritical regime, whp k vertices of weight (slightly less than) w_0 are infected. Whp, these k vertices infect all other vertices of similar weight in two more rounds. This is sufficient to start an avalanche of infection, and for the rest of this section we will restrict ourselves to this case.

If the infection gets started, then it evolves as follows. Let $\zeta := 1/(\beta - 2) > 1$, and consider the sequence B_i of nested balls of volume ν_i/n centred at 0, where $\nu_i := \nu^{\zeta^i}$. Then in the i-th round, all vertices of weight roughly $w_i := \nu_i^{1/(\beta-1)}$ in B_i are infected. In the next round, whp the vertices of weight w_i in B_i infect all vertices of weight w_{i+1} in B_{i+1}, thus spreading the infection to new regions. Note that this statement is easy to prove inductively since we assumed that *all* vertices of weight w_i in B_i are infected, so for the vertices in B_{i+1}

it suffices to count the number of neighbours of a certain weight in B_i, which is a Poisson distributed random variable. This gives a lower bound on how fast the infection spreads geometrically. It can not spread faster since whp there are no edges from B_i to $\mathbb{T}^d \setminus B_{i+1}$. This latter fact already allows us to execute a containment strategy.

On the other hand, if in round j every vertex of weight w in some region has a large probability to be infected, then in round $j+1$ every vertex of weight at least $w^{1/\zeta}$ in this region has a large (though slightly smaller) probability to be infected. To prove this formally, we consider a vertex of weight $w^{1/\zeta}$. Such a vertex (but not vertices of smaller weight) has at least w^δ neighbours of weight w, with probability at least $1 - \exp[-w^\delta]$. So we pick k such neighbours, and bound the probability that at least one of them is *not* infected by a union bound. In this way, we lose a factor of k in each round, but by going through the proof details it turns out that this factor is still negligible compared to the error term $\exp[-w^{\delta/\zeta}]$.

Complementing this infection pathway by a matching upper bound is the most challenging and technical part of the proof. In round $i-1$ there is no infected vertex in B_i, so it is not hard to argue that in round i only vertices of large weight in $\mathbb{T}^d \setminus B_{i-1}$ are infected. However, in subsequent rounds it does happen that vertices of very small weight in $\mathbb{T}^d \setminus B_{i-1}$ become infected. Fortunately, this only happens with rather small probability, which we can explicitly bound (Theorem 13 (f)) as a function of the weight. Once we have such a bound in some round, we use that whp no vertex in $\mathbb{T}^d \setminus B_{i-1}$ (not too close to the boundary) has strictly more than one neighbour in B_{i-1}. Therefore, in order to be infected, at least one of its neighbours in $\mathbb{T}^d \setminus B_{i-1}$ must have been infected in the previous round, and we can bound the probability of this event by the expected number of previously infected neighbours in $\mathbb{T}^d \setminus B_{i-1}$. It turns out that this simple bound is sufficient to provide the desired matching upper bound, safe quite some technical details which we omit.

We remark that it is in this last step where we use the assumption $\alpha > \beta - 1$ since otherwise there do exist vertices in $\mathbb{T}^d \setminus B_{i-1}$ that have several neighbours in B_{i-1}, and these vertices exist in a substantial part of B_i. Even worse, in some (large) subregion of B_i, the number of infections in round $i+1$ that come from neighbours in B_{i-1} dominates the number of infections that come from neighbours in B_i. For investigating the case $\alpha \le \beta - 1$ (which we don't in this paper), it will no longer be possible to use a bound on the infection probability that is uniform within $\mathbb{T}^d \setminus B_{i-1}$, or within $B_i \setminus B_{i-1}$.

Once the claims outlined above are proven (Theorem 13 and 14) we have almost complete control over the process. In particular, for a each vertex v with fixed weight and position (outside of the starting region B_0), and for each round j we have lower and upper bounds for the probability that v has already become infected by round j. We can thus compute rounds j_1, j_2 for which the probability is at most $o(1)$ and at least $1 - o(1)$, respectively, and we find that these rounds coincide up to lower order terms. It is still rather complicated to actually perform the calculations of j_1 and j_2 due to the many technical details which we omitted in this outline, but no further knowledge about the infection process is required.

5.2 Formal statements and sketch of proofs

In this section we will give two theorems which describe the geometrical evolution of the process in detail, and which make the intuitions from Section 5.1 precise. Theorem 13 states that **a)** certain regions cannot be reached too early by the infection, and **b)** within an infected region, vertices of too low weight have a small probability to be infected early. Hence, the theorem gives an *upper bound on the speed* of the infection process. Note that this already gives the quarantine statement (Theorem 7), see Section 5.3 for details. Afterwards, Theorem 14 gives lower bounds on the probability that a vertex in a given region is infected

at a given time, in the supercritical case. In particular, a vertex of weight $w_v = \omega(1)$ will eventually be infected whp. Thus the theorem provides a *lower bound on the speed* of the process. This lower bound also applies in the critical regime if in the first step sufficiently many heavy vertices became infected, an event which holds with at least constant probability.

In Section 5.1 we introduced balls B_i which essentially correspond to the region of infected vertices in round i. For the formal statements we need slightly smaller and larger balls, which we now define formally. In general, $\tilde{\nu}_i, \tilde{B}_i$ etc. will denote the upper bound variants.

▶ **Definition 11.** For all $0 < \varepsilon < \zeta = 1/(\beta-2)$ and all $i \geq 0$, we set

$$\nu_0 := \nu \qquad \text{and} \qquad \nu_i = \nu_i(\varepsilon) := \nu_0^{(\zeta-\varepsilon)^i},$$
$$\widetilde{\nu}_0 = \widetilde{\nu}_0(\varepsilon) := \nu^{(\beta-1)/(\beta-2)+\varepsilon} \qquad \text{and} \qquad \widetilde{\nu}_i = \widetilde{\nu}_i(\varepsilon) := \widetilde{\nu}_0^{(\zeta+\varepsilon)^i}$$

We define $B_i := B_i(\varepsilon)$ and $\widetilde{B}_i := \widetilde{B}_i(\varepsilon)$ to be the closed ball centred around 0 of volume $\min\{\nu_i(\varepsilon)/n, 1\}$ and $\min\{\widetilde{\nu}_i(\varepsilon)/n, 1\}$, respectively. Note that $B_i(\varepsilon) \subseteq \widetilde{B}_i(\varepsilon')$ for all $i \geq 0$ and all $0 < \varepsilon, \varepsilon' < \zeta$.

First we give an upper bound on the speed of the process. For a formal statement, we define the following families of "good" events.

▶ **Definition 12.** Let $\varepsilon > 0$ be a constant and let $\eta = \eta(\varepsilon) > 0$ be a constant which is sufficiently small compared to ε. Moreover, let $h = h(n)$ be a function satisfying $h(n) = \omega(1)$, $h(n) = o(\log n)$, and $h(n) = \nu^{o(1)}$. Then for all $i, \ell, j \geq 0$ we define the following families of events:

- $\mathcal{E}(i) := \{V^i \cap (\mathbb{T}^d \setminus \widetilde{B}_i) = \emptyset\}$;
- For all $w \geq w_{\min}$ let $S(w,\ell) := \{v \in V^{\leq \ell} \mid w_v \geq w\}$. We set

$$\mathcal{F}(\ell, w) = \mathcal{F}_{\varepsilon,\eta,h}(\ell, w) := \left\{ |S(w,\ell)| \leq h^\ell w^{2-\beta+\eta} \widetilde{\nu}_0^{1-(\zeta+\varepsilon)^{-\ell}(\beta-1)^{-1}} \right\},$$

  and $\mathcal{F}(\ell) = \mathcal{F}_{\varepsilon,\eta,h}(\ell) := \bigcap_{w' \geq w_{\min}} \mathcal{F}(\ell, w')$;
- $\mathcal{G}(j) = \mathcal{G}_{\varepsilon,\eta,h}(j) := \bigcap_{j'=0}^{j} (\mathcal{E}(j') \cap \mathcal{F}(j'))$.

In other words, $\mathcal{E}(i)$ means that no vertex outside of $\widetilde{B}_i$ is infected at time i, and $\mathcal{F}(\ell)$ is the event that there are not "too many" vertices which have small weight, are close to the starting region, and are infected at time ℓ. Finally, $\mathcal{G}(j)$ is the event that all "good" events hold up to time j.

▶ **Theorem 13.** *Let ε, η, h be given as in Definition 12 and assume $\alpha > \beta - 1$. Then, for sufficiently large n,*

(a) *$\mathcal{E}(0)$ is always satisfied;*
(b) *$\Pr[\mathcal{F}(0)] \geq 1 - O(h^{-1})$;*
(c) *For all $i \geq 1$ we have $\Pr[\mathcal{E}(i) \mid \mathcal{G}(i-1)] \geq 1 - h^{-\Omega(i)}$;*
(d) *For all $\ell \geq 1$ we have $\Pr[\mathcal{F}(\ell) \mid \mathcal{G}(\ell-1)] \geq 1 - h^{-\Omega(\ell)}$;*
(e) *Whp, the events $\mathcal{G}(j)$ hold for all $j \geq 0$;*
(f) *For all $i \geq 1$ and $\ell \geq 0$, and for every fixed vertex $v = (x_v, w_v)$ such that $x_v \in \mathbb{T}^d \setminus 2^{\ell+1}\widetilde{B}_{i-1}$ and $w_v \geq w_{min}$ we have*

$$\Pr\left[v \in V^{i+\ell} \,\middle|\, \mathcal{G}(i+\ell-1)\right] \leq w_v 2^{\ell d} \widetilde{\nu}_i^{-(\zeta+\varepsilon)^{-\ell-2}/(\beta-1)}.$$

The theorem can be proven by induction on $i + \ell$, with the strategies from Section 5.1, and using the lemmas from Section 4. We next state the complementary lower bound.

For all $i, \ell \geq 0$, let $w_{i,\ell} = w_{i,\ell}(\varepsilon) := \nu^{(\zeta-\varepsilon)^{i-\ell}/(\beta-2)}$, and let U_i be the set of vertices in B_i of weight at least $w_{i,0}$. Furthermore, we denote by $\mathcal{H}(i)$ the event that in round $i+3$ all vertices in U_i are infected.

▶ **Theorem 14.** *Let $0 < \varepsilon < \zeta$ and $\eta = \eta(\varepsilon) > 0$ be sufficiently small. Assume that we are in the supercritical case, or instead that $|U_i \cap V^1| \geq k$. Then the following is true:*

(a) *Whp $|U_i| = \nu_i^{\Omega(\eta)}$ and $|U_i| = O(\nu_i)$ uniformly for all $i \geq 0$.*

(b) *Whp all the events $\mathcal{H}(i)$ occur.*

(c) *There exist constants $C_0, C_1, C_2 > 0$ such that the following holds: Let $v = (x_v, w_v)$ be any vertex with fixed position and weight and let $i, \ell \geq 0$ be such that $x_v \in B_i$ and $w_v \geq \max\{w_{i,\ell}, C_0\}$. Then for sufficiently large $n \in \mathbb{N}$,*

$$\Pr[v \in V^{i+3+\ell} \mid \mathcal{H}(0), \dots, \mathcal{H}(i)] \geq 1 - \exp\left[-C_1 \nu_i^{C_2(\zeta-\varepsilon)^{-\ell}}\right].$$

Again, the theorem can be proven inductively, with the strategies from Section 5.1.

5.3 Proof sketches for main results

In this section we highlight the main steps used to deduce the results from Section 3 from Theorem 13 and Theorem 14.

Threshold and speed of the process: Theorem 1 and Theorem 2

We split the (combined) proof into six claims:

We first show the second statement of Theorem 2, so let $0 < \varepsilon < \zeta$ be a constant.

▶ **Claim 15.** *Assume that $\alpha > \beta - 1$ and $\nu = n^{o(1)}$, then $|V^{(1-\varepsilon)i_\infty}| = o(n)$ whp.*

We define integers $i \geq 0$ and $\ell \geq 0$ such that $i + \ell \geq (1-\varepsilon)i_\infty$ and $\widetilde{\nu}_i = n^{1-o(1)}$ but $2^i \widetilde{\nu}_i = o(n)$. Then whp there are only $o(n)$ vertices inside of $2^\ell \widetilde{B}_i$, by Markov's inequality, and for vertices outside we obtain the corresponding bound from Theorem 14 (f).

In the subcritical regime, (iii) or (v), we will indeed show that whp the process does not infect any vertices in the first step and therefore terminates immediately.

▶ **Claim 16.** *$V^1 = V^0$ whp.*

Since the initial infection occurs only within B_0 and all vertices have the same probability of being infected, the number of neighbours of a given vertex is Poisson distributed, and we can bound the mean by Lemma 9. Thus we can compute the expected number of vertices in $V^{=1}$, which is $o(1)$, and this proves the claim by Markov's inequality.

Next we show that in the critical regime, (ii), with constant probability no further vertices ever become infected.

▶ **Claim 17.** *$V^1 = V^0$ with probability $\Omega(1)$.*

The number of "heavy" vertices of weight $\Theta(w_0)$ is Poisson distributed with expectation $\Theta(1)$. Thus, with probability $\Omega(1)$ there are no heavy vertices. Conditioned on this event, the calculations of the subcritical regime carry over.

On the other hand, also with probability $\Omega(1)$, at least k heavy vertices exist. Each such vertex is infected in the first round with probability $\Omega(1)$ by vertices very close to it, and all these events are positively associated. This proves the following claim.

▶ **Claim 18.** *$V^1 \cap B_0$ contains at least k heavy vertices with probability $\Omega(1)$.*

Next we assume that we are in the supercritical regime (i) or (iv), or in the critical regime (ii) where we also assume that at least k heavy vertices are infected in the first round. Then we need to show the following claim.

▶ **Claim 19.** $|V^{(1+\varepsilon)i_\infty}| = \Omega(n)$ *in expectation and whp.*

We (carefully) choose $i \geq 0$ and $\ell \geq 0$ such that $i+\ell \leq (1+\varepsilon)\, B_i = \mathbb{T}^d$ and $w_{i,l} = O(1)$. Then Theorem 14 (c) tells us immediately that every vertex of weight at least C has probability $\Omega(1)$ to be infected by time $i + \ell + 3$, and since the expected number of such vertices is $\Omega(n)$, thus proving the claim for the expectation. The whp statement follows from a small technical alteration of the previous argument, which we omit for space limitations.

Now Theorem 1 follows from Claims 16, 17, 18 and 19, while Theorem 2 is proven by Claims 15 and 19.

Infection times: Theorem 4

With our previous results, the idea is very simple: given a v vertex which satisfies the assumptions of Theorem 4 we show an lower bound on its infection time L_v by Theorem 14 and an upper bound by Theorem 13, respectively. The details become quite long and technical, and are therefore omitted.

Quarantine strategies: Theorem 7

By Theorem 13, whp there is no vertex outside of $\widetilde{B}_i$ which is infected in round i. Therefore, it suffices to (permanently) remove by the end of round i all edges that cross the boundary of $\widetilde{B}_i$, i.e. all edges in $E(\widetilde{B}_i, \mathbb{T}^d \setminus \widetilde{B}_i)$. Using an argument very similar to the one used in [15, Lemma 7.1 and Theorem 7.2], where the number of edges cutting a grid is considered, we can bound $|E(\widetilde{B}_i, 2\widetilde{B}_i)|$ the expected number of close-range edges by $\widetilde{\nu}_i^{\max\{3-\beta,1-1/d\}+o(1)}$. On the other hand using Lemma 9, we can estimate the expected number $|E(\widetilde{B}_i, \mathbb{T}^d \setminus 2\widetilde{B}_i)|$ of long-range edges by $\widetilde{\nu}_i^{3-\beta}$ and the result follows. We omit the details.

6 Concluding remarks

We have shown that in the GIRG model for scale-free networks with underlying geometry, even a small region can cause an infection that spreads through a linear part of the population. We have analysed the process in great detail, and we have determined its metastability threshold, its speed, and the time at which individual vertices becomes infected. Moreover, we have shown how a policy-maker can utilise this knowledge to enforce a successful quarantine strategy. We want to emphasise that the latter result is only a proof of concept, intended to illustrate the possibilities that come from a thorough understanding of the role of the underlying geometry in infection processes. In particular, we want to remind the reader that bootstrap percolation is not a perfect model for viral infections (though it has been used to this end), but is more adequate for processes in which the probability of transmission grows more than proportional if more than one neighbours is active, like believes spreading through a social network ("What I tell you three times is true."), or action potential spreading through a neuronal network.

Therefore, this paper is only a first step. There are many other models for the spread of an infection, most notably SIR and SIRS models for epidemiological applications, and we have much yet to learn from analysing these models in geometric power-law networks like GIRGs. From a technical point of view, it is unsatisfactory that our analysis does not include the case $\alpha \leq \beta - 1$. We believe that also in this case, the bootstrap percolation process is essentially governed by the geometry of the underlying space, only in a more complex way. Understanding this case would probably also add to our toolbox for analysing less "clear-cut" processes.

References

1 Joan Adler and Uri Lev. Bootstrap percolation: visualizations and applications. *Brazilian Journal of Physics*, 33(3):641–644, 2003.

2 William Aiello, Anthony Bonato, Colin Cooper, Jeanette Janssen, and Paweł Prałat. A spatial web graph model with local influence regions. *Internet Mathematics*, 5(1-2):175–196, 2008.

3 Michael Aizenman and Joel Lebowitz. Metastability effects in bootstrap percolation. *Journal of Physics A: Mathematical and General*, 21(19):3801, 1988.

4 Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

5 Hamed Amini. Bootstrap percolation in living neural networks. *Journal of Statistical Physics*, 141(3):459–475, 2010.

6 Hamed Amini, Rama Cont, and Andreea Minca. Resilience to contagion in financial networks. *Mathematical Finance*, 26(2):329–365, 2016.

7 Hamed Amini and Nikolaos Fountoulakis. Bootstrap percolation in power-law random graphs. *Journal of Statistical Physics*, 155(1):72–92, 2014.

8 József Balogh, Béla Bollobás, Hugo Duminil-Copin, and Robert Morris. The sharp threshold for bootstrap percolation in all dimensions. *Transactions of the American Mathematical Society*, 364(5):2667–2701, 2012.

9 József Balogh, Yuval Peres, and Gábor Pete. Bootstrap percolation on infinite trees and non-amenable groups. *Combinatorics, Probability and Computing*, 15(05):715–730, 2006.

10 Albert-László Barabási. Network science: Luck or reason. *Nature*, 489(7417):507–508, 2012.

11 Gareth Baxter, Sergey Dorogovtsev, Alexander Goltsev, and José Mendes. Bootstrap percolation on complex networks. *Physical Review E*, 82(1):011103, 2010.

12 Marián Boguñá, Dmitri Krioukov, and Kimberly Claffy. Navigability of complex networks. *Nature Physics*, 5(1):74–80, 2009.

13 Marián Boguñá, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature communications*, 1:62, 2010.

14 Milan Bradonjić and Iraj Saniee. Bootstrap percolation on random geometric graphs. *Probability in the Engineering and Informational Sciences*, 28(02):169–181, 2014.

15 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Geometric inhomogeneous random graphs. *arXiv:1511.00576*, 2015.

16 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Average distance in a general class of scale-free graphs with underlying geometry. *arXiv:1602.05712*, 2016.

17 Elisabetta Candellero and Nikolaos Fountoulakis. Bootstrap percolation and the geometry of complex networks. *Stochastic Processes and their Applications*, 126(1):234–264, 2016.

18 Shai Carmi, Shlomo Havlin, Scott Kirkpatrick, Yuval Shavitt, and Eran Shir. A model of internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences*, 104(27):11150–11154, 2007.

19 Damon Centola. The spread of behavior in an online social network experiment. *Science*, 329(5996):1194–1197, 2010.

20 John Chalupa, Paul Leath, and Gary Reich. Bootstrap percolation on a bethe lattice. *Journal of Physics C: Solid State Physics*, 12(1):L31–L35, 1979.

21 Or Cohen, Anna Keselman, Elisha Moses, Rodríguez Martínez, Jordi Soriano, and Tsvi Tlusty. Quorum percolation in living neural networks. *EPL (Europhysics Letters)*, 89(1):18008, 2010.

22 Colin Cooper, Alan Frieze, and Paweł Prałat. Some typical properties of the spatial preferred attachment model. In Anthony Bonato and Jeannette Janssen, editors, *Algorithms and Models for the Web Graph: 9th International Workshop, WAW 2012, Halifax, NS,*

Canada, June 22-23, 2012. Proceedings, pages 29–40, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

23 Sergey Dorogovtsev, Alexander Goltsev, and José Mendes. K-core organization of complex networks. *Physical review letters*, 96(4):040601, 2006.

24 Sergey Dorogovtsev and José Mendes. Evolution of networks. *Advances in Physics*, 51(4):1079–1187, 2002.

25 Paul Dreyer and Fred Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009.

26 Hafsteinn Einarsson, Johannes Lengler, Konstantinos Panagiotou, Frank Mousset, and Angelika Steger. Bootstrap percolation with inhibition. *arXiv:1410.3291*, 2014.

27 Hafsteinn Einarsson, Johannes Lengler, and Angelika Steger. A high-capacity model for one shot association learning in the brain. *Frontiers in Computational Neuroscience*, 8(140), 2014.

28 Paola Flocchini, Elena Lodi, Fabrizio Luccio, Linda Pagli, and Nicola Santoro. Dynamic monopolies in tori. *Discrete applied mathematics*, 137(2):197–212, 2004.

29 Jian Gao, Tao Zhou, and Yanqing Hu. Bootstrap percolation on spatial networks. *Scientific Reports*, 5:14662, 2015.

30 Allen Gersho and Debasis Mitra. A simple growth model for the diffusion of a new communication service. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-5(2):209–216, March 1975.

31 Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83(6):1420–1443, 1978.

32 Mark S. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6):1360–1380, 1973.

33 Randi Griffin and Charles Nunn. Community structure and the spread of infectious disease in primate social networks. *Evolutionary Ecology*, 26(4):779–800, 2012.

34 Wei Huang and Chunguang Li. Epidemic spreading in scale-free networks with community structure. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(01):P01014, 2007.

35 Emmanuel Jacob and Peter Mörters. Spatial preferential attachment networks: Power laws and clustering coefficients. *The Annals of Applied Probability*, 25(2):632–662, 2015.

36 Svante Janson, Tomasz Łuczak, Tatyana Turova, and Thomas Vallier. Bootstrap percolation on the random graph $G(n,p)$. *The Annals of Applied Probability*, 22(5):1989–2047, 2012.

37 Jeannette Janssen and Abbas Mehrabian. Rumours spread slowly in a small world spatial network. In F. David Gleich, Júlia Komjáthy, and Nelly Litvak, editors, *Algorithms and Models for the Web Graph: 12th International Workshop, WAW 2015, Eindhoven, The Netherlands, December 10-11, 2015, Proceedings*, pages 107–118, Cham, 2015. Springer International Publishing.

38 Amin Karbasi, Johannes Lengler, and Angelika Steger. Normalization phenomena in asynchronous networks. In M. Magnús Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, pages 688–700, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

39 Scott Kirkpatrick, Winfried Wilcke, Robert Garner, and Harald Huels. Percolation in dense storage arrays. *Physica A: Statistical Mechanics and its Applications*, 314(1):220–229, 2002.

40 Robert Kozma. Neuropercolation. *Scholarpedia*, 2(8):1360, 2007.

41 Renaud Lambiotte and Pietro Panzarasa. Communities, knowledge creation, and information diffusion. *Journal of Informetrics*, 3(3):180–190, 2009.

42 Cristian Moukarzel and Thomas Sokolowski. Long-range k-core percolation. *Journal of Physics: Conference Series*, 246(1):012019, 2010.

43 Javier Orlandi, Jordi Soriano, Enrique Alvarez-Lacalle, Sara Teller, and Jaume Casademunt. Noise focusing and the emergence of coherent activity in neuronal cultures. *Nature Physics*, 9(9):582–590, 2013.

44 Fragkiskos Papadopoulos, Maksim Kitsak, Ángeles Serrano, Marián Boguñá, and Dmitri Krioukov. Popularity versus similarity in growing networks. *Nature*, 489(7417):537–540, 2012.

45 Marlon Ramos, Jia Shao, Saulo Reis, Celia Anteneodo, José Andrade, Shlomo Havlin, and Hernán Makse. How does public opinion become extreme? *Scientific reports*, 5:10032, 2015.

46 Christian Schmeltzer, Jordi Soriano, Igor Sokolov, and Sten Rüdiger. Percolation of spatially constrained Erdős-Rényi networks with degree correlations. *Physical Review E*, 89(1):012116, 2014.

47 Friedhelm Schwenker, Friedrich Sommer, and Günther Palm. Iterative retrieval of sparsely coded associative memory patterns. *Neural Networks*, 9(3):445–455, 1996.

48 Munik Shrestha and Cristopher Moore. Message-passing approach for threshold models of behavior in networks. *Physical Review E*, 89(2):022805, 2014.

49 Tsvi Tlusty and Jean-Pierre Eckmann. Remarks on bootstrap percolation in metric networks. *Journal of Physics A: Mathematical and Theoretical*, 42(20):205004, 2009.

50 Tatyana Turova. The emergence of connectivity in neuronal networks: from bootstrap percolation to auto-associative memory. *Brain research*, 1434:277–284, 2012.

51 Christopher Warren, Leonard Sander, and Igor Sokolov. Geography in a scale-free network model. *Physical Review E*, 66(5):056105, 2002.

52 Duncan Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9):5766–5771, 2002.

53 Xin-Jian Xu, Xun Zhang, and José Mendes. Impacts of preference and geography on epidemic spreading. *Physical Review E*, 76(5):056109, 2007.

54 Zhi-Dan Zhao, Ying Liu, and Ming Tang. Epidemic variability in hierarchical geographical networks with human activity patterns. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(2):023150, 2012.

Sublinear-Space Bounded-Delay Enumeration for Massive Network Analytics: Maximal Cliques*

Alessio Conte[1], Roberto Grossi[2], Andrea Marino[3], and Luca Versari[4]

1 **Dipartimento di Informatica, Università di Pisa, Pisa, Italy**
`conte@di.unipi.it`
2 **Dipartimento di Informatica, Università di Pisa, Pisa, Italy**
`grossi@di.unipi.it`
3 **Dipartimento di Informatica, Università di Pisa, Pisa, Italy**
`marino@di.unipi.it`
4 **Scuola Normale Superiore, Pisa, Italy**
`luca.versari@sns.it`

Abstract

Due to the sheer size of real-world networks, delay and space become quite relevant measures for the cost of enumeration in network analytics. This paper presents efficient algorithms for listing maximum cliques in networks, providing the first sublinear-space bounds with guaranteed delay per enumerated clique, thus comparing favorably with the known literature.

1998 ACM Subject Classification E.1 Data Structures – Graphs and networks, G.2.2 Graph Theory – Graph algorithms

Keywords and phrases Enumeration algorithms, maximal cliques, network mining and analytics, reverse search, space efficiency

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.148

1 Introduction

The design of efficient algorithms for enumerating all possible solutions of a given problem dates back to the 1950s [5, 19, 35]. Enumeration algorithms have the purpose of either counting the number of solutions or listing the solutions one by one. Their study originated in the area of complexity and optimization [17, 23, 39], and then spread over several other application domains, including bioinformatics, machine learning, network analytics, and social analysis [1, 26, 33]. For instance, a number of papers described how to enumerate triangles [6, 3, 22, 31] and their generalizations such as cliques or other dense subgraphs [7, 9, 11, 12, 14, 15, 21, 25, 29, 34, 38]. Among the first problems attacked is the enumeration of maximal cliques [2, 5, 7, 18, 24, 28], where a maximal clique is a subset of pairwise connected vertices that is maximal under inclusion.

This paper focuses on two worst-case efficiency measures, namely, delay and space, that become relevant for enumeration in massive networks. The delay is the maximum latency between any two consecutively reported solutions. The space is the maximum amount of extra memory that should be allocated to enumerate all the solutions, besides the amount required by the input graph.

* Work partially supported by the Italian Ministry of Education, University, and Research (MIUR) under PRIN 2012C4E3KT research project AMANDA – Algorithmics for MAssive and Networked DAta.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 148; pp. 148:1–148:15

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

Motivation. Let $G(V, E)$ be an undirected connected graph, represented with ordered adjacency lists, where $n = |V|$ is the number of vertices and $m = |E|$ is the number of edges. Although it is known that the number α of maximal cliques in real-world networks is much smaller than the exponentially many possible subsets of vertices [32], it happens that α is still large for massive networks. In this scenario the notion of delay provides a guarantee on the maximum time that a postprocessing algorithm has to wait before the next enumerated maximal clique is produced: letting $t(n, m)$ be the delay, we observe that not only the total time is output-sensitive, i.e. $O(\alpha\, t(n, m))$ plus the setup cost, but we also guarantee that listing the next solution takes $O(t(n, m))$ time in the worst case. This is a stronger notion than average throughput (e.g., number of cliques per second), which is obtained by dividing α by the total time. The former implies the latter, but not vice versa. We observe that the delay has been already used in many papers, e.g. [9, 11, 12, 23, 25, 36], as a worst-case measure.

Space is another measure that has been considered when enumerating solutions, e.g. [13, 10, 41]. Modern CPUs have multiple cores, where each core has a very fast – but small – private cache, with a shared last-level cache and a shared slow random access memory. Consider multiple enumeration threads running simultaneously on the same massive graph in the shared memory: modern machines have very large shared memory compared to the private cores, so massive graphs can be stored in shared memory but not in the private cores. If the memory footprint of each thread is small and fits the private cache of a core, the shared memory access bottleneck is only caused by accessing the graph itself, and not the private data. To make a concrete example, the private cache for CPUs in today's commodity desktops is few hundreds of kilobytes while the random access memory can host several terabytes and networks contain millions of vertices and edges. For example, network EU-2005 (Section 7) contains $n = 850$ thousand vertices, $m = 16.1$ million edges, and $\alpha = 5.7$ million cliques: here, the space of known algorithms ranges from 3 to 164 megabytes, so their working set does not fit the private cache (Table 2).

We remark that delay and space are somehow related measures. An algorithm with good overall time can accumulate solutions in the shared memory or store them temporarily in a file for a subsequent phase of postprocessing. If space is limited (especially in fast memory), this approach cannot be taken into account.

Furthermore, as most current output-sensitive approaches have a recursive nature, they can easily reach $\Theta(n)$ nesting levels in the worst case even for sparse graphs, thus using the stack should be avoided: indeed, just storing one memory word per recursion level kills sublinearity. This motivates the search for enumeration algorithms that provably use sublinear space and have good delay bounds. Some space-efficient algorithms [13, 10, 41] work well in practice for real-world networks but cannot guarantee sublinear space.

We remark that this paper does not address cache-efficient or cache-oblivious algorithms in the parallel or distributed setting, which is worth investigating in future work. What is emphasized here is that small footprint enumeration algorithms have more chances to reduce memory contention and memory bandwidth issues when run on modern processors.

Our results. We provide the first algorithms with sublinear space and bounded delay for enumerating the maximal cliques when the vertices of G are provided in degeneracy order. This means that there exists an integer d, as small as possible, such that each vertex has d or fewer neighbors appearing later in the ordering: d is called the degeneracy of G, a well-known sparsity measure [13, 15, 40, 41], and is equivalently defined as the smallest integer such that every nonempty subgraph of G has at least one vertex of degree $\leq d$. The degeneracy

Table 1 Bounds for maximal clique enumeration, where $q-1 \leq d \leq \Delta \leq n-1 \leq m$, $d = O(\sqrt{m})$. †: it does not list cliques, but outputs a compressed representation. $^{+}$: h is the smallest integer such that $|\{v \in V : |N(v)| \geq h\}| \leq h$, where $d \leq h \leq \Delta$. *: it uses matrix multiplication. A lower bound in the column for the delay means that there exists a family of graphs with that delay.

Algorithm	Time			Space
	Setup	Delay	Overall	
Bron-Kerbosch [7]	$O(m)$	unbounded	unbounded	$O(n+q\Delta)$
Tomita et al. [34]†	$O(m)$	$\Omega(n^3)$	$O(3^{n/3})$	$O(n+q\Delta)$
Eppstein et al. [16]	$O(m)$	$\Omega(3^{n/6})$	$O(d(n-d)3^{d/3})$	$O(n+d\Delta)$
Johnson et al. [23]	$O(mn)$	$O(mn)$	$\alpha\, O(mn)$	$O(\alpha n)$
Tsukiyama et al. [36]	$O(n^2)$	$O((n^2-m)n)$	$\alpha\, O((n^2-m)n)$	$O(n^2)$
Chiba-Nishizeki [11]	$O(m)$	$O(md)$	$\alpha\, O(md)$	$O(m)$
Makino-Uno [25]	$O(mn)$	$O(\Delta^4)$	$\alpha\, O(\Delta^4)$	$O(m)$
Chang et al. [9]$^{+}$	$O(m)$	$O(\Delta h^3)$	$\alpha\, O(\Delta h^3)$	$O(m)$
Makino-Uno [25]*	$O(n^2)$	$O(n^{2.37})$	$\alpha\, O(n^{2.37})$	$O(n^2)$
Comin-Rizzi [12]*	$O(n^{5.37})$	$O(n^{2.09})$	$\alpha\, O(n^{2.09})$	$O(n^{4.27})$
This paper	$\tilde{O}(m)$	$\tilde{O}(qd(\Delta+qd))$	$\alpha\, \tilde{O}(qd(\Delta+qd))$	$O(q)$
This paper	$\tilde{O}(m)$	$\tilde{O}(\min\{md, qd\Delta\})$	$\alpha\, \tilde{O}(\min\{md, qd\Delta\})$	$O(d)$

n = #vertices $\quad$ Δ = max degree $\quad$ α = #maximal cliques
m = #edges $\quad$ d = degeneracy $\quad$ q = largest clique size

ordering can be computed in $O(n+m)$ time by repeatedly removing the vertex of minimum degree from G. Also, it can be proved that $d = O(\sqrt{m})$, and that the size q of the maximum clique in G and the maximum degree in the graph Δ satisfy $q-1 \leq d \leq \Delta \leq n-1$.

The last two rows in Table 1 report our bounds in terms of the above parameters, where the $\tilde{O}()$ notation ignores $\log^{O(1)}(n+m)$ factors. We observe that our bounds are expressed in terms of the parameters q, d, Δ instead of m, n, whenever possible, as they are actually smaller than m or n. For example, network EU-2005 has $q = 387$, $d = 388$ and $\Delta = 68\,963$. Also, since both q and d are always $O(\sqrt{m})$, our space is provably sublinear (around 20 kilobytes for EU-2005!). Note that Δ is not always sublinear in the graph size as real-world networks are sparse and could have $\Delta = \Theta(n)$, as shown in Table 2 (see also [13]). Also, our $O(q)$ space is close to optimal as we have to single out a subset of q vertices from G.

The other rows in Table 1 report the bounds for the main results in the state of the art (see below for a discussion). The setup time is the preprocessing cost before starting to list the solutions, and ours is comparable to that of previous results. As for the delay, the cost in the last row is asymptotically smaller in many cases, except for dense graphs, where matrix multiplication based algorithms [12, 25] are preferable (but massive networks can hardly be processed by quadratic space algorithms). As for the overall time, we have a similar improvement for output-sensitive bounds where the α term appears. Moreover, we observe that [16] has great time performance in practice (see Section 7) but it is not output-sensitive as α does not appear in the complexity, and cannot guarantee sublinear space. Summing up, our algorithms can compete with the state of the art when suitably implemented, with the additional bonus of guaranteeing small space and bounded delay.

Related work. The idea of using small space in enumeration algorithms is not new, as witnessed by the notion of "compactness" introduced in Fukuda [17]: however, his goal is not sublinear space as in our paper, but polynomially bounded space in terms of the input size and the maximum output size of a solution. It is also worth mentioning that the class of CAT (constant amortized time) enumeration algorithms described in Ruskey's book [30] seems to be very promising but our algorithms cannot fall within this class as their amortized cost per solution is non-constant.

Our algorithms are based on the reverse search paradigm introduced by Avis and Fukuda [4] as it has been conceived to be space-efficient by its authors. Also, we reuse some of the machinery introduced by Tsukiyama et al. [36] and Makino and Uno [25]. In the reverse search the solutions are the nodes of a virtual digraph, for which a "successor" function is defined to jump from one solution to the other. (Gély et al. [20] study new combinatorial properties of this virtual digraph.) A spanning tree represents all the solutions, where the depth corresponds to the number of nested levels of the corresponding recursion. To achieve sublinear space, our algorithms employ the stateless reverse search to avoid using the stack, plus other properties that exploit the structure of the maximal clique enumeration problem.

Turning to the state of the art for the maximal clique enumeration problem, Table 1 summarizes the main results. The papers by Bron and Kerbosch [7] and Tsukiyama et al. [36] have defined the main lines of research for algorithms using polynomial space, and they are currently at the heart of many other algorithms. The Bron-Kerbosch algorithm relies on a backtracking scheme that is adopted in several efficient algorithms [16, 34, 41]. The algorithm by Tsukiyama et al. has been originally conceived for the enumeration of maximal independent sets, which is a problem equivalent to that of maximal cliques, and inspired at least in part the algorithm by Johnson et al. [23], which produces solutions in lexicographic order but requires exponential space (see also [20]). The approach has been subsequently adapted to clique enumeration by Chiba-Nishizeki [11], and Makino-Uno [25] has reinterpreted [36] in the paradigm of reverse search.

Bron-Kerbosch scheme. The Bron-Kerbosch based algorithms are a popular choice for enumerating cliques due to their simplicity and good performance in practice. The original version [7] does not provide any guarantee. The version in [34] guarantees a total running time of $O(3^{n/3})$, which is optimal for Moon-Moser graphs as they have $3^{n/3}$ cliques [8, 27]. The version in [16] further refines and improves the work for sparse graphs, which may have up to $(n-d)3^{d/3}$ cliques, by producing an algorithm with $O(d(n-d)3^{d/3})$ time. All of these approaches use $O(n+q\Delta)$ space to store sets of candidates and visited vertices. It is possible to modify these algorithms to decrease their space usage (e.g. by modifying the data structure in [16]) but, to the best of our knowledge, with state of the art techniques they would still require $\Omega(\Delta)$, which is not sublinear as Δ can be $\Theta(n)$ in sparse graphs. Algorithms that follow this scheme are characterized by their complexity being related to the worst-case number of cliques in a graph (instead of α), and give no guarantees on the cost per solution nor the delay: these can be shown to be both $\Omega(n^3)$ for [34], while [16] can have a delay of $\Omega(3^{n/6})$ for some families of graphs.

Tsukiyama et al. scheme. The motivation behind algorithms in this class is to achieve an output-sensitive cost that is proportional to the number α of maximal cliques times a function that depends on the graph parameters. The original algorithm by Tsukiyama et al. [36] is a backtracking procedure that enumerates maximal independent sets in $O(mn)$ time per solution. As a maximal independent set is a maximal clique in the complementary

graph, this gives an algorithm for enumerating maximal cliques in $O((n^2 - m)n)$ time per solution. The adaptation of the algorithm to maximal cliques has been improved by Chiba and Nishizeki [11], who bring the delay down to $O(md)$:[1] this algorithm is still based on a stateful backtracking procedure in which the recursion tree has depth $\Omega(n)$, making the required space $\Omega(n)$. Makino and Uno [25] take a step towards statelessness by adapting [36] and [23] to the paradigm of reverse search. The algorithm is still a stateful recursive approach, which takes $\Omega(n)$ space (the depth of the recursion tree). Differently from its predecessors, each node of the tree corresponds to a unique maximal clique whose children can be computed as a function of the graph and the clique itself. The Makino-Uno algorithm is provided in two versions, one combinatorial with delay $O(\Delta^4)$ and one based on matrix multiplication with delay $O(n^{2.37})$. The former has been improved to $O(\Delta h^3)$ by Chang et al. [9], where h is the smallest integer such that $|\{v \in V : |N(v)| \geq h\}| \leq h$: since the reverse search is not stateless, here the space remains $\Omega(n)$. The latter has been improved to $O(n^{2.09})$ by Comin and Rizzi [12] using matrix multiplication but requiring higher space usage and setup time.

2 Preliminaries

Let $G(V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$. We assume that G is connected and represented using ordered adjacency lists. A clique is a subset $K \subseteq V$ of pairwise connected vertices: we will use K to denote both this subset of vertices and the subgraph induced by them. Given G, let Δ be the maximum vertex degree, d the degeneracy, and q be the maximum clique size, where $q - 1 \leq d \leq \Delta \leq n - 1 \leq m$ and $d = O(\sqrt{m})$; also, let $v_1 \dots v_n$ be the vertices of V labelled in a degeneracy ordering (see the introduction). We denote by $V_{\leq i}$ the set of vertices $v_1 \dots v_i$. Let us define $N(v)$ as the set of neighbors of v and $N_>(v)$ as $\{x \in N(v) : x > v\}$. We define $N_<(v)$ analogously. Note that $|N_>(v)| \leq d$ as the graph is labelled in degeneracy ordering. Given a set of vertices $A \subset V$, we define $N(A) = \bigcap_{v \in A} N(v)$ as the set of neighbors common to all the vertices in A. We call *heads* the vertices v such that $N_<(v) = \emptyset$. Moreover, for any $v \in V$, $A_{\leq v} = A \cap V_{\leq v}$ and $A_{<v} = A_{\leq v} \setminus \{v\}$. Given two set of vertices A and B, we say that $A < B$ if A is lexicographically smaller than B. Given a set $X \subseteq V$, we define $\textsc{complete}(X)$ as the lexicographically smallest maximal clique that contains X.

▶ **Lemma 1.** *If $X_1 \subseteq X_2$, then $\textsc{complete}(X_1) \leq \textsc{complete}(X_2)$.*

Proof. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the set of maximal cliques containing respectively X_1 and X_2. Clearly $\mathcal{C}_1 \supseteq \mathcal{C}_2$, and as $\textsc{complete}(X)$ returns the lexicographically smallest maximal clique that contains X we have $\textsc{complete}(X_1) = \min(\mathcal{C}_1) \leq \min(\mathcal{C}_2) = \textsc{complete}(X_2)$. ◀

3 Reverse Search Revisited

We sketch a reverse search algorithm for the maximal clique enumeration that revises and simplifies the algorithm by Makino and Uno [25], which is itself a reinterpretation of the works of Tsukiyama et al. in [36] and Johnson et al. in [23] for maximal independent sets.

The rationale of the algorithm can be summarized as follows. Suppose we iteratively add the vertices of G to a graph G', which is initially empty. Each time a vertex v is added, the new maximal cliques can be computed by looking at the maximal cliques of G': an existing clique $K \subseteq G'$ can be extended partially or totally by v. Of course computing all

[1] The work actually exploits the arboricity, but we use d for simplicity as the arboricity is $\Theta(d)$.

Algorithm 1: Enumerate all maximal cliques (a.k.a. revisited Makino-Uno)

Input : Graph $G(V, E)$ where vertices are labeled in degeneracy ordering
Output : Maximal cliques in G

Let $v_1, v_2, \dots v_n \in V$ be the vertices labeled in degeneracy ordering, such that $v_1, \dots, v_j$ are heads, where j is the number of heads ($1 \leq j \leq n$).

```
for i ∈ {1, ..., j} do
    R_i ← COMPLETE({v_i})
    SPAWN(R_i)
Function COMPLETE(K)
    choose any v ∈ K
    foreach increasing w ∈ N(v) do
        if K ⊆ N(w) then
            K ← K ∪ {w}
    return K
Function SPAWN(K)
    CAND ← { w ∈ ⋃_{u∈K} N_>(u) \ K : w > PI(K) }
    foreach v ∈ CAND do
        K'_v ← K_<v ∩ N(v)
        D ← COMPLETE(K'_v ∪ {v})
        if COMPLETE(K'_v) = K and D_<v = K'_v then
            SPAWN(D)
```

the cliques of G in this way requires keeping most of the cliques of the graph in memory (as for independent sets in [23]), which takes exponential space. In order to avoid to store the cliques, we address the following question: given the clique K, maximal for G', which vertices in $G \setminus G'$ expand K and which cliques do they produce in G? This "production" relationship gives us the "successor" function of the reverse search paradigm. Technicalities are needed to avoid expanding K with candidates leading to the same clique more than once.

It is worth observing that, differently from [25], our algorithm assumes that the vertices of G are given in degeneracy ordering and begins the recursion from a set of root cliques (instead of the lexicographically minimum one), where each root R_i corresponds to COMPLETE run on the head v_i. Moreover, we alter the given degeneracy ordering in such a way that the heads are at the beginning. This is always possible: as heads have no backward edges, when moved backwards they will not change the number of forward edges of any vertex (see Section 6).

The pseudocode is shown in Algorithm 1. The function SPAWN makes use of the notion of parent index, borrowed from [25]:

▶ **Definition 2** (Parent Index). Given a maximal clique K, $\text{PI}(K)$ is the smallest x such that $\text{COMPLETE}(K_{\leq x}) = K$.

The CAND set contains the vertices that partially extend K and that are greater than $\text{PI}(K)$. For each vertex in CAND we try to generate a child clique; the check in SPAWN corresponds to conditions (c) and (d) from Lemma 2 in [25], which guarantees that each child clique is only generated once. We remark that each call of SPAWN returns at least a clique.

Space usage and delay. Referring to Algorithm 1, we observe that the COMPLETE function only needs to store the set K, whose size is $O(q)$, and performs $O(\Delta)$ iterations that take $\tilde{O}(q)$ time each. The cost of an iteration on vertex w is also bounded by $\tilde{O}(|N(w)|)$, so the cost of all the iterations is bounded by $\tilde{O}(m)$. Since PI can be computed by running COMPLETE q times, we can conclude the following:

▶ **Lemma 3.** *Function* COMPLETE *in Algorithm 1 takes $O(q)$ space and $\tilde{O}(\min\{q\Delta, m\})$ time. Moreover, computing* PI *takes $O(q)$ space and $\tilde{O}(q \min\{q\Delta, m\})$ time.*

For the sake of clarity, we will refer to the *vertices* in G and to the *nodes* in the recursive tree induced by our approach. The space requirement of each recursive node is bounded by

the size of CAND, as all other lines take $O(q)$ space. As CAND $\subseteq V$ is the union of up to q sets of vertices each of size at most d, we have $|\text{CAND}| \leq \min\{qd, n\}$. The recursion depth of Algorithm 1 is $O(n)$, for a total space requirement of $O(n \min\{qd, n\})$.

Consider now the delay of Algorithm 1. Note that selecting the next head takes constant time as heads are contiguous. By making use of alternative output [37], as we have no dead ends, the delay of the algorithm is bounded by the cost of $O(1)$ recursive nodes. The function SPAWN performs $|\text{CAND}|$ iterations, whose cost is dominated by COMPLETE. The total cost of the loop is thus $\tilde{O}(\min\{qd, n\} \min\{q\Delta, m\})$, which corresponds to the total cost of the recursion node as it dominates the cost of computing CAND. As a consequence we have some new interesting bounds shown in Lemma 4, which we improve in the rest of the paper.

▶ **Lemma 4.** *Algorithm 1 uses $O(n \min\{qd, n\})$ space and has $\tilde{O}(\min\{qd, n\} \min\{q\Delta, m\})$ delay.*

4 Improved Algorithm

Algorithm 1 does not meet our space requirements; still, it is the starting point to build our space efficient scheme. The first issue is its recursive nature: in function SPAWN, each time a recursive call is performed the *status* of the current call needs to be saved. As a result, we require the space needed by each recursive node multiplied by the height of the recursion tree. Note that the standard stack-based transformation of recursive programs into iterative ones, as done in [9], does not solve the issue, as the stack size would be $\Omega(n)$. We will therefore navigate implicitly the recursion tree induced by the reverse search, without using a stack. The other important issue to achieve sublinear space is the CAND set, whose size can be $\min\{qd, n\}$. Hence, we need to traverse CAND without materializing it.

To address the first issue, we represent the state of the computation with the pair:

- the current clique K
- the "bookmark" vertex v in CAND for K (where initially $v = \text{PI}(K)$).

We remark that the bookmark vertex allows us to resume the computation in the parent without the need of storing information for each recursive level.

▶ **Fact 1.** *The state $\langle K, v\rangle$ requires $O(q)$ space.*

Furthermore, we implicitly traverse the recursion tree using these navigation primitives:

- IS-ROOT(K) checks if K is the root of current recursion tree.
- PARENT-STATE(K) returns the state for the parent node of K in the recursion tree.
- GET-NEXT-CAND(K, v) finds the candidate following v in CAND for the current K.
- CHILD-EXISTS(K, v) checks whether the current state will lead to a child maximal clique.

Algorithm 2 shows how to implement and use the above primitives, and Lemmas 5, 6, 7, 10 prove their correctness.

▶ **Lemma 5.** *Let K be any maximal clique examined in Algorithm 1: K is a root iff $\text{PI}(K) = \min(K)$. As a corollary, roots have no parent.*

Proof. Let v_i be the head of the recursion tree containing K and $R_i = \text{COMPLETE}(v_i)$ the relative root. By definition we have that v_i is a head iff $N_<(v_i) = \emptyset$, so a head must be the smallest vertex of any clique. We then have $v_i = \min(R_i)$, and since $\min(D) \in K$ for any child D of K, there cannot be heads other than v_i in the current recursion tree. From this it immediately follows that no other roots are found in the subtree of the root R_i, and hence roots have no parent.

Algorithm 2: Improved enumeration of maximal cliques

Assume $\min(\emptyset) = \texttt{null}$, and adopt the following shorthands for maximal clique K: sub-clique $K'_v \equiv K_{<v} \cap N(v)$ and vertex set $B_K \equiv \{u \in V \setminus K : K_{<u} \subseteq N(u)\}$.

```
Function IMPROVED-SPAWN(K)
    v ← PI(K)
    while true do
        childless ← true
        while v ← GET-NEXT-CAND(K, v) ≠null do
            if CHILD-EXISTS(K, v) then
                K ← COMPLETE(K'_v ∪ {v})
                childless ← false
                break
        if childless then
            if IS-ROOT(K) then return
            else
                ⟨K, v⟩ ← PARENT-STATE(K)

Function IS-ROOT(K)
    return PI(K) = min(K)
Function CHILD-EXISTS(K, v)
    return N(K'_v) ∩ (B_K ∪ N_<(v)) = ∅
Function GET-NEXT-CAND(K,v)
    return min{w ∈ ⋃_{u∈K} N_>(u) \ K :
                w > v}
Function PARENT-STATE(K)
    v ← PI(K)
    return ⟨COMPLETE(K_<v), v⟩
```

We now show that $N_<(\min(K)) = \emptyset$ and $K = \text{COMPLETE}(\{\min(K)\})$ iff $\text{PI}(K) = \min(K)$. This concludes the proof as the first condition correspond to the definition of a root ($\min(K)$ being the corresponding head). Since $\{\min(K)\} = K_{\leq \min(K)}$, we have $\text{PI}(K) = \min(K)$ iff $K = \text{COMPLETE}(\{\min(K)\})$ by definition of PI. Moreover, $K = \text{COMPLETE}(\{\min(K)\})$ implies $N_<(\min(K)) = \emptyset$: indeed, if there was $w \in N_<(\min(K))$, COMPLETE would add w to $\{\min(K)\}$, making it different from K. ◀

The next lemma states that given a clique D, the parent index allows us to identify the clique that generated D and the vertex in CAND used to produce D.

▶ **Lemma 6.** *Let K, D, and v be defined as in* SPAWN *when the recursive call is performed on D. Then* $\text{PARENT-STATE}(D) = \langle K, v\rangle$ *in* IMPROVED-SPAWN.

Proof. We prove that, given a maximal clique D which is not a root, the parent of D in the computational tree is $\text{COMPLETE}(D_{<\text{PI}(D)})$ and the vertex used by the algorithm SPAWN to produce D is $\text{PI}(D)$. Since K is the parent of D, we have $D = \text{COMPLETE}(K'_v \cup \{v\})$. From the conditions tested on K'_v and D, we have $\text{COMPLETE}(D_{<v}) = \text{COMPLETE}(K'_v) = K$. Now we only need to prove that $v = \text{PI}(D)$. We have that $D = \text{COMPLETE}(K'_v \cup \{v\}) = \text{COMPLETE}(D_{\leq v}) > \text{COMPLETE}(D_{<v}) = K$, applying Lemma 1. From this it follows that for any $w < v$, $\text{COMPLETE}(D_{\leq w}) \leq \text{COMPLETE}(D_{<v}) = K < D$, thus $v = \text{PI}(D)$. ◀

Since the candidates are examined in increasing order, when returning from the current child of K, we are able to provide all the remaining children of K. Indeed let v be $\text{PI}(D)$: as a consequence of Lemma 6 we know that D was generated from K using candidate v and that $K = \text{COMPLETE}(D_{<v})$. By using the next lemma, we can provide and test all the remaining candidates, i.e. the ones greater than v.

▶ **Lemma 7.** *For a given maximal clique K, let* $\text{CAND} = \{w \in \bigcup_{x\in K} N_>(x) \setminus K : w > \text{PI}(K)\}$ *in* SPAWN *and let $z_1, \ldots, z_r$ be the sequence of vertices generated by* GET-NEXT-CAND *in* IMPROVED-SPAWN. *Then* $\text{CAND} = \{z_1, \ldots, z_r\}$.

Proof. We give the proof by induction. The first time GET-NEXT-CAND is invoked v is $\text{PI}(K)$, meaning that z_1 is the minimum element of CAND. Let z_j be the last vertex generated by

GET-NEXT-CAND and let us assume that $z_1, \ldots, z_j$ are the first j vertices of CAND. Then $\min\{w \in \bigcup_{u \in K} N_>(u) \setminus K \ : \ w > z_j\}$ is the the $(j+1)$-th element of CAND if $|\text{CAND}| > j$, `null` otherwise. ◀

The check done in CHILD-EXISTS(K, v) is equivalent to the check done in SPAWN. We will prove this in Lemma 10 using Lemmas 8 and 9.

▶ **Lemma 8.** *$N(K'_v) \cap B_K = \emptyset$ is equivalent to* COMPLETE$(K'_v) = K$*, where* $B_K = \{u \in V \setminus K : K_{<u} \subseteq N(u)\}$.

Proof. Let us first prove that if $\exists w \in N(K'_v) \cap B_K$ then COMPLETE$(K'_v) \neq K$. Note that w is adjacent to all the vertices in $K_{<w}$ and in K'_v. COMPLETE(K'_v) will iteratively add to K'_v the smallest vertex z that is a neighbor of all the vertices in K'_v, so clearly $z \leq w$. If $z \notin K$ we have COMPLETE$(K'_v) \neq K$; if $z \in K$ then $z \in K_{<w}$, thus $z \in N(w)$ and w is still a candidate for the COMPLETE procedure. As w remains a candidate when $z \in K$, the process will eventually add either w or another $z \notin K$. Hence, COMPLETE$(K'_v) \neq K$.

We now prove that if COMPLETE$(K'_v) \neq K$ then $N(K'_v) \cap B_K \neq \emptyset$. Let z be the first vertex not in K selected by COMPLETE(K'_v). Since all vertices in $K_{<z}$ were added to K'_v before z, we have $K_{<z} \subseteq N(z)$ and hence $z \in B_K$. Moreover, since z has been selected by the COMPLETE procedure, $K'_v \subseteq N(z)$, which implies $z \in N(K'_v)$. It follows that $N(K'_v) \cap B \supseteq \{z\} \neq \emptyset$. ◀

▶ **Lemma 9.** *$N(K'_v) \cap N_<(v) = \emptyset$ is equivalent to* COMPLETE$(K'_v \cup \{v\})_{<v} = K'_v$.

Proof. We prove that if $N(K'_v) \cap N_<(v) \neq \emptyset$ then COMPLETE$(K'_v \cup \{v\})_{<v} \neq K'_v$. Let z be the smallest vertex in $N(K'_v) \cap N_<(v)$. Note that $z \notin K'_v$, since $z \in N(K'_v)$. The first iteration of COMPLETE$(K'_v \cup \{v\})$ selects the smallest vertex in $N(K'_v \cup \{v\})$, which is z, since $N(K'_v \cup \{v\}) = N(K'_v) \cap N(v)$. Since $z < v$, we have $z \in$ COMPLETE$(K'_v \cup \{v\})_{<v}$, which implies COMPLETE$(K'_v \cup \{v\})_{<v} \neq K'_v$.

We now prove that if COMPLETE$(K'_v \cup \{v\})_{<v} \neq K'_v$ then $N(K'_v) \cap N_<(v) \neq \emptyset$. Note that $K'_v \subseteq$ COMPLETE$(K'_v \cup \{v\})_{<v}$ since $K'_v \subseteq V_{<v}$. Let $z = \min(\text{COMPLETE}(K'_v \cup \{v\})_{<v} \setminus K'_v)$. Since z has been selected by function COMPLETE, we have $z \in N(K'_v \cup \{v\})$, that is $z \in N(K'_v) \cap N(v)$. Since $z < v$, we have $N(K'_v) \cap N_<(v) \supseteq \{z\} \neq \emptyset$. ◀

▶ **Lemma 10.** *$N(K'_v) \cap (B_K \cup N_<(v)) = \emptyset$ iff* COMPLETE$(K'_v) = K$ *and* COMPLETE$(K'_v \cup \{v\})_{<v} = K'_v$

Using Lemmas 5, 6, 7 and 10, we finally obtain the following result:

▶ **Lemma 11.** *Function* SPAWN *in Algorithm 1 and function* IMPROVED-SPAWN *in Algorithm 2 are equivalent.*

Proof. Setting $D =$ COMPLETE$(K'_v \cup \{v\})$ we observe that IMPROVED-SPAWN(K) simulates the preorder traversal of the recursion tree induced by SPAWN(K). When all the children of the current clique K have been explored during the traversal, *childless* is set to true. In this case, if K is the root of the current tree there are no more maximal cliques to be generated with the given head v_i; otherwise, the state of the parent is restored. ◀

5 Analysis

Analogously to Section 3, the delay of Algorithm 2 is the sum of the running times of all the iterations corresponding to the same K. Specifically, the number of iterations of the while loop is $|\text{CAND}| \leq \min\{qd, n\}$ (see Lemma 7). The space usage of IMPROVED-SPAWN, thanks

to its statelessness, corresponds to that of a single iteration. In order to give space and time bounds, let us analyze the costs of our navigation primitives.

Since computing $\text{PI}(K)$ dominates the space and time costs of functions IS-ROOT and PARENT-STATE, we use the bounds in Lemma 3. The next candidate can be obtained from K and the bookmark v as follows: for each element $x \in K$, perform a binary search in $N(x)$ to obtain the smallest vertex greater than v; the minimum of these vertices is the next candidate. Thus we obtain the bounds below:

▶ **Lemma 12.** IS-ROOT(K) *and* PARENT-STATE(K) *take* $\tilde{O}(q \min\{q\Delta, m\})$ *time and* $O(q)$ *space.* GET-NEXT-CAND *uses* $\tilde{O}(q)$ *time and constant space.*

Next we give a careful analysis of CHILD-EXISTS, the dominating cost of IMPROVED-SPAWN.

Cost of testing if a child exists. Function CHILD-EXISTS makes use of the sets B_K and $N(K'_v)$. Due to our memory constraints, we cannot store $N(K'_v)$ explicitly, since it would take $\Omega(\Delta)$ space. For this reason, we need to iterate over $N(K'_v)$ without materializing it.

Analogously, we cannot store $B_K = \{v \in V \setminus K : K_{<v} \subseteq N(v)\}$, whose size can be $\Omega(n)$. The following lemma will allow us to overcome this issue:

▶ **Lemma 13.** *The set* B_K *is equal to* $V_{<\min(K)} \cup \{u \in N_>(\min(K)) \setminus K : K_{<u} \subseteq N(u)\}$.

Proof. vertices in $V_{<\min(K)}$ are in B_K as $K_{<\min(K)} = \emptyset$, thus they can be stored implicitly. All other vertices must clearly be forward neighbors of $\min(K)$. ◀

We denote the non-trivial part of B_K, i.e. $\{u \in N_>(\min(K)) \setminus K : K_{<u} \subseteq N(u)\}$, as B'_K. We can compute it by testing $|N_>(\min(K))| \leq d$ candidates in $O(|K|) = O(q)$ time.

▶ **Lemma 14.** *Computing (or iterating over)* B'_K *can be done in* $\tilde{O}(qd)$ *time.*

We will now analyze the cost of iterating over $N(K'_v)$, and two possible ways of managing B_K (storing it or iterating it implicitly) leading to two different bounds.

Iterating over $N(K'_v)$. Iterating over $N(K'_v)$ using constant space can be done as follows. Let w be the vertex of K'_v with lowest degree: iterate over the vertices $z \in N(w)$ and for each check whether $K'_v \subseteq N(z)$. This costs $\tilde{O}(|N(w)||K'_v|) = \tilde{O}(\sum_{y \in K'_v} |N(y)|)$, since w is the lowest degree element of K'_v.

▶ **Lemma 15.** *The total cost of iterating over all of the sets* $N(K'_v)$ *for* v *in* $\bigcup_{x \in K} N_>(x)$ *is* $\tilde{O}(d \min\{q\Delta, m\})$ *time.*

Proof. Let $C_K = \bigcup_{x \in K} N_>(x)$. Since the cost of iterating over a given $N(K'_v)$ is bounded by $\tilde{O}(\sum_{y \in K'_v} |N(y)|)$, the following steps prove the lemma:

$$\begin{aligned}
\sum_{v \in C_K} \sum_{y \in K \cap N_<(v)} |N(y)| &= \sum_{v \in C_K} \sum_{y \in K} |N(y)| I_{\{y \in N_<(v)\}} \\
&= \sum_{y \in K} \sum_{v \in C_K} |N(y)| I_{\{v \in N_>(y)\}} \\
&= \sum_{y \in K} \sum_{v \in N_>(y)} |N(y)| \\
&= \sum_{y \in K} |N_>(y)| \cdot |N(y)| \\
&\leq d \sum_{y \in K} |N(y)| \leq d \min\{q\Delta, m\}
\end{aligned}$$

◀

Algorithm 3: In-place algorithm for moving heads to the beginning of V

Input : $G(V, E)$, with V in degeneracy ordering and E as ordered adjacency lists $L_1 \dots L_n$.
Output : $G(V, E)$ with V relabeled in degeneracy ordering so that the j heads are $v_1, \dots, v_j$.

```
1 s, e ← n + 1
2 foreach s in decreasing order from n to 1 do                    // Reorder, relabel N_>(v)
3     if s is not a head then
4         e ← e − 1
5         foreach v < s in L_s do replace s with e in L_v
6         replace all v < s with 0 in L_s
7         swap L_s and L_e
8 foreach v in decreasing order from n to 1 do                    // Relabel N_<(v)
9     foreach x ≠ 0 in L_v do replace the rightmost 0 in L_x with v
```

Storing B_K. By applying Lemma 13, since we can check in constant time whether a vertex is in $V_{<\min(K)}$, we only need to store B'_K, whose size is $O(d)$ by definition. The computation of B'_K, which costs $\tilde{O}(qd)$ time applying Lemma 14, is done once for K and once for each child. Hence its cost is payed at most twice for each clique.

Observe that the cost of CHILD-EXISTS is dominated by that of the computation of $N(K'_v)$, since testing $N(K'_v) \cap B_K = \emptyset$ and $N(K'_v) \cap N_<(v) = \emptyset$ can be done in $\tilde{O}(1)$ time for each element of $N(K'_v)$. Applying Lemma 15, we can conclude that:

▶ **Lemma 16.** *Function* CHILD-EXISTS *can be implemented such that the cumulative cost of all the calls to* CHILD-EXISTS(K, v) *for a fixed K is $O(d)$ space and $\tilde{O}(d \min\{q\Delta, m\})$ time.*

Iterating over B_K. In order to avoid storing B'_K, we can compute the intersection between $N(K'_v)$ and B_K iterating over both sets. The iterator for B'_K works as described in Lemma 14. Since the elements of both $N(K'_v)$ and B'_K are iterated in increasing order, the cost of computing their intersection for each call of CHILD-EXISTS is the sum of the costs of the two iterations. Consider the sum of these costs over all the calls of CHILD-EXISTS: applying Lemma 14 and Lemma 15, we obtain a total cost of $\tilde{O}(d \min\{q\Delta, m\} + qd \min\{qd, n\})$, since $|\text{CAND}| \leq \min\{qd, n\}$. As a result, we have the following lemma:

▶ **Lemma 17.** *Function* CHILD-EXISTS *can be implemented such that the cumulative cost of all the calls to* CHILD-EXISTS(K, v) *for a fixed K is $O(q)$ space and $\tilde{O}(d \min\{q\Delta, m\} + qd \min\{qd, n\})$ time.*

The final algorithm corresponds to plugging Algorithm 2 into Algorithm 1, i.e. replacing SPAWN with IMPROVED-SPAWN. In order to show that the final algorithm takes $O(q)$ or $O(d)$ space as well, we should also perform the setup described in Section 6. Using Algorithm 3 as setup, and applying Fact 1 and Lemmas 12, 16, and 17, we obtain our final result (recall that $q - 1 \leq d \leq \Delta \leq n - 1 \leq m$ and $d = O(\sqrt{m})$).

▶ **Theorem 18.** *Let G be an undirected connected graph with n vertices and m edges, whose adjacency lists are ordered and whose vertices are labeled in degeneracy ordering. Let Δ be the maximum degree of its vertices, q the size of its largest clique, and d its degeneracy. Then there exists an algorithm that lists all the maximal cliques in G that has $\tilde{O}(m)$ setup time, $O(q)$ space usage, and $\tilde{O}(qd(\Delta + qd))$ delay. The latter bound improves to $\tilde{O}(qd\Delta)$ if space usage is increased to $O(d)$. Space is always sublinear in the size of G.*

Table 2 Experimental results of our comparison with output-sensitive algorithms in the state of the art. For the graph statistics (upper part), we refer to Table 1. For the comparison (lower part) we have considered the total time (TIME), the delay (DELAY), and the space (MEM).

	DBLP-2008			AMAZON-0505			IN-2004			EU-2005		
n	511 163			410 236			1 353 703			862 664		
m	1 871 070			2 439 436			13 126 172			16 138 468		
α	447 563			1 034 135			3 384 922			5 727 256		
Δ	576			2 760			21 869			68 963		
d	114			10			488			388		
q	115			11			489			387		
ALGO.	TIME *sec*	DELAY *ms*	MEM *MiB*	TIME *sec*	DELAY *ms*	MEM *MiB*	TIME *sec*	DELAY *ms*	MEM *MiB*	TIME *sec*	DELAY *ms*	MEM *MiB*
CN	>2h	475	97.56	>2h	636	78.26	>2h	5 285	258.21	>2h	4 077	164.54
MU	39.5	18	3.01	16.9	22	3.01	6 102.7	3 691	52.62	>2h	11 395	30.52
CXQ	21.2	4	0.11	60.7	12	0.25	>2h	6 004	1.02	>2h	621	3.15
RALG2	1.8	0.6	1.57	3.1	0.8	0.81	66.6	401	6.46	237.4	245	3.16
ALG2	2.3	15	0.01	4.4	0.9	0.01	100.7	410	0.03	363.8	277	0.02

CN: Chiba-Nishizeki [11] MU: Sparse graph Makino-Uno [25] CXQ: Chang et al. [9]
RALG2: Recursive Algorithm 2 ALG2: Algorithm 2

6 Setup

Algorithm 3 gives some details on how to modify the degeneracy ordering of the vertices so as to place the heads at the beginning in $\tilde{O}(m)$ time and $O(1)$ space. As already noted, the resulting order is still a degeneracy ordering. The main idea is to maintain a *sliding window* over the list of adjacency lists. The window incrementally collects the heads while moving from the last vertex to the first one in the ordering, so that its final position is the beginning of the ordering. The window is moved by swapping the greatest vertex before the window, which is called s and the greatest vertex in the window, which is called e. In case s is a head, the window is not shifted but simply enlarged to include s. While moving the window, the occurrences of the labels of the swapped vertices s and e must be updated accordingly. To this aim just the occurrences of s in $N_>(v)$ are relabeled, for each v neighbor of s. At the end, the backward neighborhoods can be updated by looking at the forward ones.

It is straightforward to see that the time needed by Algorithm 3 is $\tilde{O}(m)$. Indeed, in the loop in line 2, each adjacency list is iterated at most once. The loop in line 8 has the same time bound, i.e. $\tilde{O}(m)$: it traverses the adjacency lists of all vertices performing replace operations, which can be done in logarithmic time on ordered lists. The constant space usage follows from the fact that just $O(1)$ variables are stored.

7 Experimental Evaluation

In this section, we report some preliminary experiments on some large real-world networks that have been taken from LASAGNE (`lasagne-unifi.sourceforge.net/`). Our computing platform is a 24-core machine with Intel(R) Xeon(R) CPU E5-2620 v3 at 2.40GHz, with 128GB of shared memory. The operating system is a Ubuntu 14.04.2 LTS, with Linux kernel version 3.16.0-30. The code has been written in `C++` and forced to run on a single core.

Table 2 reports the results of the comparison between our algorithm and existing algorithms which use at most linear space and are output-sensitive, i.e. the ones providing a bounded delay or cost per solution, namely CN, MU, and CXQ (see Table 1). We limited the running time to two hours. The algorithm RALG2 refers to a recursive version of Algorithm 2. ALG2 refers to Algorithm 2 which takes $O(q)$ memory by using Lemma 17 (similar results

can be found for the iterative version which uses Lemma 16). For each algorithm and for each graph, we report the total time, the maximum delay found while running, and the memory usage (excluding the input size). Note that both RALG2 and ALG2 are significantly faster than CN, MU, and CXQ. It is worth observing that the results highlight how the delay of RALG2 and ALG2 depends on d and q as shown in Theorem 18.

The running times of ALG2 are in general higher than RALG2, even though its performance is competitive. On the other hand, ALG2 uses the smallest amount of memory, namely always less than 0.03 MiB even when these graphs have hundreds of thousands of vertices and millions of edges. The most striking result is for EU-2005, having more than 850 thousands of vertices and more than 16 millions of edges, where our algorithm uses just 0.02 MiB. Finally, observe that the memory seems to be proportional to d and q, since IN-2004 and EU-2005 have relatively higher memory consumption.

For the sake of completeness, we also considered the state of the art for Bron-Kerbosch based algorithms. In particular, we ran the algorithm in [15, 16] using the code provided by the authors (we are grateful to them). Even though its cost per solution can be higher than ours and its delay can be exponential, this algorithm is on the average 3.7 times faster than ALG2. On the other hand, as this algorithm uses linear memory, its memory consumption is on the average 878.9 times larger than the memory of ALG2.

References

1 Nesreen K. Ahmed, Jennifer Neville, Ryan A. Rossi, and Nick G. Duffield. Efficient graphlet counting for large networks. In *2015 IEEE International Conference on Data Mining, ICDM 2015*, pages 1–10. IEEE, 2015.

2 E. A. Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM J. Comput.*, 2(1):1–6, 1973.

3 Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

4 David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, 1996.

5 Edward Bierstone. Cliques and generalized cliques in a finite linear graph. *Unpublished report*, 1960.

6 Andreas Björklund, Rasmus Pagh, Virginia Vassilevska Williams, and Uri Zwick. Listing triangles. In *Automata, Languages, and Programming – 41st International Colloquium, ICALP 2014, Proceedings, Part I*, pages 223–234, 2014.

7 Coenraad Bron and Joep Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973.

8 Frédéric Cazals and Chinmay Karande. A note on the problem of reporting maximal cliques. *Theor. Comput. Sci.*, 407(1-3):564–568, 2008.

9 Lijun Chang, Jeffrey Xu Yu, and Lu Qin. Fast maximal cliques enumeration in sparse graphs. *Algorithmica*, 66(1):173–186, 2013.

10 James Cheng, Linhong Zhu, Yiping Ke, and Shumo Chu. Fast algorithms for maximal clique enumeration with limited memory. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD, pages 1240–1248, 2012.

11 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985.

12 Carlo Comin and Romeo Rizzi. An improved upper bound on maximal clique listing via rectangular fast matrix multiplication. *CoRR*, abs/1506.01082, 2015. URL: `http://arxiv.org/abs/1506.01082`.

13 Alessio Conte, Roberto De Virgilio, Antonio Maccioni, Maurizio Patrignani, and Riccardo Torlone. Finding all maximal cliques in very large social networks. In *Proceedings of*

the 19th International Conference on Extending Database Technology, EDBT 2016., pages 173–184, 2016.

14 Nan Du, Bin Wu, Liutong Xu, Bai Wang, and Pei Xin. Parallel algorithm for enumerating maximal cliques in complex network. In *Mining Complex Data*, pages 207–221. Springer, 2009.

15 David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *ISAAC (1)*, pages 403–414, 2010.

16 David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics*, 18, 2013.

17 Komei Fukuda. Note on new complexity classes ENP, EP and CEP. `https://www.inf.ethz.ch/personal/fukudak/old/ENP_home/ENP_note.html`, 1996. Accessed: 2016-02-17.

18 Leonhard Gerhards and Wolfgang Lindenberg. Clique detection for nondirected graphs: Two new algorithms. *Computing*, 21(4):295–322, 1979.

19 Eo N. Gilbert. Enumeration of labelled graphs. *Canad. J. Math*, 8(1):05–411, 1956.

20 Alain Gély, Lhouari Nourine, and Bachir Sadi. Enumeration aspects of maximal cliques and bicliques. *Discrete Applied Mathematics*, 157(7):1447–1459, 2009.

21 Christopher J. Henry and Sheela Ramanna. *Transactions on Rough Sets XVI*, chapter Maximal Clique Enumeration in Finding Near Neighbourhoods, pages 103–124. Springer, 2013.

22 Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978.

23 David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988. `doi:10.1016/0020-0190(88)90065-8`.

24 H. C. Johnston. Cliques of a graph-variations on the Bron-Kerbosch algorithm. *International Journal of Parallel Programming*, 5(3):209–238, 1976.

25 Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *Algorithm Theory-SWAT 2004*, pages 260–272. Springer, 2004.

26 R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298:824–827, 2002.

27 John W. Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3(1):23–28, 1965.

28 Gordon D. Mulligan and Derek G. Corneil. Corrections to Bierstone's algorithm for generating cliques. *J. ACM*, 19(2):244–247, 1972.

29 Long Pan and Eunice E. Santos. An anytime-anywhere approach for maximal clique enumeration in social network analysis. In *SMC*, pages 3529–3535. IEEE, 2008.

30 Frank Ruskey. *Combinatorial Generation*. 2003.

31 Thomas Schank and Dorothea Wagner. Finding, counting and listing all triangles in large graphs, an experimental study. In *Experimental and Efficient Algorithms, 4th InternationalWorkshop, WEA 2005, Proceedings*, pages 606–609, 2005.

32 Matthew C. Schmidt, Nagiza F. Samatova, Kevin Thomas, and Byung-Hoon Park. A scalable, parallel algorithm for maximal clique enumeration. *Journal of Parallel and Distributed Computing*, 69(4):417–428, 2009.

33 Nino Shervashidze, S V N Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten M. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009*, volume 5 of *JMLR Proceedings*, pages 488–495, 2009.

34 Etsuji Tomita, Akira Tanaka, and Haruhisa Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theor. Comput. Sci.*, 363(1):28–42, 2006.

35 Horace M. Trent. A note on the enumeration and listing of all possible trees in a connected linear graph. *Proceedings of the National Academy of Sciences*, 40(10):1004–1007, 1954.

36 Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM J. Comput.*, 6(3):505–517, 1977.

37 Takeaki Uno. Two general methods to reduce delay and change of enumeration algorithms. *National Institute of Informatics (in Japan) Technical Report E*, 4:2003, 2003.

38 Takeaki Uno. An efficient algorithm for solving pseudo clique enumeration problem. *Algorithmica*, 56(1):3–16, 2008.

39 Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.

40 Kunihiro Wasa, Hiroki Arimura, and Takeaki Uno. Efficient enumeration of induced subtrees in a k-degenerate graph. In Hee-Kap Ahn and Chan-Su Shin, editors, *Algorithms and Computation: 25th International Symposium, ISAAC 2014, Proceedings*, pages 94–102, Cham, 2014. Springer International Publishing.

41 Yanyan Xu, James Cheng, Ada Wai-Chee Fu, and Yingyi Bu. Distributed maximal clique computation. In *Big Data (BigData Congress), 2014 IEEE International Congress on*, pages 160–167. IEEE, 2014.

On the Size and the Approximability of Minimum Temporally Connected Subgraphs*

Kyriakos Axiotis[†1] **and Dimitris Fotakis**[2]

1 **Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA**
kaxiotis@mit.edu

2 **School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece**
fotakis@cs.ntua.gr

Abstract

We consider *temporal graphs* with discrete time labels and investigate the size and the approximability of minimum temporally connected spanning subgraphs. We present a family of minimally connected temporal graphs with n vertices and $\Omega(n^2)$ edges, thus resolving an open question of (Kempe, Kleinberg, Kumar, JCSS 64, 2002) about the existence of sparse temporal connectivity certificates. Next, we consider the problem of computing a minimum weight subset of temporal edges that preserve connectivity of a given temporal graph either from a given vertex r (r-MTC problem) or among all vertex pairs (MTC problem). We show that the approximability of r-MTC is closely related to the approximability of Directed Steiner Tree and that r-MTC can be solved in polynomial time if the underlying graph has bounded treewidth. We also show that the best approximation ratio for MTC is at least $O(2^{\log^{1-\varepsilon} n})$ and at most $O(\min\{n^{1+\varepsilon}, (\Delta M)^{2/3+\varepsilon}\})$, for any constant $\varepsilon > 0$, where M is the number of temporal edges and Δ is the maximum degree of the underlying graph. Furthermore, we prove that the unweighted version of MTC is APX-hard and that MTC is efficiently solvable in trees and 2-approximable in cycles.

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

Keywords and phrases Temporal Graphs, Temporal Connectivity, Approximation Algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.149

1 Introduction

Graphs and networks are ubiquitous in Computer Science, as they provide a natural and useful abstraction of many complex systems (e.g., transportation and communication networks) and processes (e.g., information spreading, epidemics, routing), and also of the interaction between individual entities or particles (e.g., social networks, chemical and biological networks). Traditional graph theoretic models assume that the structure of the network and the strength of interaction are time-invariant. However, as observed in e.g., [3, 19], in many applications of graph theoretic models, the availability and the weights of the edges are actually time-dependent. For instance, one may think of information spreading and distributed computation in dynamic networks (see e.g., [6, 12, 19, 20]), of mobile adhoc and sensor networks (see e.g.,

* The full version of this work is available at http://arxiv.org/abs/1602.06411

† This work was done while K. Axiotis was with the School of Electrical and Computer Engineering, National Technical University of Athens, Greece.

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 149; pp. 149:1–149:14

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

[22]), of transportation networks and route planning (see e.g., [3, 15]), of epidemics, biological and ecological networks (see e.g., [17, 19]), and of influence systems and coevolutionary opinion formation (see e.g., [5, 9]).

Several variants of time-dependent graphs have been suggested as abstractions of such settings and computational problems (see e.g., [6] and the references therein). No matter the particular variant, the main research questions are usually related either to optimizing or exploiting temporal connectivity or to computing short time-respecting paths (see e.g., [1, 2, 3, 13, 15, 19, 21]). In this work, we adopt the simple and natural model of temporal graphs with discrete time labels [19] (and its extension with multiple labels per edge [21]), and study the existence of dense minimally connected temporal graphs and the approximability of temporally connected spanning subgraphs with minimum total weight.

Temporal Graphs and Temporal Connectivity. A *temporal graph* is defined on a time-invariant set of n vertices. Each (undirected) edge e is associated with a set of discrete time labels denoting when e is available. If every edge is associated with a single time label, as in [19], the temporal graph is *simple*. An edge e available at time t comprises a temporal edge (e, t) and there is a positive weight $w(e, t)$ associated with it. A (resp. strict) *temporal* (or *time-respecting*) path is a sequence of temporal edges with non-decreasing (resp. increasing) time labels. So, temporal paths respect the time availability constraints of the edges.

Given a source vertex r, a temporal graph is (temporally) *r-connected* if there is a temporal path from r to any other vertex. A temporal graph is (temporally) *connected* if there exists a temporal path from any vertex to any other vertex. We study the existence of dense minimally connected temporal graphs and the optimization problems of computing a minimum weight subset of temporal edges that preserve either r-connectivity or connectivity. We refer to these optimization problems as (Minimum) Single-Source Temporal Connectivity (or r-MTC, in short) and (Minimum) All-Pairs Temporal Connectivity (or MTC, in short). They arise as natural generalizations of Minimum Arborescence and Minimum Spanning Tree in temporal networks, and to the best of our knowledge, their approximability has not been determined so far (but see [1, 18] for some results on variants or special cases).

Previous Work and Motivation. The model of simple temporal graphs with discrete time labels was introduced in [19]. It is essentially equivalent to the model of scheduled networks [3], where each edge is available in a time interval. [3, 19] investigated how time availability restrictions on the edges affect certain graph properties. Berman [3] presented an algorithm for reachability by temporal paths and proved that an analogue of the max-flow-min-cut theorem holds for temporal graphs. Kempe et al. [19] focused on vertex-disjoint temporal paths and showed that Menger's theorem does not generalize to temporal graphs. They also identified a simple forbidden topological minor for Menger's theorem in temporal graphs. Mertzios at al. [21] introduced multiple labels per edge and studied the number of temporal edges required for a temporal design to guarantee certain graph properties. Interestingly, they proved that a variant of Menger's theorem, which also takes time into account, holds in all temporal graphs. A key technical tool in [3, 19, 21] is the time-expanded version of a temporal graph, which reduces reachability, edge-disjoint path and vertex-disjoint path questions in temporal graphs to similar questions in standard directed graphs.

Our motivation comes from a natural open question in [19, Section 6]. Attempting an analogy between spanning trees of (standard undirected) graphs and connectivity certificates of temporal graphs, Kempe et al. asked whether any simple temporal graph admits a sparse connectivity certificate. They observed that any r-connected temporal graph has a time-

respecting arborescence with $n-1$ edges that serves as a sparse r-connectivity certificate. For all-pairs temporal connectivity, however, minimum temporal connectivity certificates may have different sizes. Kempe et al. observed that an allocation of time labels to the edges of the hypercube makes it minimally temporally connected. Hence, there are temporal graphs on n vertices with temporal connectivity certificates of $\Omega(n \log n)$ edges. An open question in [19, Section 6] was to determine the tightest function $c(n)$ for which any temporally connected graph on n vertices has a temporal connectivity certificate with at most $c(n)$ edges. A trivial upper bound on $c(n)$ is $O(n^2)$, since taking n time-respecting arborescences, each rooted at a different vertex, results in a temporally connected subgraph. Kempe et al. observed that if we consider strict temporal paths and allow for the same time label at different edges, $c(n) = \Omega(n^2)$ (e.g., consider K_n with the same time label on all edges). Nevertheless, for connectivity with strict temporal paths and distinct time labels, the best known lower bound on $c(n)$ is $\Omega(n \log n)$ ([1], again by a labeling of the hypercube).

Contribution. In this work, we resolve the open question of [19] and derive upper and lower bounds on the approximability of Single-Source and All-Pairs Temporal Connectivity.

In Section 3, we construct a family of simple temporal graphs with $3n$ vertices and roughly $n(n+9)/2$ edges which are almost minimally temporally connected, in the sense that the removal of any subset of $5n$ edges results in a disconnected temporal graph [1] (Theorem 1). Hence, we show that $c(n) = \Theta(n^2)$ (i.e., there are graphs with dense minimum temporal connectivity certificates), thus resolving the open question of [19]. Our construction is essentially best possible and can be easily extended to connectivity by strict temporal paths (with distinct time labels on the edges). An interesting feature of our construction (and an indication of its tightness) is that slightly increasing the time label of a single temporal edge results in a temporal connectivity certificate with $O(n)$ edges!

Given the huge gap on the size of temporal connectivity certificates, it is natural to ask about the complexity and the approximability of Single-Source and All-Pairs Temporal Connectivity. Previous work shows that we can decide if a temporal graph is connected in polynomial time (see e.g., [1, 3, 19]) and that Single-Source Temporal Connectivity can be solved in polynomial time in the unweighted case. Another interesting observation is that if we use the time-expanded version of a temporal graph for Minimum Temporal Connectivity, the resulting optimization problems are quite similar to Group Steiner Tree problems. In fact, this observation serves as the main intuition behind several of our results.

In Section 4, we show that the polynomial-time approximability of Single-Source Temporal Connectivity (r-MTC) is closely related to the approximability of the classical Directed Steiner Tree problem. Using a transformation from Directed Steiner Tree to r-MTC (Theorem 2) and [16, Theorem 1.2], we show that r-MTC cannot be approximated within a ratio of $O(\log^{2-\varepsilon} n)$, for any constant $\varepsilon > 0$, unless NP $\subseteq$ ZTIME$(n^{\text{poly}\log n})$. Our transformation also implies that any $o(n^{\varepsilon})$-approximation for r-MTC would improve the best known approximation ratio of Directed Steiner Tree. On the positive side, using a transformation from r-MTC to Directed Steiner Tree and the algorithm of [7], we obtain a polynomial-time $O(n^{\varepsilon})$-approximation, for any constant $\varepsilon > 0$, and a quasipolynomial-time $O(\log^3 n)$-approximation for r-MTC (Theorem 3). We also show that r-MTC can be solved in polynomial time if the underlying graph has bounded treewidth (Theorem 4).

[1] Based on Theorem 1, we can easily obtain a family of minimally connected temporal graphs with $\Omega(n^2)$ edges (e.g., we remove temporal edges from the graph, as long as connectivity is preserved). For simplicity and clarity, we avoid presenting a tight (but more complicated) construction of dense minimally connected temporal graphs, and stick to almost minimal graphs in the proof of Theorem 1.

In Section 5, we consider the approximability of All-Pairs Temporal Connectivity (MTC). Theorem 3 implies an $O(n^{1+\varepsilon})$-approximation for MTC (Corollary 5). An approximation-preserving reduction to Directed Steiner Forest and [14, Theorem 1.1] imply a polynomial-time $O((\Delta M)^{2/3+\varepsilon})$-approximation for MTC, where M is the number of temporal edges and Δ is the maximum degree of the underlying graph (Theorem 6). If M is quasilinear and Δ is polylogarithmic, we obtain an $O(n^{2/3+\varepsilon})$-approximation. On the negative side, a reduction from Symmetric Label Cover implies that MTC cannot be approximated within a factor of $O(2^{\log^{1-\varepsilon} n})$ unless NP $\subseteq$ DTIME$(n^{\text{poly}\log n})$ (Theorem 7, see also [10, Section 4]). We also show that the unweighted version of MTC is APX-hard (Theorem 8).

In Section 6, we show that MTC can be solved optimally, in polynomial time, if the underlying graph is a tree (Theorem 9), and that MTC is 2-approximable if the underlying graph is a cycle (Theorem 10, but it is open whether MTC remains NP-hard for cycles).

For clarity, we focus on connectivity by (non-strict) temporal paths. However, all our results can be extended (with small changes in the proofs and with the same approximation guarantees and running times) to the case of connectivity by strict temporal paths.

Comparison to Previous Work. Akrida et al. [1] study connectivity by strict temporal paths. Allocating distinct time labels to the hypercube, they obtain a minimal temporally connected graph with $\Omega(n \log n)$ edges. They also show that any allocation of distinct labels to K_n results in a temporal graph that is not minimally connected. However, they do not give any lower bound on the size of temporal connectivity certificates for K_n. Our Theorem 1 improves on the lower bound of [1] from $\Omega(n \log n)$ to $\Omega(n^2)$. [1] also shows that computing the maximum number of edges that are redundant for temporal connectivity is APX-hard.

Huang et al. [18] consider the Single-Source (but not the All-Pairs) version of Minimum Temporal Connectivity in simple scheduled networks [3]. They show that the problem is APX-hard. Using a transformation to Directed Steiner Tree, they show that the approximation guarantees of [7] carry over to Single-Source Temporal Connectivity for scheduled networks. Although the approximation guarantees are the same, the reduction of [18] is slightly different and less general than ours in Theorem 3 (which we discovered independently). In addition to the approximability result, we present strong inapproximability bounds for r-MTC and show that it is polynomially solvable for graphs with bounded treewidth.

Erlebach et al. [13] study the problem of computing a shortest exploration schedule of a temporal graph, i.e., a shortest strict temporal walk that visits all vertices. They prove that it is NP-hard to approximate the shortest exploration schedule within a factor of $O(n^{1-\varepsilon})$, for any $\varepsilon > 0$, and construct temporal graphs whose exploration requires $\Theta(n^2)$ steps. Since the notion of exploration schedules is much stronger than (r-)connectivity, their results do not have any immediate implications for r-MTC and MTC (e.g., the $\Theta(n^2)$-explorable graphs of [13, Lemma 4] admit a temporally connected subgraph with $O(n)$ edges).

2 The Model and Preliminaries

Throughout, we let $[k] \equiv \{1, \ldots, k\}$, for any integer $k \geq 1$. An (edge weighted) *temporal graph* $\mathcal{G}(V, E, L)$ with vertex set V, edge set E and lifetime L is a sequence of (undirected edge-weighted) graphs $(G_t(V, E_t, w_t))_{t\in[L]}$, where $E_t \subseteq E$ is the set of edges available at time t and $w_t(e)$ (or $w(e, t)$) is the nonnegative weight of each edge $e \in E_t$. We often write $\mathcal{G}$ or $\mathcal{G}(V, E)$, for brevity. A temporal graph $\mathcal{G}$ is *unweighted* if $w(e, t) = 1$ for all $e \in E_t$ and all $t \in [L]$. For each edge $e \in E_t$, we say that (e, t) is a *temporal edge* of $\mathcal{G}$. For each edge $e \in E$, $L_e = \{t \in [L] : e \in E_t\}$ denotes the set of time units (or *time labels*) when e is available. A temporal graph is *simple* if $|L_e| = 1$ for all edges $e \in E$.

We let n be the number of vertices and $M = \sum_e |L_e|$ be the number of temporal edges of $\mathcal{G}$. For temporal connectivity problems, we can assume that at least one temporal edge is available in each time unit, and thus, $L \leq M$. The (static) graph $G(V, E)$ is the *underlying graph* of $\mathcal{G}$. We say that $\mathcal{G}$ has some (non-temporal) graph theoretic property (e.g., is a tree, a cycle, a clique, has bounded treewidth) if the underlying graph G has this property.

For a vertex set S, $G[S]$ (resp. $\mathcal{G}[S]$) is the underlying (resp. temporal) graph induced by S. A spanning subgraph $\mathcal{G}'$ of a temporal graph $\mathcal{G} = (G_t(V, E_t, w_t))_{t\in[L]}$ is a sequence of graphs $(G'_t(V, E'_t, w_t))_{t\in[L]}$ such that $E'_t \subseteq E_t$. The total weight of $\mathcal{G}'$ is $\sum_{t\in[L]} \sum_{e\in E'_t} w(e, t)$.

A *temporal* (or *time-respecting*) path is an alternating sequence of vertices and temporal edges $(v_1, (e_1, t_1), v_2, (e_2, t_2), \ldots, v_k, (e_k, t_k), v_{k+1})$, such that $e_i = \{v_i, v_{i+1}\} \in E_{t_i}$, for all $i \in [k]$, and $1 \leq t_1 \leq t_2 \leq \cdots \leq t_k$. A temporal path is *strict* if $t_1 < t_2 < \cdots < t_k$. Such a temporal path is from v_1 to v_{k+1} (or a temporal $v_1 - v_{k+1}$ path).

A temporal graph $\mathcal{G}$ is (temporally) *r-connected*, for a given source $r \in V$, if there is a temporal path from r to any vertex $u \in V$. A temporal graph $\mathcal{G}$ is (temporally) *connected*, if there is a temporal path from u to v for any ordered pair $(u, v) \in V \times V$. If all temporal paths are strict, $\mathcal{G}$ is strictly connected (or strictly r-connected). An *(r-)connectivity certificate* of $\mathcal{G}$ is any spanning subgraph of $\mathcal{G}$ that is also (r-)connected.

Given a temporal graph $\mathcal{G}$ and a source vertex r, the problem of (Minimum) *Single-Source Temporal Connectivity* (r-MTC) is to compute a temporally r-connected spanning subgraph of $\mathcal{G}$ with minimum total weight. The optimal solution to r-MTC is a simple temporal graph whose underlying graph is a tree (see e.g., [19, Section 6]). Given a temporal graph $\mathcal{G}$, the problem of (Minimum) *All-Pairs Temporal Connectivity* (MTC) is to compute a temporally connected spanning subgraph of $\mathcal{G}$ with minimum total weight.

An algorithm A has *approximation ratio* $\rho \geq 1$ for a minimization problem, such as Single-Source and All-Pairs Temporal Connectivity, if for any instance I, the cost of A on I is at most ρ times I's optimal cost.

Directed Steiner Tree and Forest. To understand the approximability of r-MTC and MTC, we use reductions from and to Directed Steiner Tree and the Directed Steiner Forest.

Given a directed edge-weighted graph $G(V, E)$ with n vertices, a source $r \in V$ and a set of k terminals $S \subseteq V$, the Directed Steiner Tree (DST) problem asks for a subgraph of G that includes a directed path from r to any vertex in S and has minimum total weight. The best known algorithm for DST is due to Charikar et al. [7] and achieves an approximation ratio of $O(k^\varepsilon)$, for any constant $\varepsilon > 0$, in polynomial time, and of $O(\log^3 k)$ in quasipolynomial time. On the negative side, [16, Theorem 1.2] shows that DST cannot be approximated within a factor $O(\log^{2-\varepsilon} n)$, for any constant $\varepsilon > 0$, unless NP $\subseteq$ ZTIME$(n^{\text{poly}\log n})$.

Given a directed edge-weighted graph $G(V, E)$ with n vertices and m edges, and a collection $D \subseteq V \times V$ of k ordered vertex pairs, the Directed Steiner Forest (DSF) problem asks for a subgraph of G that contains an $s - t$ path for each $(s, t) \in D$ and has minimum total weight. [14] presents a polynomial-time $O(n^\varepsilon \min\{n^{4/5}, m^{2/3}\})$-approximation for DSF, for any constant $\varepsilon > 0$.

3 A Lower Bound on the Size of Temporal Connectivity Certificates

In this section, we construct an infinite family of simple temporal graphs with $\Theta(n)$ vertices and lifetime $\Theta(n)$ such that any temporal connectivity certificate has $\Omega(n^2)$ edges. Our construction is essentially best possible, since any temporal graph with n vertices and lifetime L admits a connectivity certificate with $O(\min\{n^2, nL\})$ edges.

▶ **Theorem 1.** *For any even $n \geq 2$, there is a simple connected temporal graph with $3n$ vertices, $n(n+9)/2-3$ edges and lifetime at most $7n/2$, so that the removal of any subset of $5n$ edges results in a disconnected temporal graph.*

Proof sketch. For any even n, we construct a simple connected temporal graph $\mathcal{G}$ with $\Theta(n)$ vertices and $\Theta(n^2)$ edges so that virtually any edge is essential for temporal connectivity.

We start with describing the construction. For any even n, $\mathcal{G}$ consists of $3n$ vertices partitioned into three sets $A = \{a_1, \ldots, a_n\}$, $H = \{h_1, \ldots, h_n\}$ and $C = \{c_1, \ldots, c_n\}$, with n vertices each. The underlying graph $G[A]$ is the complete graph K_n and comprises the *dense* part of the construction with $\Theta(n^2)$ edges. The edges of $G[A]$ are partitioned into $n/2$ edge-disjoint paths $p_1, \ldots, p_{n/2}$. Each path p_i has length $n-1$ and spans all vertices in A (see Figure 1.a). All edges of each path p_i have time label i.

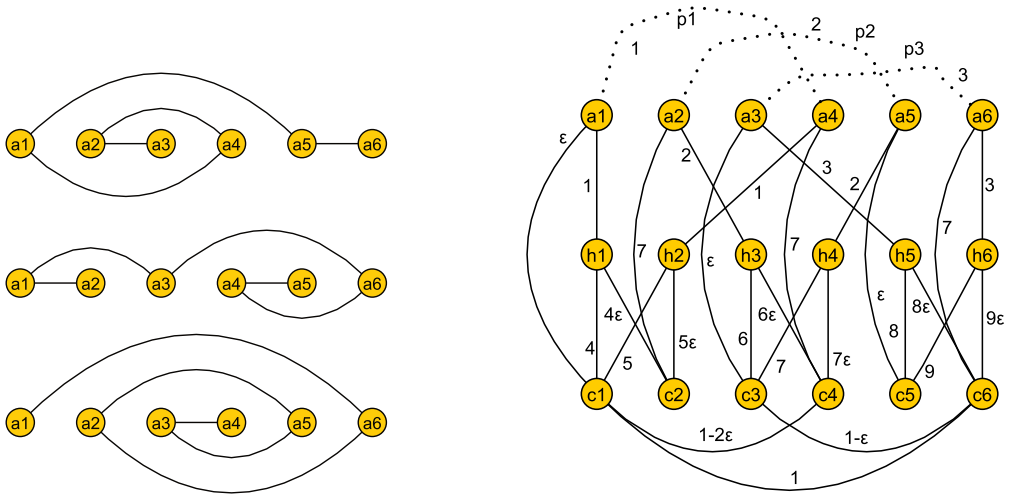
The vertices of H comprise the *intermediate* part of the construction. There are no edges with both endpoints in H. For every $i \in [n/2]$, one endpoint of the path p_i is connected to h_{2i-1} and the other endpoint is connected to h_{2i}. Both edges have time label i.

The vertices of C form the *interconnecting* part of the construction. For each $i \in [n/2]$, we refer to c_{2i-1} (resp. c_{2i}) as the *entry vertex* (resp. the *exit vertex*) for the vertices h_{2i-1} and h_{2i}. There are two edges connecting c_{2i-1} to h_{2i-1} and h_{2i} with labels $n/2+2i-1$ and $n/2+2i$, respectively, and two edges connecting c_{2i} to h_{2i-1} and h_{2i} with labels $(n/2+2i-1)\epsilon$ and $(n/2+2i)\epsilon$, respectively, for some fixed $\epsilon \in (0, 1/(4n))$. We also connect the vertices of C to each other. For every $i \in [n/2-2]$, there are edges connecting c_{2i-1} to c_{2i+2} and to c_n, and a single edge connecting c_{n-3} to c_n. To allocate time labels to these edges, we order them in decreasing order of their endpoint with higher index, breaking ties by ordering them in increasing order of their endpoint with lower index, i.e., the order is $\{c_1, c_n\}$, $\{c_3, c_n\}, \ldots, \{c_{n-3}, c_n\}$, $\{c_{n-5}, c_{n-2}\}$, $\{c_{n-7}, c_{n-4}\}, \ldots, \{c_1, c_4\}$. The time label of the k-th edge in this order is $1-(k-1)\epsilon$. Finally, for every $i \in [n/2]$, there are an edge with time label ϵ connecting the vertex c_{2i-1} to the vertex a_{2i-1} in A and an edge with time label $n+1$ connecting the vertex c_{2i} to the vertex a_{2i} in A (see also Figure 1.b).

The total number of edges is $n(n+9)/2-3$, the number of different time labels is at most $7n/2$, and each edge has a single label.

Next, we present the intuition and discuss the main technical claims. The construction is based on the collection $p_1, \ldots, p_{n/2}$ of $n/2$ edge-disjoint paths, where all edges in each path p_i have label i. Extending each path p_i to vertices h_{2i-1} and h_{2i}, we get a path that connects h_{2i} to h_{2i-1} (and vice versa) and to all vertices in A at time i. Moreover, different time labels make these paths essentially independent of each other, in the sense that if a temporal walk begins and ends at time i, it can use only edges with label i (i.e., only edges of this path) to connect h_{2i} to h_{2i-1}. Formalizing this intuition, we can show that the unique temporal path from h_{2i} to h_{2i-1} is through path p_i. Therefore, all edges of $G[A]$ must be present in any temporally connected spanning subgraph of $\mathcal{G}$. To achieve a dense underlying graph $G[A]$, we observe that the collection of $n/2$ edge-disjoint paths can be defined so that they go through the same n vertices, in a different order each (see Figure 1.a). This describes the main intuition behind our construction and explains how the dense and the intermediate parts work. The only problem now is that H-vertices with high indices, e.g., h_n, cannot reach H-vertices with low indices, e.g., h_1. The vertices in the interconnecting part C serve to carefully connect each h_j to each h_i, with $j > i+1$, without destroying the property that the only temporal path from h_{2i} to h_{2i-1} is through path p_i.

For every vertex pair $h_{2i-1}, h_{2i} \in H$, we introduce a vertex pair $c_{2i-1}, c_{2i} \in C$. As an entry vertex, c_{2i-1} is connected to h_{2i-1} and h_{2i} with "large" labels (larger than $n/2$). Hence, starting from the rest of $\mathcal{G}$, we can reach h_{2i-1} and h_{2i} through c_{2i-1}, but we cannot continue



(a) Partition into 3 Hamiltonian paths. (b) Putting the 3 parts together.

Figure 1 The temporal graph constructed in the proof of Theorem 1 for $n = 6$.

to the edges of p_i (with label $i \leq n/2$). As an exit vertex, c_{2i} is connected to h_{2i-1} and h_{2i} with "very small" labels (at most $1/4$). Thus, starting from c_{2i}, we can reach first h_{2i-1} and h_{2i}, and then all vertices in A and any vertex h_j with index $j > 2i$. Moreover, to avoid creating a temporal path from h_{2i} to h_{2i-1}, the label of the edge $\{h_{2i}, c_{2i-1}\}$ (resp. $\{h_{2i}, c_{2i}\}$) is larger than the label of the edge $\{h_{2i-1}, c_{2i-1}\}$ (resp. $\{h_{2i-1}, c_{2i}\}$).

It remains now to connect the C-vertices to each other, without creating any alternative temporal paths from h_{2i} to h_{2i-1}, for any $i \in [n/2]$. For each $i \in [n/2]$, the edges between C-vertices should create temporal paths from c_{2i-1} and c_{2i} to any vertex c_j with index $j < 2i - 1$. On the other hand, they should not create any temporal $c_{2i} - c_{2i-1}$ paths, since then we would have a new temporal $h_{2i} - h_{2i-1}$ path. We introduce roughly n edges between C-vertices and carefully select their "small" labels in $[3/4, 1]$. Furthermore, to achieve temporal connectivity between all vertex pairs, we introduce an edge $\{c_{2i-1}, a_{2i-1}\}$ with the minimum time label ϵ and an edge $\{c_{2i}, a_{2i}\}$ with label $n + 1$, for each $i \in [n/2]$.

To complete the proof, we need to consider all possible types of ordered vertex pairs and to show that the temporal graph $\mathcal{G}$ is indeed connected. Moreover, since any subset of at least $5n$ edges includes some edges of $G[A]$, we can show that the removal of any edge from $G[A]$ with label i destroys the unique temporal path from h_{2i} to h_{2i-1}. ◀

We should highlight that increasing the label of edge $\{a_1, c_1\}$ from ϵ to 1, in the graph of Theorem 1, results in a temporal graph that admits a connectivity certificate of size $\Theta(n)$. Moreover, it is not hard to modify the construction of Theorem 1 so that all time labels of the edges are distinct, the temporal graph $\mathcal{G}$ is connected by strict temporal paths, and the removal of any subset of $5n$ edges results in a disconnected temporal graph. Therefore, the quadratic lower bound of Theorem 1 also applies to connectivity by strict temporal paths and improves on the lower bound of $\Omega(n \log n)$ in [1, Theorem 3].

4 The Approximability of Single-Source Temporal Connectivity

We proceed to study the approximability of Minimum Single-Source Temporal Connectivity. We show that the polynomial-time approximability of r-MTC is closely related to the

approximability of the classical Directed Steiner Tree (DST) problem and that r-MTC can be solved in polynomial-time for graphs of bounded treewidth.

4.1 A Lower Bound on the Approximability of r-MTC

We start with an approximation-preserving transformation from DST to r-MTC. The intuition is that we can use strict temporal paths to "simulate" the directed edges of DST.

▶ **Theorem 2.** *Any polynomial-time $\rho(n)$-approximation algorithm for r-*MTC *on simple temporal graphs implies a polynomial-time $\rho(n^2)$-approximation algorithm for* DST*.*

Proof sketch. We present an approximation-preserving transformation from the DST to r-MTC. Given an instance $I = (G(V, E, w), S, r)$ of DST with $|V| = n$, we construct a temporal graph $\mathcal{G}'$ with n^2 vertices so that (i) any Steiner tree connecting r to S in G can be mapped to an r-connected subgraph of $\mathcal{G}'$ with no larger cost; and (ii) given any r-connected subgraph of $\mathcal{G}'$, we can efficiently compute a feasible Steiner tree for I with no larger cost.

Each vertex $u \in V$ corresponds to a vertex u in the temporal graph $\mathcal{G}'$. For every directed edge $e = (u, v)$ of G, we create $n - 1$ strict temporal $u - v$ paths of length 2. Specifically, for every $u \in V$, $\mathcal{G}'$ contains auxiliary vertices z_i^u, for all $i \in [n-1]$, and temporal edges $\{u, z_i^u\}$ with time label i and weight 0. For every edge $e = (u, v) \in E$, $\mathcal{G}'$ contains temporal edges $\{z_i^u, v\}$ with time label $i + 1$ and weight $w(e)$, for all $i \in [n-1]$. Let $Z = \{z_i^u\}_{u \in V, i \in [n-1]}$ be the set of all auxiliary vertices. For every vertex $x \in Z \cup (V \setminus S)$, $x \neq r$, $\mathcal{G}'$ contains a temporal edge $\{r, x\}$ with time label $n + 1$ and weight 0. These edges ensure that r is connected to all non-terminal and auxiliary vertices at no additional cost. ◀

Directed Steiner Tree cannot be approximated within a ratio of $O(\log^{2-\varepsilon} n)$, for any constant $\varepsilon > 0$, unless NP $\subseteq$ ZTIME$(n^{\text{poly}\log n})$ [16, Theorem 1.2]. Theorem 2 implies that this inapproximability result carries over to r-MTC. Moreover, any polynomial-time $o(n^{\varepsilon})$-approximation algorithm for r-MTC would immediately improve the best known approximation ratio of the notoriously difficult DST problem.

4.2 An Approximation Algorithm for r-MTC

The following shows an approximation-preserving reduction from r-MTC to DST (see also the more general proof of Theorem 6). Then, we can use the algorithm of [7] and approximate r-MTC within a ratio of $O(n^{\varepsilon})$, for any constant $\varepsilon > 0$, in polynomial time, and within a ratio of $O(\log^3 n)$ in quasipolynomial time.

▶ **Theorem 3.** *Any polynomial-time $\rho(k)$-approximation algorithm for* DST *implies a polynomial-time $\rho(n)$-approximation algorithm for r-*MTC *on general temporal graphs.*

4.3 A Polynomial-Time Algorithm for Graphs with Bounded Treewidth

The following shows that r-MTC can be solved in polynomial time, by dynamic programming, if the underlying graph has bounded treewidth (see e.g., [11] about nice tree decompositions and dynamic programming algorithms for graphs of bounded treewidth).

▶ **Theorem 4.** *Let $\mathcal{G}$ be a temporal graph on n vertices with lifetime L, source vertex r and treewidth at most k. Then, there is a dynamic programming algorithm which given a nice tree decomposition of G, computes an optimal solution to r-*MTC *in time $O(nk^2 3^k (L + k)^{k+1})$.*

5 The Approximability of All-Pairs Minimum Temporal Connectivity

In this section, we study the approximability of the all-pairs version of Minimum Temporal Connectivity in general temporal graphs. Reducing MTC to r-MTC and to Directed Steiner Forest, we obtain polynomial-time approximation algorithms for MTC, albeit with not so strong guarantees (Corollary 5 and Theorem 6). To justify the poor approximation ratios, we reduce Symmetric Label Cover (SLC) to MTC and show that any $\rho(n)$-approximation for MTC implies a $\rho(n^2)$-approximation for SLC (Theorem 7). Moreover, using an approximation-preserving reduction from the Steiner Tree problem, we show that the unweighted version of MTC is APX-hard (Theorem 8).

5.1 Approximation Algorithms for MTC

Using every vertex of the temporal graph as a source vertex and taking the union of the solutions obtained by the algorithm of Theorem 3 for r-MTC, we obtain the following.

▶ **Corollary 5.** *For any constant $\varepsilon > 0$, there is a polynomial-time $O(n^{1+\varepsilon})$-approximation algorithm for* MTC *on temporal graphs with n vertices.*

Next, we present a reduction from MTC to Directed Steiner Forest (DSF) that leads to a different algorithm. Although the approximation ratio may be worse than $O(n^{1+\varepsilon})$ in general, this algorithm gives significantly better guarantees if the total number of temporal edges is quasilinear (and if the maximum degree of the underlying graph is polylogarithmic).

▶ **Theorem 6.** *Let $\mathcal{G}$ be a temporal graph with n vertices and M temporal edges such that the underlying graph has maximum degree Δ. Then, for any constant $\varepsilon > 0$, there is a polynomial-time $O(M^{\varepsilon}\min\{M^{4/5}, (\Delta M)^{2/3}\})$-approximation algorithm for* MTC *on $\mathcal{G}$. If $M = O(n \operatorname{poly}\log n)$, we obtain an approximation ratio of $O(n^{4/5+\varepsilon})$. If both $M = O(n \operatorname{poly}\log n)$ and $\Delta = O(\operatorname{poly}\log n)$, we obtain an approximation ratio of $O(n^{2/3+\varepsilon})$.*

Proof. The reduction from DSF to MTC is a generalized version of the reduction used in the proof Theorem 3. Let I be an instance of MTC consisting of an underlying graph $G(V, E)$, a finite set of time labels L_e for each edge e, and a weight $w(e, t)$ for any temporal edge (e, t). We show how to transform I into an instance I' of DSF so that (i) any feasible solution of I can be mapped to a feasible solution of I' with no larger total weight; and (ii) given a feasible solution of I', we can compute a feasible solution of I with no larger total weight.

For convenience, we denote H the edge-weighted directed graph of the DSF instance I'. For every temporal edge (e, t) of $\mathcal{G}$, H contains two vertices $h^1_{(e,t)}$ and $h^2_{(e,t)}$. Intuitively, $h^1_{(e,t)}$ indicates that we may use (e, t) and $h^2_{(e,t)}$ indicates that we actually use (e, t). For each edge $e \in E$, let $l_1(e) < l_2(e) < ... < l_k(e)$ be the time labels in L_e. For every $i \in [k-1]$, H contains a directed edge $(h^1_{(e,l_i(e))}, h^1_{(e,l_{i+1}(e))})$ with weight 0. Intuitively, these edges indicate that we can wait and use e at some later time up to $l_k(e)$. Moreover, for every $i \in [k]$, H contains a directed edge $(h^1_{(e,l_i(e))}, h^2_{(e,l_i(e))})$ with weight $w(e, l_i(e))$. This edge indicates that we actually use the temporal edge $(e, l_i(e))$ and incur the corresponding cost.

For every ordered pair of temporal edges (e_1, t_1), (e_2, t_2) of $\mathcal{G}$, such that $e_1 \neq e_2$, t_2 is the smallest time label in L_{e_2} such that $t_2 \geq t_1$ ($t_2 > t_1$, for strict connectivity), and e_1 and e_2 share a common endpoint, H contains a directed edge $(h^2_{(e_1,t_1)}, h^1_{(e_2,t_2)})$ with weight 0.

For every vertex $v_i \in V$, $i \in [n]$, H contains a pair of terminal vertices s_i and t_i. For every temporal edge (e, t) incident to v_i, H contains a directed edge $(s_i, h^1_{(e,t)})$ with weight 0 and a directed edge $(h^2_{(e,t)}, t_i)$ with weight 0. The set of connection requirements of the DSF instance I' consists of all pairs (s_i, t_j) for all $i, j \in [n]$ with $i \neq j$.

By construction, any temporal $v_i - v_j$ path, which consists of a temporal edge sequence $((e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k))$, corresponds to a directed $s_i - t_j$ path in H of the form

$$s_i, h^1_{(e_1,t_1)}, h^2_{(e_1,t_1)}, h^1_{(e_2,t''_2)}, h^1_{(e_2,t'_2)}, h^1_{(e_2,t_2)}, h^2_{(e_2,t_2)}, \ldots, h^1_{(e_k,t_k)}, h^2_{(e_k,t_k)}, t_j$$

with the same weight and vice versa. Using this observation, we can now establish claims (i) and (ii). Specifically, to show (i), we construct a feasible solution to I' that includes all directed edges of weight 0 and the directed edges $(h^1_{(e,t)}, h^2_{(e,t)})$ corresponding to the temporal edges (e, t) used in the feasible solution to I. Clearly, the two solutions have the same total weight and any temporal $v_i - v_j$ path in the solution to I corresponds to an $s_i - t_j$ path in the solution to I'. To show (ii), we first observe that any directed path from some s_i to some t_j should include some directed edges of the form $(h^1_{(e,t)}, h^2_{(e,t)})$ with weight $w(e, t)$. So, we construct a feasible solution to I that includes the temporal edges (e, t) corresponding to the positive-weight directed edges $(h^1_{(e,t)}, h^2_{(e,t)})$ included in the feasible solution to I'.

In the resulting DSF instance I', the total number of vertices is $O(n + M) = O(M)$ and the number of connection requirements is $O(n^2)$. If the maximum degree of the underlying graph is Δ, the total number of edges is dominated by the edges of the form $(h^2_{(e_1,t_1)}, h^1_{(e_2,t_2)})$, which are $O(\Delta M)$. Applying the approximation algorithm of [14, Theorem 1.1] to the DSF instance I', we obtain a polynomial-time $O(M^\varepsilon \min\{M^{4/5}, (\Delta M)^{2/3}\})$-approximation algorithm, for any constant $\varepsilon > 0$. In the special case where the number of temporal edges is $M = O(n\text{poly}\log n)$, we obtain an $O(n^{4/5+\varepsilon})$-approximation, for any constant $\varepsilon > 0$. If both $M = O(n\text{poly}\log n)$ and the maximum degree of the underlying graph $\Delta = O(\text{poly}\log n)$, we obtain a polynomial-time $O(n^{2/3+\varepsilon})$-approximation algorithm for any constant $\varepsilon > 0$. ◀

5.2 A Lower Bound on the Approximability of MTC

In this section, we present an approximation-preserving reduction from Symmetric Label Cover to MTC. Our reduction along with standard inapproximability results for Symmetric Label Cover indicate that MTC in general temporal networks is hard to approximate.

▶ **Theorem 7.** MTC *on temporal graphs with n vertices cannot be approximated within a factor of $O(2^{\log^{1-\varepsilon} n})$, for any constant $\varepsilon > 0$, unless* NP $\subseteq$ DTIME$(n^{\text{poly}\log n})$.

Proof. We present a polynomial-time approximation-preserving reduction from the Symmetric Label Cover (SLC) problem to MTC. In SLC (see e.g., [10, Definition 4.1]), we are given a complete bipartite graph $H(U, W)$, with $|U| = |W|$, a finite set of colors C and a binary relation $R(u, w) \subseteq C \times C$ for every vertex pair $(u, w) \in U \times W$. We seek to assign a color subset $\sigma(u) \subseteq C$ to each vertex $u \in U \cup W$ so that for every vertex pair $(u, w) \in U \times U$, there are colors $a \in \sigma(u)$ and $b \in \sigma(w)$ with $(a, b) \in R(u, w)$ and $\sum_{u \in U \cup W} |\sigma(u)|$, i.e., the total number of colors used, is minimized.

Given an instance of SLC, we create a temporal graph $\mathcal{G}$ whose vertex set V is partitioned into six sets V_U, $V_{C(U)}$, V_W, $V_{C(W)}$, V_X and $\{p, q\}$. There is a correspondence between the vertices of the bipartite graph H and the vertices of $\mathcal{G}$ in the sets V_U and V_W. The vertex sets $V_{C(U)} = V_U \times C$ and $V_{C(W)} = V_W \times C$ serve to encode the color assignment to the vertices of U and W in the SLC instance. Moreover, V_X contains a vertex (u, w, a, b) for every vertex pair $(u, w) \in U \times W$ and every allowable color pair $(a, b) \in R(u, w)$. Intuitively, the vertices of V_X serve to ensure that the color assignment is consistent. Finally, the vertices p and q ensure that the temporal graph $\mathcal{G}$ is connected.

For every $u \in V_U$ and $(u, a) \in V_{C(U)}$, $\mathcal{G}$ contains a temporal edge $\{u, (u, a)\}$ with label 1 and weight 1. Similarly, for every $w \in V_W$ and $(w, b) \in V_{C(W)}$, $\mathcal{G}$ contains a temporal

edge $\{w,(w,b)\}$ with label 4 and weight 1. For every vertex $(u,w,a,b) \in V_X$, $\mathcal{G}$ contains the temporal edges $\{(u,a),(u,w,a,b)\}$ with label 2 and weight 0 and $\{(u,w,a,b),(w,b)\}$ with label 3 and weight 0. $\mathcal{G}$ contains temporal edges with label 5 and weight 0 between p and every vertex in $V_U \cup V_{C(U)} \cup V_{C(W)} \cup V_X$ and between q and every vertex in V_U. Moreover, $\mathcal{G}$ contains temporal edges with label 0 and weight 0 between p and every vertex in V_W and between q and every vertex in $V_W \cup V_{C(U)} \cup V_{C(W)} \cup V_X$. We note that $\mathcal{G}$ is temporally connected and has $O(k^2c^2)$ vertices, where $k = |U| = |W|$ and $c = |C|$ (in fact, the number of vertices of $\mathcal{G}$ is of the same order as the total size of all binary relations $R(u,w)$).

We next show that this reduction is approximation-preserving. We first show that any feasible solution to the SLC instance can be mapped to a temporally connected subgraph $\mathcal{G}'$ of $\mathcal{G}$ with at most the same weight. Let us fix any assignment σ of a color set to each vertex of H that is feasible for the SLC instance. We first include in $\mathcal{G}'$ all temporal edges of weight 0. For every vertex $u \in U$ with assigned colors $\sigma(u)$, we include in $\mathcal{G}'$ the temporal edges $\{u,(u,a)\}$, for all $a \in \sigma(u)$. The total weight of these edges is $|\sigma(u)|$. Similarly, for every vertex $w \in W$, we include in $\mathcal{G}'$ the temporal edges $\{w,(w,b)\}$, for all $b \in \sigma(w)$. The total weight of these edges is $|\sigma(w)|$. Therefore, the total weight of the temporal subgraph $\mathcal{G}'$ is equal to the cost of the solution σ for the SLC problem.

It remains to show that $\mathcal{G}'$ is temporally connected. All vertices in $V_U \cup V_{C(U)} \cup V_{C(W)} \cup V_X \cup \{p\}$ are connected with each other (through p) by temporal edges with time label 5. There are also temporal $p-q$ and $q-p$ paths consisting of edges with time label 5 through the vertices of V_U. Similarly, all vertices in $V_W \cup V_{C(U)} \cup V_{C(W)} \cup V_X \cup \{q\}$ are connected with each other (through q) by temporal edges with time label 0. Moreover, there is a temporal path (using edges with time labels 0 and 5) from every vertex in $V_W \cup V_{C(U)} \cup V_{C(W)} \cup V_X \cup \{q\}$ to every vertex in V_U. Also, p is connected to every vertex in V_W with temporal edges of time label 0 and vice versa. All these vertex pairs are connected through temporal paths entirely consisting of 0-weight edges. The really interesting case concerns vertex pairs $(u,w) \in V_U \times V_W$. By the feasibility of the solution σ, for every vertex pair $(u,w) \in U \times W$, there are colors $a \in \sigma(u)$ and $b \in \sigma(w)$ such that $(a,b) \in R(u,w)$. Therefore, the temporal $u-w$ path $(u,(u,a),(u,w,a,b),(w,b),b)$ is included in $\mathcal{G}'$. Hence, $\mathcal{G}'$ is temporally connected.

We also need to show that given a temporally connected subgraph $\mathcal{G}'$ of $\mathcal{G}$, we can efficiently compute an assignment σ of a color set to each vertex in $U \cup W$ that is feasible for the SLC instance and has total cost no larger than the total weight of $\mathcal{G}'$. For every $u \in V_U$ and every temporal edge of the form $(\{u,(u,a)\},1)$ included in $\mathcal{G}'$, we include the color a in $\sigma(u)$. Similarly, for every $w \in V_W$ and every temporal edge of the form $(\{w,(w,b)\},4)$ included in $\mathcal{G}'$, we include the color b in $\sigma(w)$. Since these are the only edges of $\mathcal{G}$ (and $\mathcal{G}'$) with positive weight, the total cost of σ is equal to the total weight of $\mathcal{G}'$.

It remains to show that σ is a feasible solution to the SLC instance. Let $(u,w) \in U \times W$ in the SLC instance. The crucial observation is that the only way to connect $u \in V_U$ to $w \in V_W$ in $\mathcal{G}'$ is through some temporal path $(\{u,(u,a)\},1)$, $(\{(u,a),(u,w,a,b)\},2)$, $(\{(w,b),(u,w,a,b)\},3)$, $(\{w,(w,b)\},4)$, for $(a,b) \in R(u,w)$. This claim immediately implies the feasibility of the assignment σ. To prove this claim, we observe that a temporal $u-w$ path cannot use any temporal edge incident to p or q, since all edges between V_U and $\{p,q\}$ have time label 5 and all edges between V_W and $\{p,q\}$ have time label 0. So, any temporal $u-w$ path in $\mathcal{G}'$ has to move from u to some vertex $(u,a) \in V_{C(U)}$. Such a vertex $(u,a) \in V_{C(U)}$ does not have any neighbors in V_U other than u. Hence, the next vertex of any temporal $u-w$ path in $\mathcal{G}'$ must be to visit some $(u,w,a,b) \in V_X$, where $w \in W$ and $(a,b) \in R(u,w)$. Similarly, since such a vertex $(u,w,a,b) \in V_X$ does not have any neighbors in $V_{C(U)}$ other than (u,a) and any neighbors in $V_{C(W)}$ other than (w,b), we conclude that the next step

of any temporal $u - w$ path in $\mathcal{G}'$ must be to the vertex $(w, b) \in V_{C(W)}$. But now, the last temporal edge used has time label 3, which implies that the $u - w$ path cannot use any edges with time labels 1 and 2 anymore. Thus, the path cannot return to $V_U \cup V_{C(U)}$. The only choice now is that the path moves to w through the temporal edge $(\{w, (w, b)\}, 4)$, which establishes the claim about the structure of any temporal $u - v$ path in $\mathcal{G}'$.

The discussion above establishes the correctness of the reduction from SLC to MTC. Using the fact that the number of vertices of $\mathcal{G}$ is quadratic in the number of vertices of H and standard inapproximability results for SLC (e.g., [10]), we conclude the proof. ◀

Adjusting the proof of Theorem 7, we can get a reduction from the MinRep problem, which is considered in [8], to MTC. Thus, any polynomial-time $\rho(n)$-approximation for MTC on simple temporal graphs implies a polynomial-time $\rho(n^2)$-approximation for MinRep. Since the best known approximation ratio for MinRep is $O(n^{1/3} \log^{2/3} n)$ [8, Section 2], any $O(n^{1/6})$-approximation to MTC would imply an improved approximation ratio for MinRep.

5.3 Inapproximability of Unweighted MTC

The following shows an approximation-preserving reduction from the Steiner Tree problem on undirected graphs with edge weights either 1 or 2 to MTC on unweighted temporal graphs, where all temporal edges have weight equal to 1. Since this version of the Steiner Tree problem is known to be APX-hard [4], we obtain the following.

▶ **Theorem 8.** MTC *on unweighted temporal graphs is* APX*-hard, and thus it does not admit a PTAS, unless* P = NP.

6 All-Pairs Temporal Connectivity on Trees and Cycles

We can do better if the underlying graph is either a tree or a cycle. We can show that if the underlying graph is a tree, there is an optimal solution to the MTC problem that uses each edge with at most two time labels. Using this structural property, we can show that MTC can be solved efficiently by dynamic programming if the underlying graph is a tree.

▶ **Theorem 9.** *Let $\mathcal{G}$ be a temporal tree on n vertices with lifetime L. There is a dynamic programming algorithm that computes an optimal solution to* MTC *on $\mathcal{G}$ in time $O(nL^4)$.*

We also observe that if the underlying graph is a cycle $C_n = (v_0, v_1, \ldots, v_{n-1}, v_0)$, any temporally connected subgraph $\mathcal{G}'$ can be partitioned into *sectors*. A sector is a connected part $(v_i, v_{i+1}, \ldots, v_k)$ of the cycle for which there is a vertex $v_j \notin \{v_i, \ldots, v_{k-1}\}$ such that the temporal paths $p_{\text{incr}} = (v_i, v_{i+1}, \ldots, v_j)$ and $p_{\text{decr}} = (v_k, v_{k-1}, \ldots, v_{j+1})$ are present in $\mathcal{G}'$ (the vertex indices along C_n are taken modulo n). Intuitively, any vertex in the sector $(v_i, v_{i+1}, \ldots, v_k)$ can reach every vertex in C_n through the paths p_{incr} and p_{decr}. Then, we can show that there is an optimal solution to the MTC problem on C_n where each edge is shared by at most two different sectors. Then, ignoring edges shared by different sectors and using dynamic programming to determine a near optimal partitioning of C_n into sectors, we obtain the following.

▶ **Theorem 10.** *There is a polynomial-time* 2*-approximation algorithm for the* MTC *problem on any temporal cycle C_n.*

References

1. E.C. Akrida, L. Gąsieniec, G.B. Mertzios, and P.G. Spirakis. On Temporally Connected Graphs of Small Cost. In *Proc. of the 13th Workshop on Approximation and Online Algorithms (WAOA'15)*, volume 9499 of *LNCS*, pages 84–96, 2015.
2. E.C. Akrida, L. Gąsieniec, G.B. Mertzios, and P.G. Spirakis. Ephemeral networks with random availability of links: The case of fast networks. *Journal of Parallel and Distributed Computing*, 87:109–120, 2016.
3. K.A. Berman. Vulnerability of scheduled networks and a generalization of Menger's Theorem. *Networks*, 28(3):125–134, 1996.
4. M.W. Bern and P.E. Plassmann. The Steiner Problem with Edge Lengths 1 and 2. *Information Processing Letters*, 32(4):171–176, 1989.
5. K. Bhawalkar, S. Gollapudi, and K. Munagala. Coevolutionary opinion formation games. In *Proc. of the 45th ACM Symposium on Theory of Computing (STOC'13)*, pages 41–50, 2013.
6. A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems (IJPEDS)*, 27(5):387–408, 2012.
7. M. Charikar, C. Chekuri, T.-Y. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed steiner problems. *Journal of Algorithms*, 33(1):73–91, 1999.
8. M. Charikar, M. Hajiaghayi, and H.J. Karloff. Improved Approximation Algorithms for Label Cover Problems. *Algorithmica*, 61(1):190–206, 2011.
9. B. Chazelle. The dynamics of influence systems. In *Proc. of the 53rd IEEE Symposium on Foundations of Computer Science (FOCS'12)*, pages 311–320, 2012.
10. Y. Dodis and S. Khanna. Design Networks with Bounded Pairwise Distance. In *Proc. of the 31st ACM Symposium on Theory of Computing (STOC'99)*, pages 750–759, 1999.
11. R.G. Downey and M.R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.
12. C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E.Viola. On the Complexity of Information Spreading in Dynamic Networks. In *Proc. of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*, pages 717–736, 2013.
13. T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *Proc. of the 42nd International Colloquium on Automata, Languages and Programming (ICALP'15)*, volume 9134 of *LNCS*, pages 444–455, 2015.
14. M. Feldman, G. Kortsarz, and Z. Nutov. Improved approximation algorithms for Directed Steiner Forest. *Journal of Computer and System Sciences*, 78(1):279–292, 2012.
15. L. Foschini, J. Hershberger, and S. Suri. On the Complexity of Time-Dependent Shortest Paths. *Algorithmica*, 68(4):1075–1097, 2014.
16. E. Halperin and R. Krauthgamer. Polylogarithmic inapproximability. In *Proc. of the 35th ACM Symposium on Theory of Computing (STOC'03)*, pages 585–594, 2003.
17. P. Holme and J. Saramäki. Temporal Networks. *Physics Reports*, 519(3):97–125, 2012.
18. S. Huang, A.W.-C. Fu, and R. Liu. Minimum Spanning Trees in Temporal Graphs. In *Proc. of the 2015 ACM International Conference on Management of Data (SIGMOD'15)*, pages 419–430, 2015.
19. D. Kempe, J.M. Kleinberg, and A. Kumar. Connectivity and Inference Problems for Temporal Networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.
20. F. Kuhn, N.A. Lynch, and R.Oshman. Distributed computation in dynamic networks. In *Proc. of the 42nd ACM Symposium on Theory of Computing (STOC'10)*, pages 513–522, 2010.

21 G.B. Mertzios, O. Michail, I. Chatzigiannakis, and P.G. Spirakis. Temporal network optimization subject to connectivity constraints. In *Proc. of the 40th International Colloquium on Automata, Languages and Programming (ICALP'13)*, volume 7966 of *LNCS*, pages 657–668, 2013.

22 R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proc. of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing (DIALM-POMC'05)*, pages 104–110, 2005.

Improved Protocols and Hardness Results for the Two-Player Cryptogenography Problem*

Benjamin Doerr[1] and Marvin Künnemann[2]

1 **LIX, École Polytechnique, Palaiseau, France**
doerr@lix.polytechnique.fr
2 **Max Planck Institute for Informatics and Saarbrücken Graduate School of Computer Science, Saarbrücken, Germany**
marvin@mpi-inf.mpg.de

Abstract

The cryptogenography problem, introduced by Brody, Jakobsen, Scheder, and Winkler (ITCS 2014), is to collaboratively leak a piece of information known to only one member of a group (i) without revealing who was the origin of this information and (ii) without any private communication, neither during the process nor before. Despite several deep structural results, even the smallest case of leaking one bit of information present at one of two players is not well understood. Brody et al. gave a 2-round protocol enabling the two players to succeed with probability 1/3 and showed the hardness result that no protocol can give a success probability of more than 3/8.

In this work, we show that neither bound is tight. Our new hardness result, obtained by a different application of the concavity method used also in the previous work, states that a success probability of better than 0.3672 is not possible. Using both theoretical and numerical approaches, we improve the lower bound to 0.3384, that is, give a protocol leading to this success probability. To ease the design of new protocols, we prove an equivalent formulation of the cryptogenography problem as solitaire vector splitting game. Via an automated game tree search, we find good strategies for this game. We then translate the splits that occurred in this strategy into inequalities relating position values and use an LP solver to find an optimal solution for these inequalities. This gives slightly better game values, but more importantly, also a more compact representation of the protocol and a way to easily verify the claimed quality of the protocol.

Unfortunately, already the smallest protocol we found that beats the previous 1/3 success probability takes up 16 rounds of communication. The protocol leading to the bound of 0.3384 even in a compact representation consists of 18248 game states. These numbers suggest that the task of finding good protocols for the cryptogenography problem as well as understanding their structure is harder than what the simple problem formulation suggests.

1998 ACM Subject Classification G.2.m Miscellaneous

Keywords and phrases randomized protocols, anonymous communication, computer-aided proofs, solitaire games

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.150

1 Introduction

Motivated by a number of recent influential cases of whistle-blowing, Brody, Jakobsen, Scheder, and Winkler [2] proposed the following *cryptogenography problem* as model for anonymous information disclosure in public networks. We have k players (potential information leakers).

* An extended online version of this article can be accessed at [4].

43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016).
Editors: Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi;
Article No. 150; pp. 150:1–150:13

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

A random one of them holds a secret, namely a random bit. All other players only know that they are not the secret holder. Now without any non-public communication, the players aim at both making the secret public and hiding the identity of the secret holder. More precisely, we are looking for a (fully public) communication protocol in which the players as only form of communication broadcast bits, which may depend on public information (including all previous communication), private knowledge (with respect to the secret), and a private source of randomness. After this phase of communication, the protocol outputs a single bit depending solely on all data sent in the communication phase. The complete protocol (regulating the communication and the output function) and all communication is public, and is monitored by an eavesdropper who aims at identifying the secret owner. We say that a run of the protocol is a success for the players, if the protocol output is the secret bit and the eavesdropper fails to identify the secret owner; otherwise it is a success for the eavesdropper. Since everything is public, optimal strategies for the eavesdropper are easy to find (see below). We shall therefore always assume that the eavesdropper plays an optimal strategy. The players' success probability (for a given protocol) then is the probability (taken over the random decisions of the players and the random initial secret distribution) that simultaneously (i) the protocol outputs the true secret and (ii) an optimal eavesdropper does not blame the secret holder.

It is immediately clear that some positive (players') success probability is easy to obtain. A protocol without any communication and outputting a random bit achieves a success probability of $\frac{1}{2} - \frac{1}{2k}$ (the eavesdropper has no strictly better alternative than guessing a random player). Surprisingly, Brody et al. could show that the players, despite the complete absence of private communication, can do better. For two players, they present a protocol having a success probability of $\frac{1}{3}$ (instead of the trivial $\frac{1}{4}$). For k sufficiently large, they present a protocol with success probability 0.5644. They also show two hardness results, namely that a success probability of more than $\frac{3}{4}$ cannot be obtained, regardless of the number of players, and that $\frac{3}{8}$ is an upper bound for the two-player case. While all these results are easy to state, they build on deep analyses of the cryptogenography problem, in particular, on clever reformulations of the problem in terms of certain convex combinations of secret distributions (protocol design) and functions that are concave on a certain infinite set of two-dimensional subspaces ("allowed planes") of the set of secret distributions (hardness results).

The starting point for our work is the incomplete understanding of the two-player case. While the gap between upper and lower bound of $\frac{3}{8} - \frac{1}{3} \approx 0.04167$ is small, our impression is that the current-best protocol achieving the $\frac{1}{3}$ success probability in two rounds together with the abstract hardness result do not give us much understanding of the structure of the cryptogenography problem. We therefore imagine that a better understanding of this smallest-possible problem of leaking one bit from two players, ideally by determining an optimal protocol (that is, matching a hardness result), could greatly improve the situation.

Our Results. We shall be partially successful in achieving these goals. On the positive side, we find protocols with strictly larger success probability than $\frac{1}{3}$ (namely 0.3384) and we prove a stricter hardness result of 0.3672. Our new protocols look very different from the 2-round protocol given by Brody et al., in particular, they use infinite protocol trees (but have an expected finite number of communication rounds). These findings motivate and give new starting points for further research on the cryptogenography problem.

On the not so positive side, our work on better protocols indicates that good cryptogenographic protocols can be very complicated. The simplest protocol we found that beats the $\frac{1}{3}$

barrier already has a protocol tree of depth 16, that is, the two players need to communicate for 16 rounds in the worst case. While we still manage to give a human-readable description and performance proof for this protocol, it is not surprising that previous works not incorporating a computer-assisted search did not find such a protocol. Our best protocol, giving a success probability of 0.3384, already uses 18248 non-equivalent states.

Technical contributions. To find the improved protocols, we use a number of theoretical and experimental tools. We first reformulate the cryptogenography problem as a solitaire vector splitting game over vectors in $\mathbb{R}^{2\times k}_{\geq 0}$. Both for human researchers and for automated protocol searches, this reformulation seems to be easier to work with than the previous reformulation via convex combinations of distributions lying in a common allowed plane [2]. It also proved to be beneficial for improving upon the hardness result.

Restrictions of the vector splitting game to a finite subset of $\mathbb{R}^{2\times k}_{\geq 0}$, e.g., $\{0,\dots,T\}^{2\times k}$, can easily be solved via dynamic programming, giving (due to the restriction possibly sub-optimal) cryptogenographic protocols. Unfortunately, for $k = 2$ even discretizations as fine as $T = 40$ are not sufficient to find protocols beating the $1/3$ barrier and memory usage quickly becomes a bottleneck issue. However, exploiting the simple fact that the game values are homogeneous (that is, multiplying a game position by a non-negative scalar changes the game value by this factor), we can (partially) simulate a much finer discretization in a coarse one. This *extended dynamic programming approach* easily gives cryptogenographic success probabilities larger than $1/3$. Reading off the corresponding protocols, due to the reuse of the same position in different contexts, needs more care, but in the end gives without greater difficulties also the improved protocols.

When a cryptogenographic protocol reuses a state a second time (with a non-trivial split in between), then there is no reason to re-iterate this part of the protocol whenever this position occurs. Such a protocol allows infinite paths, while still needing only an expected finite number of rounds. Since the extended dynamic programming approach in finite time cannot find such protocols, we use a linear programming based post-processing stage. We translate each splitting operation used in the extended dynamic programming search into an inequality relating game values. By exporting these into an LP-solver, we do not only obtain better game values (possibly corresponding to cryptogenographic protocols with infinite paths, for which we would get a compact representation by making the cycles explicit), but also a way to easily certify these values using an optimality check for a linear program instead of having to trust the ad-hoc dynamic programming implementation.

Related work. Despite a visible interest of the research community in the cryptogenography problem, the only relevant follow-up work is Jakobsen's paper [5], which analyses the cryptogenography problem for the case that several of the players know the secret. This allows to leak a much larger amount of information as made precise in [5]. Due to the asymptotic nature of these results, unfortunately, they do not give new insight in the 2-player case. Other work on anonymous broadcasting typically assumes bounded computational power of the adversary (see, e.g., [6]); we refer to [3] for a survey on anonymous communication in communication networks.

In [2], the cryptogenography problem was reformulated to the problem of finding the point-wise minimal function f on the set of secret distributions that is point-wise not smaller than some given function g and that is concave on an infinite set of 2-dimensional planes. Such restricted notions of concavity (or, equivalently, convexity) seem to be less understood. We found work by Matoušek [9] for a similar convexity problem, however, with only a finite

number of one-dimensional directions in which convexity is required. We do not see how to extend these results to our needs.

2 Finding Better Cryptogenography Protocols

This section is devoted to the design of stronger cryptogenographic protocols. In particular, we demonstrate that a success probability of more than 1/3 can be achieved. We start by making the cryptogenography problem precise (Section 2.1) and introduce an equivalent formulation as solitaire vector splitting game (Section 2.2). We illustrate both formulations using the best known protocol for the 2-player case (Section 2.3). In Section 2.4, we state basic properties that simplify the analysis of protocols and aid our automated search for better protocols, which is detailed in Section 2.6. In Section 2.5, we give a simple, human-readable proof that 1/3 is not the optimal success probability by analyzing a protocol with success probability $\frac{449}{1334} \approx 0.3341$. We describe how to post-optimize and certify the results obtained by the automated search using linear programming in Section 2.7 and summarize our findings (in particular, the best lower bound we have found) in Section 2.8. Proofs and details that had to be omitted due to space constraints can be found in an extended version of this article [4].

2.1 The Cryptogenography Problem

Let us fix an arbitrary number k of players called $1, \dots, k$ for simplicity. We write $[k] := \{1, \dots, k\}$ for the set of players. We assume that a random one of the them, the "secret owner" $J \in [k]$, initially has a secret, namely a random bit $X \in \{0, 1\}$. The task of the players is, using public communication only, to make this random bit public without revealing the identity of the secret owner. More precisely, we assume that the players, before looking at the secret distribution, (publicly) decide on a communication protocol π. This is again public, that is, all bits sent are broadcast to all players, and they may depend only on previous communication, the private knowledge of the sender (whether he is the secret owner or not, and if so, the secret), and private random numbers of the sender. At the end of the communication phase, the protocol specifies an output bit Y (depending on all communication).

The aspect of not disclosing the identity of the secret owner is modeled by an adversary, who knows the protocol (because it was discussed in public) and who gets to see all communication (and consequently also knows the protocol output Y). The adversary, based on all this data, blames one player K. The players win this game if the protocol outputs the true secret (that is, $Y = X$) and the adversary does not blame the secret owner (that is, $K \neq J$), otherwise the adversary wins. It is easy to see what the best strategy for the adversary is (given the protocol and the communication), so the interesting part of the cryptogenography problem is finding strategies that maximize the probability that the players win assuming that the adversary plays optimally. We call this the (players') success probability of the protocol.

While the game starts with a uniform secret distribution, it will be useful to regard arbitrary secret distributions. In general, a *secret distribution* is a distribution D over $\{0, 1\} \times [k]$, where D_{ij} is the probability that player $j \in [k]$ is the secret owner and the secret is $i \in \{0, 1\}$. Modulo a trivial isomorphism, D is just a vector in $\mathbb{R}^{2\times k}_{\geq 0}$ with $\|D\|_1 = 1$. We denote by $\Delta = \Delta_k$ the set of all these distributions (this was denoted by $\Delta(\{0, 1\} \times [k])$ in [2]).

Brody et al. [2] observe that any cryptogenographic protocol can be viewed as successive rounds of one-bit communication, where in each step some (a priori) secret distribution probabilistically leads to one of two follow-up (a posteriori) distributions (depending on the bit transmitted) such that the a priori distribution is a convex combination of these and a certain proportionality condition is fulfilled (all three distributions lie in the same "allowed plane"). Conversely, whenever the initial distribution can be written as such a convex combination of certain distributions, then there is a round of a cryptogenographic protocol leading to these two distributions (with certain probabilities). Consequently, the problem of finding a good cryptogenographic protocol is equivalent to iteratively rewriting the initial equidistribution as certain convex combinations of other secret distributions in such a way that the success probability, which can be expressed in terms of this rewriting tree, is large.

2.2 The Solitaire Vector Splitting Game

Instead of directly using the "convex combination" formulation of Brody et al., we propose a slightly different reformulation as *solitaire vector splitting game*. This formulation seems to ease finding good cryptogenographic protocols (lower bounds for the success probability), both for human researchers and via automated search (Section 2.5). The main advantage of our formulation is that it takes as positions all $2k$-dimensional vectors with non-negative entries, whereas the cryptogenographic protocols are only defined on distributions over $\{0,1\} \times [k]$. In this way, we avoid arguing about probabilities and convex combinations and instead simply split a vector (resembling a secret distribution) into a sum of two other vectors. Furthermore, a simple monotonicity property (Proposition 2.5) eases the analyses. Still, there is an easy translation between the two formulations, so that we can re-use whatever results were found in [2].

For reasons of space, we do not repeat in detail the "convex combination" formulation and its equivalence to cryptogenographic protocols (the latter alone takes around two pages in [2]). We focus instead on the formal introduction of the vector splitting game (and argue that its equivalent to the "convex combination" formulation in Lemma 2.6) and refer to [2, 4] for a more detailed exposition.

▶ **Definition 2.1.** Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$. We say that (D_0, D_1) is a *j-allowed split of D* if $D = D_0 + D_1$ and D_0 (and thus also D_1) is proportional to D on $\{0,1\} \times ([k] \setminus \{j\})$, that is, there is a $\lambda \in [0,1]$ such that $(D_0)_{|\{0,1\}\times([k]\setminus\{j\})} = \lambda D_{|\{0,1\}\times([k]\setminus\{j\})}$. We call (D_0, D_1) an *allowed split* of D if it is a j-allowed split of D for some $j \in [k]$.

The objective of the vector splitting game is to recursively apply allowed splits to a given vector $D \in \mathbb{R}^{2\times k}_{\geq 0}$ with the target of maximizing the sum of the

$$p(D') := \max_{x\in\{0,1\}} \left(\sum_{j\in[k]} D'_{x,j} - \max_{j\in[k]} D'_{x,j} \right)$$

values of the resulting vectors (note that when D' is a distribution, then $p(D')$ is the 0-bit success probability of D'; for more details, the reader is referred to [2, 4]). More precisely, an n-round play of the vector splitting game is described by a binary tree of height at most n, where the nodes are labeled with *game positions* in $\mathbb{R}^{2\times k}_{\geq 0}$. The root is labeled with the initial position D. For each non-leaf v, the labels of the two children form an allowed split of the label of v. The payoff of such a play is $\sum_{D'} p(D')$, where D' runs over all leaves of the game tree. The aim is to maximize the payoff. Right from this definition, it is clear that

the maximum payoff achievable in an n-round game started in position D, the *value* of this game, is $\textsc{succ}_n(D)$ as defined below.

▶ **Definition 2.2.** For all $n \in \mathbb{N}$ and for all $D \in \mathbb{R}^{2\times k}_{\geq 0}$, we recursively define

(i) $\textsc{succ}_0(D) := \max\limits_{x\in\{0,1\}} \left(\sum\limits_{j\in[k]} D_{x,j} - \max\limits_{j\in[k]} D_{x,j} \right)$;

(ii) $\textsc{succ}_n(D) := \max\limits_{(D_0,D_1)} \left(\textsc{succ}_{n-1}(D_0) + \textsc{succ}_{n-1}(D_1) \right)$, if $n \geq 1$. Here the maximum is taken over all allowed splits (D_0, D_1) of D.

For an example of an admissible game, we refer to Figure 1 in Section 2.3.

It is easy to see that the game values are non-decreasing in the number of rounds, but bounded. The limiting value is thus well-defined.

▶ **Lemma 2.3.** *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ and $n \in \mathbb{N}$. Then $\textsc{succ}_n(D) \leq \|D\|_1$ and $\textsc{succ}_{n+1}(D) \geq \textsc{succ}_n(D)$. Consequently, $\textsc{succ}(D) := \lim_{n\to\infty} \textsc{succ}_n(D)$ is well-defined and is equal to $\sup_{n\in\mathbb{N}} \textsc{succ}_n(D)$.*

▶ **Proposition 2.4** (scalability)**.** *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ and $\lambda \geq 0$. Then $\textsc{succ}_n(\lambda D) = \lambda\, \textsc{succ}_n(D)$ for all $n \in \mathbb{N}$. Consequently, $\textsc{succ}(\lambda D) = \lambda\, \textsc{succ}(D)$.*

▶ **Proposition 2.5** (monotonicity)**.** *Let $D, E \in \mathbb{R}^{2\times k}_{\geq 0}$ with $E \geq D$ (component-wise). Then $\textsc{succ}_n(E) \geq \textsc{succ}_n(D)$ for all $n \in \mathbb{N}$. Consequently, $\textsc{succ}(E) \geq \textsc{succ}(D)$.*

From the previous definitions and observations, we derive that the game values for games starting with a distribution D, that is, $\|D\|_1 = 1$, and the success probabilities of the optimal cryptogenographic protocols for D, are equal.

▶ **Lemma 2.6.** *Let $D \in \mathbb{R}^{2\times k}_{\geq 0}$ with $\|D\|_1 = 1$. Then for all $n \in \mathbb{N}$, our definitions of $\textsc{succ}_n$ coincide with the ones of Brody et al., which are the success probabilities of the best n-round cryptogenographic protocols for the distribution D. Consequently, also the definition of $\textsc{succ}(D)$ coincides.*

2.3 Example: The Best-so-far 2-Player Protocol

We now turn to the case of two players. We use this subsection to describe the best known protocol for two players in the different languages. We also use this mini-example to sketch the approaches used in the following subsections to design superior protocols.

For two players, we usually write a game position $D = (D_{01}, D_{11}, D_{02}, D_{12}) \in \mathbb{R}^{2\times 2}_{\geq 0}$ as $D = (a, b, c, d)$. The 0-round game value (equaling the success probability of the 0-bit protocol) then is

$$\textsc{succ}_0(D) = \max\{\min\{a, c\}, \min\{b, d\}\}.$$

As a warmup, let us describe the best known 2-player protocol TwoBit in the two languages. In the language of Brody et al., the first player can send a (randomized) bit that transforms the initial distribution $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ with probability $\frac{1}{2}$ each into the distributions $(\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{6})$ and $(\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3})$. In the first case, the second player can send a bit leading to each of the distributions $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0)$ and $(\frac{1}{3}, \frac{1}{3}, 0, \frac{1}{3})$ with probability $\frac{1}{2}$, both having a 0-bit success probability of $\frac{1}{3}$. In the second possible result of the first move, the first player can lead to an analogous situation. Consequently, after two rounds of the protocol we end up with four equally likely distributions all having a 0-bit success probability of $\frac{1}{3}$. Hence the protocol TwoBit has a success probability of $\frac{1}{3}$.

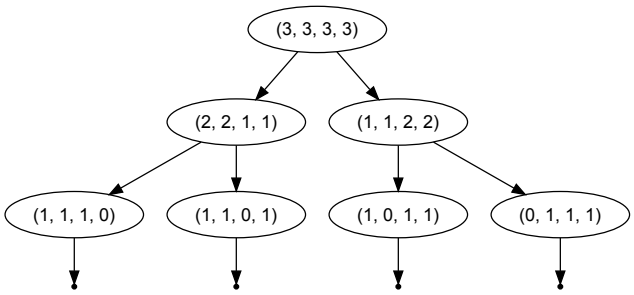


Figure 1 Game tree corresponding to TwoBit.

In the language of the splitting games, we can forget about the probabilities and simply split up the initial distribution. Using the scaling invariance, to ease reading we scaled up all numbers by a factor of 12. Figure 1 shows the game tree corresponding to the TwoBit protocol. It shows that $\text{SUCC}_2(3,3,3,3) \geq 4$, proving again the existence of a cryptogenographic protocol for the distribution $(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) = \frac{1}{12}(3,3,3,3)$ with success probability $\frac{4}{12} = \frac{1}{3}$.

Note that each allowed split (D^0, D^1) of D implies the inequality $\text{SUCC}(D) \geq \text{SUCC}(D^0) + \text{SUCC}(D^1)$, which follows from clause (ii) of Definition 2.2 and taking the limit $n \to \infty$. Hence the game tree giving the $\frac{1}{3}$ lower bound for the success probability equivalently gives the following proof via inequalities.

$$\begin{aligned} \text{SUCC}(3,3,3,3) &\geq \text{SUCC}(2,2,1,1) + \text{SUCC}(1,1,2,2), \\ \text{SUCC}(2,2,1,1) = \text{SUCC}(1,1,2,2) &\geq \text{SUCC}(1,0,1,1) + \text{SUCC}(0,1,1,1), \\ \text{SUCC}(1,0,1,1) = \text{SUCC}(0,1,1,1) &\geq \text{SUCC}_0(0,1,1,1) = 1. \end{aligned}$$

The splitting game and the inequality view will in the following be used to design stronger protocols (better lower bounds for the optimal success probability). We shall compute good game trees by computing lower bounds for the game values of a discrete set of positions via repeatedly trying allowed splits. For example, the above game tree for the starting position $(3,3,3,3)$ could have easily be found by recursively computing the game values for all positions in $\{0,1,2,3\}^4$.

It turns out that such an automated search leads to better results when we also allow scaling moves (referring to Proposition 2.4). For example, in the above mini-example of computing optimal game values for all positions $\{0,1,2,3\}^4$, we could try to exploit the fact that $\text{SUCC}(1,1,1,1) = \frac{1}{3}\text{SUCC}(3,3,3,3)$. Such scaling moves are a cheap way of working in $\{0,1,2,3\}^4$ while trying to gain the power of working in $\{0,1,\dots,9\}^4$, which would be computationally more costly, especially with regard to memory usage. Scaling moves may lead to repeated visits of the same position, resulting in cyclic structures. Here translating the allowed splits used in the tree into the inequality formulation and then using an LP-solver is an interesting approach (detailed in Section 2.7). It allows to post-optimize the game trees found, in particular, by solving cyclic dependencies. This leads to slightly better game values and compacter representations of game trees.

2.4 Useful Facts

For some positions of the vector splitting game, the true value is easy to determine. We do this here to later ease the presentation of the protocols.

▶ **Proposition 2.7.** *We have* $\textsc{succ}(a,b,c,d) \le \min\{a,c\} + \min\{b,d\}$.

▶ **Proposition 2.8.** *Let* $D = (a,b,c,0)$. *Then* $\textsc{succ}(D) = \textsc{succ}_0(D) = \min\{a,c\}$.

▶ **Proposition 2.9.** *If* $D = (a,b,c,d)$ *is such that* $a + b \le \min\{c,d\}$, *then* $\textsc{succ}(D) = a + b$.

2.5 Small Protocols Beating the 1/3 Barrier

We now present a sequence of protocols showing that there are cryptogenographic protocols having a success probability strictly larger than $\frac{1}{3}$. These protocols are still relatively simple, so we also obtain a human-readable proof of the following result.

▶ **Theorem 2.10.** $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \ge \frac{449}{1334} \approx 0.3341$.

Proof. To be able to give a readable mathematical proof, we argue via inequalities for game values $\textsc{succ}(\cdot)$. We later discuss how the corresponding protocols (game trees) look like.

We first observe the following inequalities, always stemming from allowed splits (the underlined entries are proportional). Whenever Proposition 2.8 or 2.9 determine a value, we exploit this without further notice.

$$\begin{aligned}\textsc{succ}(\underline{12,12},12,12) &\ge \textsc{succ}(\underline{7,7},6,4) + \textsc{succ}(\underline{5,5},6,8),\\ \textsc{succ}(\underline{5,5},6,8) &\ge \textsc{succ}(\underline{2,2},0,2) + \textsc{succ}(\underline{3,3},6,6) = 2+6=8.\end{aligned}$$

This proves $\textsc{succ}(12,12,12,12) \ge 8 + \textsc{succ}(7,7,6,4)$. To analyze $\textsc{succ}(7,7,6,4)$, we use the allowed split

$$\textsc{succ}(7,7,\underline{6,4}) \ge \textsc{succ}(4,5,\underline{3,2}) + \textsc{succ}(3,2,\underline{3,2}) \tag{1}$$

and regard the two positions $(4,5,3,2)$ and $(3,2,3,2)$ separately in some detail.

Claim 1: The value of $(4,5,3,2)$ satisfies $\textsc{succ}(4,5,3,2) \ge \frac{55}{12}$. By scaling, we have $\textsc{succ}(4,5,3,2) = \frac{1}{2}\textsc{succ}(8,10,6,4)$. We present the allowed splits

$$\begin{aligned}\textsc{succ}(\underline{8,10},6,4) &\ge \textsc{succ}(\underline{4,5},2,4) + \textsc{succ}(\underline{4,5},4,0) = \textsc{succ}(4,5,2,4) + 4,\\ \textsc{succ}(4,5,\underline{2,4}) &\ge \textsc{succ}(1,2,\underline{1,2}) + \textsc{succ}(3,3,\underline{1,2}) = \textsc{succ}(1,2,1,2) + 3,\end{aligned}$$

hence $\textsc{succ}(8,10,6,4) \ge \textsc{succ}(1,2,1,2) + 7$. To bound the latter term, we use the scaling $\textsc{succ}(1,2,1,2) = \frac{1}{6}\textsc{succ}(6,12,6,12)$ and consider the allowed splits

$$\begin{aligned}\textsc{succ}(\underline{6,12},6,12) &\ge \textsc{succ}(\underline{5,10},3,9) + \textsc{succ}(\underline{1,2},3,3) = \textsc{succ}(5,10,3,9) + 3,\\ \textsc{succ}(5,10,\underline{3,9}) &\ge \textsc{succ}(0,6,\underline{2,6}) + \textsc{succ}(5,4,\underline{1,3}) = 6+4=10.\end{aligned}$$

Thus $\textsc{succ}(6,12,6,12) \ge 13$ and $\textsc{succ}(1,2,1,2) \ge \frac{13}{6}$. This shows $\textsc{succ}(4,5,3,2) \ge \frac{1}{2}(\textsc{succ}(1,2,1,2)+7) \ge \frac{55}{12}$.

Claim 2: We have $\textsc{succ}(3,2,3,2) \ge \frac{5}{3} + \frac{2}{9}\textsc{succ}(7,7,6,4)$. By scaling, we obtain $\textsc{succ}(3,2,3,2) = \frac{1}{3}\textsc{succ}(9,6,9,6)$ and compute

$$\begin{aligned}\textsc{succ}(9,6,\underline{9,6}) &\ge \textsc{succ}(6,3,\underline{6,4}) + \textsc{succ}(3,3,\underline{3,2}),\\ \textsc{succ}(6,3,\underline{6,4}) &\ge \textsc{succ}(3,0,\underline{3,2}) + \textsc{succ}(3,3,\underline{3,2}) = 3 + \textsc{succ}(3,3,3,2),\end{aligned}$$

and hence $\textsc{succ}(9,6,9,6) \ge 3 + 2\,\textsc{succ}(3,3,3,2)$. To bound the latter term, we scale $\textsc{succ}(3,3,3,2) = \frac{1}{3}\textsc{succ}(9,9,9,6)$ and present the allowed splits

$$\begin{aligned}\textsc{succ}(\underline{9,9},9,6) &\ge \textsc{succ}(\underline{7,7},6,4) + \textsc{succ}(\underline{2,2},3,2),\\ \textsc{succ}(\underline{2,2},3,2) &\ge \textsc{succ}(\underline{1,1},1,0) + \textsc{succ}(\underline{1,1},2,2) = 1+2=3.\end{aligned}$$

Thus $\textsc{succ}(3,3,3,2) \ge 1 + \frac{1}{3}\textsc{succ}(7,7,6,4)$, implying $\textsc{succ}(3,2,3,2) \ge \frac{5}{3} + \frac{2}{9}\textsc{succ}(7,7,6,4)$.

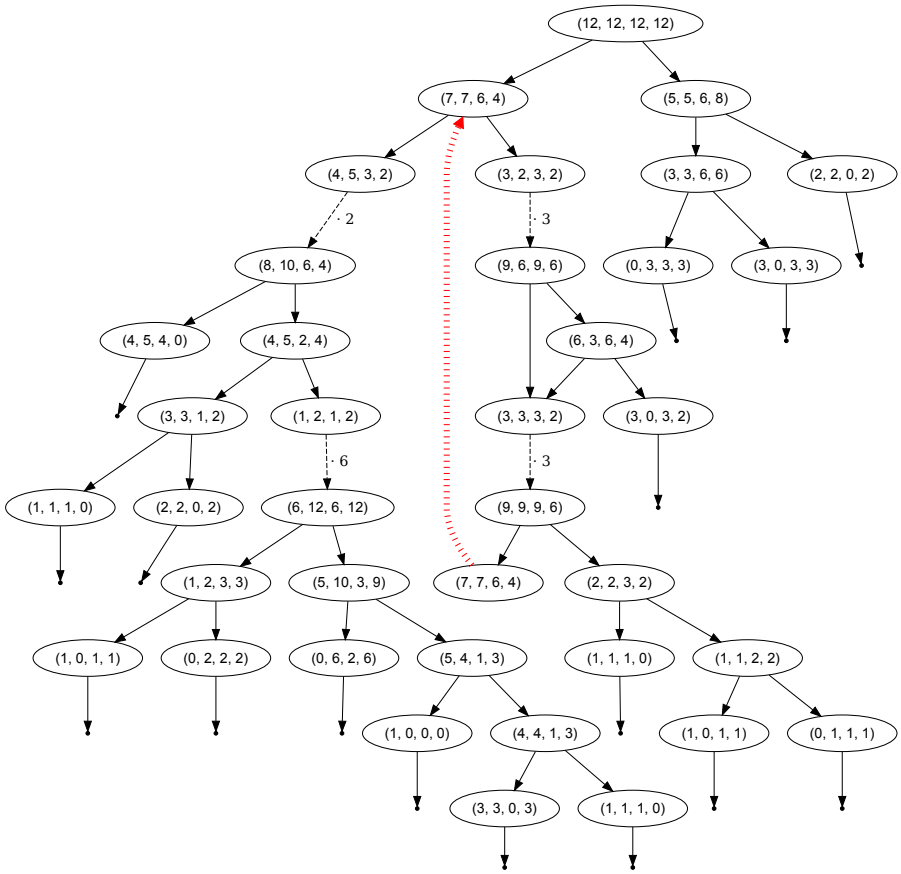


Figure 2 Game tree representation of the protocols of Theorem 2.10.

Putting things together. Claims 1 and 2 together with (1) give

$$\textsc{succ}(7,7,6,4) \geq \tfrac{75}{12} + \tfrac{2}{9}\,\textsc{succ}(7,7,6,4).$$

By solving this elementary equation, we obtain $\textsc{succ}(7,7,6,4) \geq \frac{225}{28}$, and it follows that $\textsc{succ}(12,12,12,12) \geq \frac{225}{28} + 8 = \frac{449}{28} = 16 + \frac{1}{28}$. Scaling leads to the claim of the theorem $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \geq \frac{1}{3} + \frac{1}{1344} = \frac{449}{1344} = 0.33407738\ldots$. ◀

When translating the inequalities into a game tree (see Figure 2 for the result), we first observe that in Claim 2 we obtained two different nodes labeled with the position $(3,3,3,2)$. Since there is no reason to treat them differently, we can identify these two nodes and thus obtain a more compact representation of the game tree. This is the reason why the node labeled $(3,3,3,2)$ in Figure 2 has two incoming edges.

Interestingly, such identifications can lead to cycles. If we translate the equations for position $(7,7,6,4)$ and its children into a graph, then we observe that the node for $(7,7,6,4)$ has a descendant also labeled $(7,7,6,4)$ (this is what led to the inequality $\textsc{succ}(7,7,6,4) \geq \frac{75}{12} + \frac{2}{9}\,\textsc{succ}(7,7,6,4)$). By transforming this inequality to $\textsc{succ}(7,7,6,4) \geq \frac{225}{28}$, we obtain a statement that is true, but that does not anymore refer to an actual (finite) game tree. However, there is a sequence of game trees with values converging to the value we determined. These trees are obtained from recursively applying the above splitting procedure for $(7,7,6,4)$ a certain number ℓ of times and then using the 0-round tree for the lowest node

Table 1 Lower bounds $s(T,\dots,T)/(4T)$ on $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})$ stemming from the automated search (line 1). Given are also the number of iterations until the automated search procedure converged, i.e., stopped finding improvements using allowed splits or scalings, and the number of game positions and constraints that had an influence on the value of $s(T,\dots,T)$.

T	15	20	25	30
Automated search	0.3369432925	0.3376146092	0.3379027186	0.3381689066
Iterations	119	129	141	146
Constraints	535	1756	4217	13958
Game Positions	394	1326	2956	9646

labeled $(7,7,6,4)$. The value of this game tree is $8+\sum_{i=0}^{\ell-1}(\frac{2}{9})^i\frac{75}{12}+(\frac{2}{9})^\ell\,\textsc{succ}_0(7,7,6,4)=8+\frac{75}{12}\frac{1-(\frac{2}{9})^\ell}{1-\frac{2}{9}}+6\cdot(\frac{2}{9})^\ell=\frac{449}{28}-\frac{57}{28}\,(\frac{2}{9})^\ell$. Hence for $\ell\geq 3$, this is more than 16 (which represents a success probability of $\frac{1}{3}$), corresponding to a game tree of height[1] $4+4\ell\geq 16$.

2.6 Automated Search

The vector splitting game formulation allows to search for good cryptogenographic protocols as follows. We try to determine the game values of all positions from a discrete set $\mathcal{D}:=\{0,\dots,T\}^{2\times k}$ by repeatedly applying allowed splits. More precisely, we store a function $s:\mathcal{D}\to\mathbb{R}$ that gives a lower bound on the game value $\textsc{succ}(D)$ of each position $D\in\mathcal{D}$. We initialize this function with $s\equiv\textsc{succ}_0$ and then in order of ascending $\|D\|_1$ try all allowed splits $D=D_0+D_1$ and update $s(D)\leftarrow s(D_0)+s(D_1)$ in case we find that $s(D)$ was smaller.

Recall that for any secret distribution D, the game value $\textsc{succ}(D)$ is the supremum success probability of cryptogenographic protocols for D. Hence, e.g., the value $s(T,\dots,T)/(2Tk)\leq\textsc{succ}(1/(2k),\dots,1/(2k))$ is a lower bound for this success probability. As we will discuss later, by keeping track of the update operations performed, we can not only compute such a lower bound, but also concrete protocols.

Since even for $k=2$, the size of the position space $\mathcal{D}$ and the number of allowed splits increase quickly with T, only moderate choices of T are computationally feasible, limiting the power of this approach drastically. However, using the scaling invariance $\lambda\textsc{succ}(D)=\textsc{succ}(\lambda D)$, we can introduce a scaling step: we iteratively optimize using allowed splits[2], then backpropagate the computed values (updating, e.g., $s(1,1,1,1)\leftarrow(1/T)\cdot s(T,T,T,T)$) to repeat the process. Surprisingly, this simple modification is sufficient to find protocols that are better than the previous best protocol TwoBit. For a more precise description of the algorithm, see the extended version of this article [4].

The success probabilities of the protocols computed following the above approach, using different values for T, are given in the first line of Table 1. Further results exploiting the post-optimization are given in Table 2 in Section 2.8.

2.7 Post-Optimization via Linear Programming

When letting the automated search also keep track of at what time which update operation was performed, this data can be used to extract strategies for the splitting game (and

[1] Note that the height refers to the number of transmitted bits and thus does not include the number of (virtual) scaling moves.

[2] In fact, we use *relaxed* splits, in which some coordinate in the resulting distributions may additionally be rounded down – this slightly increases the set of admissible splits. For details, see [4].

Table 2 Lower bounds $s(T,\dots,T)/(4T)$ on $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})$ stemming from the automated search only (line 1) and from the LP solution of the linear system extracted from the automated search data (line 2), when the number of iterations is restricted to 20.

T	30	35	40	45	50
Autom. search	0.3381086510	0.3381937725	0.3383218072	0.3383946540	0.3384414508
LP solution	0.3381527322	0.3382301900	0.3383547901	0.3384303130	0.3384736461
Iterations	20	20	20	20	20
Constraints	5373	8882	12410	18659	24483
Game states	4126	6789	9396	13992	18248

cryptogenographic protocols). Some care has to be taken to only extract those intermediate positions that had an influence on the final game value for the position we are interested in.

While this approach does deliver good cryptogenographic protocols, manually verifying the correctness of the updates or analyzing the structure of the underlying protocol quickly becomes a difficult task, as the size of the protocol grows rapidly. Fortunately, it is possible to output a compact, machine-verifiable certificate for the lower bound obtained by the automated search that might even prove a better lower bound than computed: Each update step in the automated search corresponds to a valid inequality of the form $\textsc{succ}(D) \geq \textsc{succ}(D_0) + \textsc{succ}(D_1)$, $\textsc{succ}(D) \geq \textsc{succ}_0(D)$ or $\lambda \cdot \textsc{succ}(D) = \textsc{succ}(\lambda \cdot D)$. We can extract the (sparse) set $\text{ineq}(T,T,T,T)$ of those inequalities that lead to the computed lower bound on $\textsc{succ}(T,T,T,T)$.

Consider replacing each occurrence of $\textsc{succ}(D')$ in the set of inequalities $\text{ineq}(T,T,T,T)$ found by the automated search by a variable $s_{D'}$. We obtain a system of linear inequalities S that has the feasible solution $s_{D'} = \textsc{succ}(D')$ (for every occurring vector D'). Hence in particular, the optimal solution of the linear program of *minimizing* s_D *subject to* S is a lower bound on $\textsc{succ}(D)$. It is easy to see that this solution is at least as good as the solution stemming from the automated search alone. It can, however, even be better, in particular when a game strategy yields cyclic visits to certain positions. Table 2 contains, for different values of T, the success probabilities found by automated search (run with a bounded iteration number of 20) and by this above linear programming approach. The table also contains the number of linear inequalities (and game positions) that were extracted from the automated search run. We observe that consistently the LP-based solution is minimally better. We also observe that the number of constraints is still moderate, posing no difficulties for ordinary LP solvers (which stands in stark contrast to feeding all allowed splits and scalings over the complete discretization to the LP solver, which quickly becomes infeasible).

Hence the advantage of our approach of extracting the constraints from the automated search stage is that it generates a much sparser sets of constraints that still are sufficiently meaningful. After solving the LP, we can further sparsify this set of inequalities by deleting all inequalities that are not tight in the optimal solution of the LP, since these cannot correspond to the best splits found for the corresponding vector D, yielding a smaller set of relevant inequalities, which might help to analyze the structure of strong protocols.

2.8 Our Best Protocol

We report the best protocol we found using the approach outlined in the previous sections.

▶ **Theorem 2.11.** *In the 2-player cryptogenography problem,* $\textsc{succ}(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}) \geq 0.3384736$.

Proof. On `http://people.mpi-inf.mpg.de/~marvin/verify.html`, we provide a linear program based on feasible inequalities on the discretization $\mathcal{D}$ with $T = 50$. To verify the

result, one only has to (1) check validity of each inequality, i.e., checking whether each constraints encodes a feasible scaling, allowed split or zero-bit success probability and (2) solve the linear program. Since we represent the distributions $D = (a, b, c, d)$ using a normal form $a \geq b, c, d$ (to break symmetries), checking validity of each splitting constraint is not completely trivial, but easy. We provide a simple checker program to verify validity of the constraints. The LP is output in a format compatible with the LP solver `lp_solve`[3]. ◀

3 A Stronger Hardness Result

In this section, we prove that any 2-player cryptogenographic protocol has a success probability of at most 0.3672. This improves over the previous 0.375 bound of [2].

▶ **Theorem 3.1.** *We have* $\text{SUCC}(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) \leq \frac{47}{128} = 0.367188$.

To prove the result, we apply the concavity method used by Brody et al. [2] which consists in finding a function s that (i) is lower bounded by SUCC_0 for all distributions and (ii) satisfies a certain concavity condition. Similar concavity arguments have been applied before in information complexity and information theory (see, e.g., [1, 7, 8]). We first relax the lower bound requirement to hold only for six particular simple distributions (namely $(1, 0, 0, 0), \ldots, (0, 0, 0, 1), (\frac{1}{2}, 0, \frac{1}{2}, 0)$ and $(0, \frac{1}{2}, 0, \frac{1}{2})$) instead of all distributions. This simplifies the search for a suitable stronger candidate function satisfying (i) - it remains to verify condition (ii) for the thus found candidate function.

More specifically, we adapt the upper bound function of Brody et al. [2] to

$$\begin{aligned} s(a, b, c, d) &:= \frac{1 - f(a, b, c, d)}{4}, \\ f(a, b, c, d) &:= a^2 + b^2 + c^2 + d^2 - 6(ac + bd) + 8abcd. \end{aligned}$$

In fact, we have changed their upper bound function by introducing an additional term of $8abcd$, which attains a value of zero on the distributions $(\frac{1}{2}, 0, \frac{1}{2}, 0)$, $(1, 0, 0, 0)$, etc., thus not affecting the zero-bit success probability condition of the concavity method. Due to space constraints, we defer the quite technical verification of the concavity condition to [4].

Since this function attains the value of $s(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) = \frac{47}{128}$ at the uniform distribution, we obtain the stronger upper bound of $\text{SUCC}(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}) \leq \frac{47}{128}$ of Theorem 3.1.

4 Conclusion

Despite the fundamental understanding of the cryptogenography problem obtained by Brody et al. [2], determining the success probability even of the 2-player case remains an intriguing open problem. The previous best protocol with success probability 1/3, while surprising and unexpected at first, is natural and very symmetric (in particular when viewed in the vector splitting game formulation). We disprove the hope that it is an optimal protocol by exhibiting less intuitive and less symmetric protocols having success probabilities up to 0.3384. Concerning hardness results, our upper bound of 0.3671875 shows that also the previous upper bound of 3/8 was not the final answer. These findings add to the impression that the cryptography problem offers a more complex nature than its simple description might suggest and that understanding the structure of good protocols is highly non-trivial.

[3] `http://lpsolve.sourceforge.net`

We are optimistic that our methods support a further development of improved protocols and bounds. (1) Trivially, investing more computational power or optimizing the automated search might lead to finding better protocols. (2) Our improved protocols might motivate to (manually) find infinite protocol families exploiting implicit properties and structure of these protocols. (3) Our reformulations, e.g., as vector splitting game, might ease further searches for better protocols and for better candidate functions for a hardness proof.

References

1 Mark Braverman, Ankit Garg, Denis Pankratov, and Omri Weinstein. From information to exact communication. In *Proc. 45th Annual ACM Symposium on Theory of Computing (STOC'13)*, pages 151–160. ACM, 2013.

2 Joshua Brody, Sune K. Jakobsen, Dominik Scheder, and Peter Winkler. Cryptogenography. In *Proc. 5th Conference on Innovations in Theoretical Computer Science (ITCS'14)*, pages 13–22, 2014.

3 George Danezis and Claudia Diaz. A survey of anonymous communication channels. Technical Report MSR-TR-2008-35, Microsoft Research, January 2008.

4 Benjamin Doerr and Marvin Künnemann. Improved Protocols and Hardness Results for the Two-Player Cryptogenography Problem. *ArXiv e-prints*, 2016. `arXiv:1603.06113`.

5 Sune K. Jakobsen. Information theoretical cryptogenography. In *Proc. 41st International Colloquium on Automata, Languages, and Programming (ICALP'14)*, volume 8572 of *Lecture Notes in Computer Science*, pages 676–688. Springer, 2014.

6 Sune K. Jakobsen and Claudio Orlandi. How to bootstrap anonymous communication. In *Proc. 7th Conference on Innovations in Theoretical Computer Science (ITCS'16)*, pages 333–344, 2016.

7 Nan Ma and Prakash Ishwar. The infinite-message limit of two-terminal interactive source coding. *IEEE Trans. Information Theory*, 59(7):4071–4094, 2013.

8 Nan Ma, Prakash Ishwar, and Piyush Gupta. Interactive source coding for function computation in collocated networks. *IEEE Trans. Information Theory*, 58(7):4289–4305, 2012.

9 Jiří Matoušek. On directional convexity. *Discrete & Computational Geometry*, 25(3):389–403, 2001.